# Network Warehouses: Efficient Information Distribution to Mobile Users

Arik Motskin, Stanford University
Ian Downes, Stanford University
Branislav Kusy, CSIRO ICT Centre
Omprakash Gnawali, Stanford University
Leonidas Guibas, Stanford University

***Abstract*—We consider the problem of distributing time-sensitive information from a collection of sources to mobile users traversing a wireless mesh network. Our strategy is to distributively select a set of well-placed nodes (*warehouses*) to act as intermediaries between the information sources and clusters of users. Warehouses are selected via the distributed construction of Hierarchical Well-Separated Trees (HSTs), which are sparse structures that induce a natural spatial clustering of the network.**

**Unlike many traditional multicast protocols, our approach is not data driven. Rather, it is agnostic to the number and position of sources as well as to the mobility patterns of users. Whereas source-rooted tree multicast algorithms construct a separate routing infrastructure to support each source, our sparse and flexible infrastructure is precomputed and efficiently reused by sources and users, its cost amortized over time. Moreover, the route acquisition delay inherent in on-demand wireless ad hoc network protocols is avoided by exploiting the HST addressing scheme. Our algorithm ensures with high probability a guaranteed stretch bound for the information delivery path, and is robust to lossy links and node failure by providing alternative HST-induced routes. Nearby users are clustered and their requests aggregated, further reducing communication overhead.**

## I. Introduction

Wireless network deployments are increasing in size and number throughout corporate and university campuses, hotels, and even cities. In many of these settings, the wireless network can be the default, and sometimes only, means of network connectivity. Most of these deployments are structured as a last-hop access to an underlying wired-network infrastructure. However, there is growing demand for a different class of wireless networks — multi-hop wireless mesh networks — which extend the reach of wireless connectivity without relying on an underlying wired infrastructure. Companies such as Meraki and Open-mesh produce devices to support these mesh networks, and projects like Funkfeuer in Austria and cities like Austin, Texas have deployed such networks to provide Internet access to their residents [1]. Such networks are seen as the only viable option to provide internet connectivity to rural areas of developing nations [17].

We study the problem of building an information delivery infrastructure on a wireless mesh network. The purpose of this infrastructure is to connect a set of static information sources residing in the network to a potentially evolving set of sinks via short, aggregated connecting paths. Typically, a set of mobile users, traveling through the geographic region spanned by the mesh network, connects to the network through nearby static mesh nodes. The users are interested in subscribing to data feeds from fixed information sources; the challenge is to maintain the connecting paths between the sources and the users, even as the users travel through the network, accessing the wireless mesh through an evolving set of proxy nodes. Real-time media applications such as voice and video conferencing can be challenging to accommodate in such networks, which are usually optimized for point-to-point routing and access. With a small number of sources and sinks, point-to-point (ex. compact) routing would suffice. However, with a significant number of simultaneous data streams and end users, it is critical to employ an approach — like multicast — that aggregates traffic, while efficiently handling the challenge of mobile users and link dynamics.

We seek an algorithm to maintain a connecting subgraph between sources and sinks with the following properties:

1) **Decentralized Computation**: Any connecting infrastructure must be built and stored by the nodes in a decentralized manner.
2) **Guaranteed Stretch**: Paths joining sources and sinks must have guaranteed constant-stretch bounds.
3) **Low Storage**: Each node should maintain a polylogarithmic-sized routing table.
4) **Aggregation**: The total number of edges in the connecting subgraph should be small in order to reduce total routing overhead.

To achieve these desiderata, in our approach the network nodes precompute a sparse and flexible infrastructure consisting of Hierarchical Well-Separated Trees (HSTs), which are overlay networks that induce natural spatial clusterings of the original network. Nodes of an HST are the nodes of the original network, but its edges are virtual and map to *paths* in the network. HSTs were originally developed as a tool to approximate arbitrary metrics using trees [2], [7] — given any pair of nodes, a single HST induces an $O(\log n)$ stretch path between the nodes in expectation ($n$ is the number of nodes).

We select a subset of nodes in the HSTs to be *warehouses*, which are intermediaries between the sources and the users. A warehouse, positioned close to a cluster of users, collects information directly from the source and then disseminates the data to all members of the cluster. If a cluster of users were to all appear in a small subtree of an HST, then the root of that subtree is a natural aggregation point, and a perfect choice for a warehouse.

Our protocol, which we call the *HST-based Protocol*, takes advantage of these spatial properties of the HST: a user,

interested in the data from a source $s$, *subscribes* to $s$ at a nearby warehouse; the warehouse collects and aggregates all subscriptions from nearby users, and sends a single subscription request to $s$. The source sends data via paths along an HST to the warehouse, which in turn disseminates information to the users via the paths of a smaller HST subtree. As a user moves through the network, the HSTs allow it to maintain connectivity to the source, without initiating new route discovery.

Unlike the data driven approaches of multicast or compact routing, the HST infrastructure is agnostic to where and how the many sources are positioned throughout the network, as well as to the mobility patterns of the users. The HST infrastructure is efficiently reused by sources and users over time; though storage and stretch remain low, aggregation is achieved by reusing warehouses across multiple sources simultaneously. Finally, the route acquisition delay inherent in on-demand protocols is avoided by efficiently exploiting the HST addressing system.

### A. Contributions

Our main contributions are as follows:

- We improve on previously known HST stretch bounds, reducing from logarithmic stretch (in expectation) in a single HST to constant stretch (w.h.p.) on $\log n$ HSTs, for networks with constant expansion rate.
- We design an information delivery algorithm using HSTs with the desired properties of **decentralized computation**, **stretch**, **storage** and **aggregation**.
- We verify in simulation that the proposed algorithm is viable in practical-sized wireless mesh networks.

### B. Modeling Assumptions

We assume that our network has *constant expansion rate* [11]. That is, there exists a constant $\alpha$ such that for any node $u$, $|B(u, 2r)| \leq \alpha |B(u, r)|$, where $|B(u, r)|$ counts the number of nodes within distance $r$ of $u$. For simplicity, in this paper we will generally be describing the construction properties of HSTs and proving theoretical stretch bounds with distance measured in terms of *hop count*; however, it is straightforward to show that all HST properties that we derive still hold in the more general expected transmission count (ETX) distance model [4]. Observe that, although the assumption of constant expansion rate is more restrictive than that of constant doubling dimension [13], it is still general enough to allow for holes and other irregularities in the network topology, and does not assume the restrictive unit-disk graph connectivity model, so there can exist long communication links. We assume that mobile users connect to the network via *proxy* static nodes. As a user moves, it connects to the network through different proxy nodes.

### C. Related Work

In this section, we survey the most closely related work in information delivery and routing (noting how they fare in our four desiderata), and in HSTs. See Table I for a summary of the properties of these methods and the HST-based Protocol.

*Multicast and Information Delivery:* The classic and well-studied approach to information delivery in wired and wireless networks is multicast communication. For example, a multicast per-source tree approach such as DVMRP [5] works by building reverse-shortest path trees between users and sources. While shortest-path tree multicast guarantees unit stretch between sources and users, it requires $\Omega(n)$ storage per node as a result. A sparse mode shared tree approach such as PIM-SM [6] utilizes a rendez-vous point (RP) core in the network, which acts as a meeting place between users and sources. Though storage is greatly reduced with RPs (each of RPs has a shortest-path tree), guaranteed stretch bounds are impossible when forced to route through a small number of RPs, unless the structure is augmented with additional source-rooted trees (which would result in the same expensive overhead drawbacks as shortest-path tree multicast). In the wireless ad-hoc community, where changing topology is the primary concern, on-demand protocols such as ODMRP [14] have been developed to manage dynamically changing routes. Since routes are computed in on-demand fashion, such protocols naturally suffer from route acquisition delay. Finally, approaches based on computing the minimum Steiner Tree [12], though effective at route aggregation, can result in arbitrarily bad stretch to *individual* users; moreover, the global nature of this approach results in expensive maintenance of the Steiner Tree infrastructure as users move.

*Low-state Routing:* In compact routing, the goal is to maintain low-stretch routes with small routing tables at each node [8], [15]. While stretch can be bounded at a constant (and experimentally, often very close to 1), state is generally polynomial [15], and aggregation of paths (with multiple sources and multiple sinks) is incidental.

*HSTs:* Study of HSTs initially focused on approximating metrics using trees [2], [3], [7], while more recent work examines the way HSTs can enable efficient resource management and matching in sensor networks [9].

Our plan for the remainder of the paper is as follows. In Section II, we describe the HST and its distributed construction. Section III contains our theoretical stretch results for the HST. In Section IV, we describe our main information delivery algorithm. We present simulation results in Section V, and conclude in Section VI.

## II. Hierarchical Well-Separated Trees

In this section, we describe the concept of the Hierarchical Well-Separated Tree (HST), the main building block of our information delivery infrastructure. The original motivation for the HST was given by Bartal [2], [3], who proposed the idea of approximating metrics using a distribution over trees. The advantage of using a tree structure derived from a network — as opposed to using the original network itself — is that trees are simpler objects to work with in a variety of applications, while distances can be guaranteed up to some provable approximation factor.

**Definition 1** *A weighted tree $T$, with root $r$, is a $\psi$-Hierarchical Well-Separated Tree ($\psi$-HST) if*

1) *all edges between a node and its children have the same weight*
2) *edge weights decrease by a factor $\psi$ along any root-to-leaf path*
3) *the number of edges in any root-to-leaf path is the same*

In general, according to Definition 1, the HST is an abstract object that need not have any vertices or edges in common with a baseline graph. On the other hand, the HST is most useful in the network context when constructed as a sparse overlay structure to some underlying graph. We will be concerned with building a HST $T$ so that nodes of $T$ correspond directly to the nodes of an underlying graph $G = (V, E)$ (where $|V| = n, |E| = m$), while edges of $T$ correspond to *paths* in $G$. The set of *leaf nodes* of $T$ corresponds to the original node set $V$. The *root* and *internal* nodes of $T$ will also correspond to specially chosen nodes of $V$, although it will be useful to think of an internal node $u$ as being in charge of a *cluster* of nodes (in particular, the set of leaf nodes in the subtree rooted at $u$). Thus, a node $v \in V$ appears at least once in $T$ as a leaf, and potentially more times in higher levels of $T$.

We say that an internal node $u$ in $T$ is at *level* $i$ if the path from $u$ to a leaf in its subtree is of length $i$. The root $r$ resides at level $\delta$ and the leaf nodes at level 0. Note that, by Definition 1, the value $\delta$ is unconstrained, although the 2-HST construction that we will use [7], [9] implies $\delta = O(\log n)$. For specificity, in this paper we will be concerned with 2-HSTs; throughout, this value of $\psi$ will be assumed unless stated otherwise. In our 2-HST, the weight of an edge joining a level $i$ and $i-1$ node will be $2^i$.

For any two nodes $u, v \in V$, we define the metric $d(u, v)$ as measuring the length of the shortest path between $u$ and $v$ in $G$ (in hop counts)[1]. The metric $d^T(u, v)$ measures the length of the path between leaf nodes $u$ and $v$ in HST $T$, defined as the sum of the edge weights along the unique tree path joining these leaf nodes. Fakcharoenphal *et al.* [7] show that any $n$-vertex graph can be approximated by a distribution over HSTs built over the graph, where the expected distortion is $O(\log n)$. That is, for any two nodes $u, v$ in the original graph $G$, $E[d^T(u, v)] \leq O(\log n)d(u, v)$. Here, we say that $d^T$ $O(\log n)$*-probabilistically approximates* $d$.

### A. How to Construct an HST

To build the randomized HST so that it $O(\log n)$-*probabilistically approximates* the metric of $G$, Fakcharoenphal *et al.* [7] take a centralized top-down approach, constructing successive nested partitions of the node set $V$. Gao *et al.* [9] demonstrated a distributed HST-building procedure using a bottom-up approach. They use successive floods of increasing radius that allow leaf nodes to determine their ancestors in the HST. It should be noted that both the top-down and bottom-up approaches produce an identical HST. An example of two HSTs built on the same set of nodes can be found in Figure 1.

*1) Signature Computation:* The key step in the HST construction is that each node $v$ computes a $O(\log n)$-length *signature* $S(v)$. This identifying address encodes the unique sequence of each node's ancestors in the HST. We first fix a permutation $\pi : V \to \{1, 2, \ldots, n\}$, selected uniformly at random from the set of all permutations on $n$ objects. For a node $u$, we call $\pi(u)$ the *rank* of $u$. Next, a parameter $\beta$ is selected uniformly at random from the interval $[1/2, 1]$. For each integer $i$ in the range $[1, \delta + 1]$, where $\delta = \lceil \log_2 D \rceil$ and $D$ is the diameter of $G$, we set $\beta_i = \beta \cdot 2^i$. Then, for each such $i$, we define $S(v)_i$, the $i^{\text{th}}$ element of $v$'s signature vector, as the node with the smallest rank in the ball of hop-count radius $\beta_i$ centered at $v$. More formally, for each $i$,

$$S(v)_i = \underset{u \in B(u, \beta_i)}{\arg\min}\ \pi(u)$$

We call $S(v)_i$ the *level* $i$ *ancestor* of $v$, and define $S(v)_0 = v$ for all $v \in V$. Observe that for node $w \in V$ such that $\pi(w) = 1$, $w$ must be the level $\delta$ ancestor of all nodes, since it is ranked the lowest in the graph. Each signature vector is unique (since $S(u)_0 = u$ for all $u \in V$), and a node can appear as both the level $i$ and level $j$ ancestor for another node, even if $i \neq j$. Moreover, observe that two nodes $u_1$ and $u_2$ often have common ancestors at a particular level; intuitively, this is more likely if $u_1$ and $u_2$ are close to one another (though not guaranteed, even if the nodes are neighbors).

However, note that even if $S(u_1)_i = S(u_2)_i = z$, we may have that $S(u_1)_{i+1} \neq S(u_2)_{i+1}$. In such a case, when reconstructing the HST tree, we *cannot* have $u_1$ and $u_2$ appear as leaf nodes in the same subtree rooted at level $i$ node $z$, since their level $i+1$ ancestor is different. In this case, the node $z$ would appear *multiple times* in the HST at level $i$. Therefore, we require a post-processing [16] step to accurately reconstruct the HST given the set of signature vectors[2].

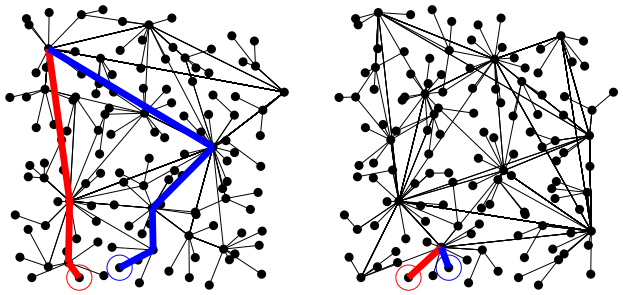
The formal proof that we have indeed now constructed a 2-HST with $O(\log n)$ stretch is given in [7]. However, observe that given leaf nodes $u, v$, finding $d^T(u, v)$ requires computing the *least common ancestor (lca)* of $u$ and $v$ in $T$, which can be found by identifying the smallest index $i$ such that $S(u)_k = S(v)_k$ for all $k \geq i$. Suppose $lca(u, v) = w$ is at level $i$. Then

$$d^T(u, v) = 2 \cdot \textstyle\sum_{j=1}^{i} 2^j = 2 \cdot (2^{i+1} - 2)$$

On the other hand, as $w$ is a level $i$ ancestor to both $u$ and $v$, $d(u, w), d(v, w) \leq \beta \cdot 2^i \leq 2^i$, so by the triangle inequality $d(u, v) \leq 2^{i+1}$. Therefore, $d(u, v) \leq d^T(u, v)$, while the stretch result [7] gives an upper bound that $d^T(u, v) \leq O(\log n)d(u, v)$.

[1]Throughout this section, we use the hop-count distance for simplicity of presentation. However, the HST construction and stretch bounds are also valid for the ETX metric, which measures the distance between two nodes as the expected number of transmissions to reach one from the other.

[2]In this and previous works on distributed HST construction [9], simply computing the set of signature vectors is sufficient to guarantee the stretch bounds. It is in fact not necessary to reconstruct the entire HST with this post-processing step.



(a) Single HST (b) Best of log(n) HSTs

Fig. 1: Two HSTs built on the same network: (a) the shortest path between two nodes can be significantly distorted by a single HST, (b) the best tree among $\log n$ HSTs induces an $O(1)$-stretch route (w.h.p).

### B. Communication and Storage of HST Construction

Gao *et al*. [9] give a distributed algorithm, consisting of specially designed floods of increasing radius, in order to compute the 2-HST. Each node in the graph computes its signature vector — its $O(\log n)$ list of ancestors at increasing levels of the tree. Moreover, each node computes pointers that encode the paths to its ancestors and descendants in the HST[3]. The total communication cost of constructing the 2-HST in this manner (aggregate message complexity of the floods) is $O(\log n)$ per node in expectation (randomness is over the initial permutation $\pi$ and parameter $\beta$). With regards to storage, the total requirement is $O(\log n)$ per node in expectation; this quantity comprises an $O(\log n)$ length signature vector of node IDs, as well as all next-hop pointers for encoding the paths to ancestors and descendants.

## III. Theoretical Stretch Bounds

In this section we show that by using multiple, independent HSTs in a graph $G$ with constant expansion rate, any cluster of nodes will all be contained with high probability within a small subtree in one of the HSTs of asymptotically optimal height. This property is interesting because, given a single HST $T$ for graph $G$, it is not particularly unlikely for *neighboring* nodes in $G$ to appear in far-flung areas of $T$ (see Figure 1); the exciting aspect of this result is that seemingly so few HSTs are necessary to establish a thorough covering of the network. This result is critical to our information delivery scheme because, as a corollary, while a single HST provides logarithmic stretch in expectation for the network [7], [9], we show in Section III-A that $O(\log n)$ HSTs induce with high probability a *constant*-stretch path between any two points in the network.

Throughout this section, assume we have a graph $G$ with constant expansion rate $\alpha$. We adapt proof techniques from Fakcharoenphal [7] to bound the probability that a cluster of nodes in $G$ has a least common ancestor that is *not too high* up one of the HSTs.

**Lemma 1** *Consider a ball $B = B(p, r)$, centered at node $p$, with hop count radius $r$. Let $d \leq 2r$ denote the diameter of $B$. Assume there exists a fixed parameter $\beta$ selected uniformly at random from the interval $[1/2, 1]$. Finally, consider a single 2-HST $T$, built as in Section II. A node $w$ is called a* center *at level $i$, if it appears on the $i^{th}$ level of $T$, meaning that for some node $v$, $w$ is the lowest ranked node in the ball $B(v, \beta \cdot 2^i)$.*

*Then, with probability bounded by $1/2$, all nodes in $B$ appear as leaf nodes in a subtree of $T$ rooted at a node $w_s$ at level $\lceil \log_2 d \rceil + O(1)$. This means that, with probability bounded by $1/2$, all nodes in $B$ have a common ancestor $w_s$ in the HST which is at graph distance $O(d)$ from all of them.*

*Proof:* We say that level $i$ of the HST *separates* the ball $B$ if nodes in $B$ are assigned to different ancestors in level $i$. We say that a center *settles* the ball $B$ at level $i$ if it is the lowest-ranked center to which at least one node in $B$ is assigned, and it *cuts* the ball $B$ in level $i$ if it settles $B$, and at least one node in $B$ is *not* assigned to that center. Let $j^*$ be the smallest integer such that $d \leq 2^{j*}$, and let $c$ be a fixed constant. Denote $k = j^* + 1 + c$. We examine the probability that level $k$ separates $B$.

A center at level $k$ uses the radius $\beta_k = \beta \cdot 2^k$. Consider a node $w_s$ that potentially *cuts* ball $B$, with $d(w_s, p)$ denoting the distance to the center of $B$. Observe that if $d(w_s, p) \leq \beta_k - r$, then $w_s$ can certainly not cut $B$, since $w_s$ will be contained in the ball of radius $\beta_k$ for all nodes in $B$. Similarly, if $d(w_s, p) \geq \beta_k + r$, then $w_s$ cannot cut $B$ since no node of $B$ can be contained in the ball of radius $\beta_k$.

Therefore, for it to be even possible for $w_s$ to cut $B$, we need to have that $d(w_s, p) \in [\beta_k - r, \beta_k + r] \subset [2^{k-1} - r, 2^k + r]$. Assuming that $w_s$ lies in this annulus $A$ around $p$, observe next that $\beta_k = \beta \cdot 2^k$ is chosen uniformly at random from the interval $\mathcal{I} = [2^{k-1}, 2^k]$ (due to the random parameter $\beta \in [1/2, 1]$). Therefore, $w_s$ will only cut $B$ if $\beta_k$ falls into the narrow bad subinterval (of length $d$) of $\mathcal{I}$ (which is of length $2^{k-1}$). In particular, the probability that $\beta_k$ falls into the bad interval is bounded by $d/2^{k-1} \leq 2^{-c}$.

If node $w_s$ did in fact cut the ball, then some node $v \in B$ is assigned to it. Then certainly $w_s$ must have the smallest rank in the region $B(v, \beta_k)$, where $\beta_k \geq 2^{k-1}$. This happens with probability at most $1/|B(v, 2^{k-1})|$ (taken over the random node rank permutation), where the notation $|B(\cdot, \cdot)|$ indicates the number of nodes in the ball $B(\cdot, \cdot)$. Therefore,

$$\mathbb{P}(w_s \text{ cuts } B) \leq \frac{2^{-c}}{|B(v, 2^{k-1})|}.$$

Finally, observe that

$$\mathbb{P}(\text{level } k \text{ separates } B) \leq \sum_{w_s \in A} \frac{2^{-c}}{|B(v, 2^{k-1})|}$$

By the definition of annulus $A$, we have that for any node $a \in A$, $d(v, a) \leq (2^k + r) + r < 2^{k+1}$, so the number of nodes in $A$ is certainly bounded by $|B(v, 2^{k+1})|$. Therefore,

$$\begin{aligned}\mathbb{P}(\text{level } k \text{ separates } B) &\leq |B(v, 2^{k+1})| \cdot \frac{2^{-c}}{|B(v, 2^{k-1})|} \\ &\leq \alpha^2 \cdot 2^{-c}\end{aligned}$$

where the last inequality follows by the assumption of constant expansion rate $\alpha$. For fixed $\alpha$, there exists large enough constant $c$ such that $\mathbb{P}(\text{level } k \text{ separates } B) \leq 1/4$.

[3]Recall that an edge in the HST corresponds to a path in the original graph.

Note that, by the exact same proof and for the same value of $c$, it follows that for any $m > 0$,

$$\mathbb{P}(\text{level } k+m \text{ separates } B) \leq \frac{1}{4} \cdot \left(\frac{1}{2}\right)^m$$

If we denote by $\delta$ the top level of the HST, then

$$\begin{aligned}\mathbb{P}(\text{level } i \text{ separates } B \text{ for } i \geq k) &\leq \sum_{i=k}^{\delta} \frac{1}{4} \cdot \left(\frac{1}{2}\right)^{i-k} \\ &\leq \frac{1}{2}\end{aligned}$$

■

Next, we show that by building $O(\log n)$ independent HSTs, at least one of these trees will have the entire cluster appearing as leaves within the same small subtree.

**Theorem 1** *Suppose we construct $\lceil \log_2 n \rceil$ 2-HSTs independently on a graph with constant expansion rate. Then with high probability, all nodes in a ball $B$ of hop-count diameter $d$ appear as leaves in a subtree of one of the HSTs rooted at a node $w$ at level $\lceil \log_2 d \rceil + O(1)$. Therefore, w.h.p, all nodes in $B$ have a common ancestor $w$ in one of the HSTs which is at hop-count distance $O(d)$ from all of them.*

*Proof:* For each of the $\lceil \log_2 n \rceil$ HSTs, by the Lemma, the probability that the nodes of $B$ do *not* appear in the same subtree of some node at level $\lceil \log_2 d \rceil + O(1)$ is bounded by $1/2$. On the other hand, the probability that the nodes of $B$ do not appear in such a subtree in any of the $\lceil \log_2 n \rceil$ HSTs is less than or equal to $(1/2)^{\lceil \log_2 n \rceil} = O(1/n)$. ■

### A. Point-to-Point Routing

A critical element of the information delivery scheme described in Section IV is the ability to quickly compute a constant stretch route between a warehouse and an information source. Despite the sparsity of the HST infrastructure, we can achieve a constant stretch, in the sense that, for any nodes $u, v$, the distance of traveling along the paths induced by the *best* of the $\log n$ HSTs is bounded by a constant factor (with high probability) from the shortest path in $G$. The *best* HST is the one which has the shortest leaf-to-leaf tree distance. The proof of the following Corollary follows directly from Theorem 1, in that any two nodes with shortest path distance $d$ can be considered to be in a cluster of diameter $d$.

**Corollary 1** *Given nodes $u, v \in G$ with $d(u,v) = d$, with $G$ having constant expansion rate, and a set $T^*$ of $\log n$ independently built HSTs on $G$, there exists with high probability at least one HST $T \in T^*$ such that the path in $G$ induced by the $u$ to $v$ leaf-to-leaf path in $T$ is of length $O(d)$.*

Figure 2 shows that in practice, the stretch of this HST-induced path (i.e., the constant hidden in the $O(\cdot)$ notation of Corollary 1) is quite small.

In Section II, we defined a node's HST signature as its length $O(\log n)$ list of ancestors. Given $\log n$ HSTs, each node is therefore endowed with a $O(\log^2 n)$ length address,

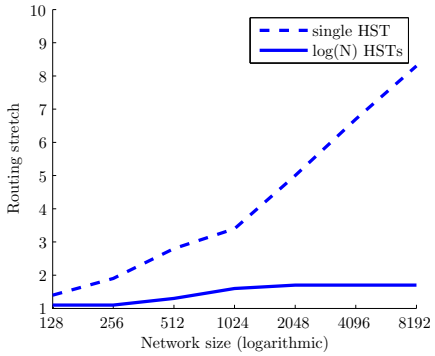



Fig. 2: Comparison of the stretch with a single HST and $\log n$ HSTs in the perturbed grid topology. The graph gives the $99^{th}$ percentile of the path stretch for 1000 randomly selected pairs of nodes (separated by 10 hops). While the path stretch grows when only a single HST is used, a $\log n$ number of HSTs is sufficient to guarantee that a constant stretch path exists in one of the HSTs with high probability.

comprising a signature for each HST. We now show how to compute the best HST for a pair of nodes $u$ and $v$ given these addresses. Given $v$'s address, $u$ can compute the *best* HST over which to route by identifying the HST which induces the *lowest least common ancestor* of leaf nodes $u$ and $v$; Corollary 1 guarantees that an HST with a constant stretch ancestor exists with high probability. The *lca* computation requires $\log n$ string comparisons, costing $O(\log n)$ for each of the $O(\log n)$ pairs of HST signature vectors. Recall from Section II that, as part of the $O(n \log n)$ aggregate storage cost of an HST, each node stores pointers that realize the paths represented by HST edges. Thus, $u$ can route up the best HST to the *lca*, which then routes back down the tree to $v$.

## IV. Information Delivery to Mobile Users

In this section we present the details of the HST-based Protocol. We assume that the network consists of static nodes that provide routing infrastructure for information sources and mobile users. Users connect to the network through nearby proxy static nodes.

Our approach is to employ a set of *warehouses*, positioned throughout the network, to act as intermediaries between the information sources and the users: warehouses aggregate subscription requests from nearby users, and then communicate directly with the information sources. Given a collection $C$ of users moving through the network as a tight cluster, we observe that, from a cost perspective, it is advantageous for all members of $C$ to be served at any given time by the same nearby warehouse. The routing infrastructure consists of multiple HSTs that span the network and warehouses are selected from among the internal nodes of the HSTs. If all of the members of a cluster appear in a small subtree in some HST, then the root of that subtree is a natural choice to serve as the cluster's warehouse. We describe below precisely how users and sources connect via the network warehouse system and how we address their loss of connectivity due to mobility or network dynamics. We then show how the theoretical results

of Section III ensure that these user-to-warehouse-to-source paths are of constant stretch with high probability.

### A. Algorithm

Initially, the network constructs a set $\mathcal{T}$ of $\lceil \log_2 n \rceil$ independent 2-HSTs, according to the distributed procedure of Section II-B. The $O(n \log^2 n)$ cost of this construction is amortized over the lifetime of the network as the procedure needs to be done only once. The candidate warehouses are all nodes residing at level $d^*$ of the HSTs, where $d^* \in [0, \lceil \log_2 n \rceil + 1]$ is an integer parameter in the model. That is, if a node $u \in V$ is a level $d^*$ ancestor in some HST $T \in \mathcal{T}$, then $u$ is considered a candidate warehouse. Observe that the larger the parameter $d^*$, the fewer warehouses there are; $d^* = 0$ implies that the leaf nodes in every HST are candidates, while $d^* = \lceil \log_2 n \rceil + 1$ means the roots of the HSTs are candidates. The parameter $d^*$ is tuned according to the expected user mobility and clustering patterns, with clusters of smaller (larger) diameter benefitting from smaller (larger) values of $d^*$.

Suppose there is an information source residing at static node $s$, and a user $i$ wishes to subscribe to that source's feed. We assume that users know the $O(\log^2 n)$-length HST address of source $s$ (through a one-time flood, for example). The information delivery algorithm proceeds in three steps: subscription of users to receive data from source $s$; data distribution from $s$ to all subscribed users; reconfiguration of delivery routes due to mobility and link dynamics.

The user subscription procedure is shown in Algorithm 1: users send subscription requests directly to source $s$ via their best HSTs. If the source is sufficiently far so that a direct communication link is infeasible, nodes designated as warehouses intercept and aggregate requests from nearby users; each activated warehouse then proceeds to act as an information delivery intermediary between the sources and the users it serves. The selection of the *best* HST between user and source in line 2 follows the least common ancestor comparison of HST signature vectors as described in Section III-A. Recall that the same node $w \in V$ may potentially appear as a candidate warehouse in more than one of the HSTs, so activation in line 7 is necessary on a per-HST basis.

**Algorithm 1** warehouse activation

1: User $i$ connects to a neighboring proxy node ($v$), selecting between multiple neighbors based on their link quality.
2: Node $v$ determines the *best* HST $T \in \mathcal{T}$ between itself and $s$.
3: Node $v$ sends a subscription request for source $s$ along the designated path in $T$.
4: **if** the subscription request reaches $s$ without traveling as high as level $d^*$ in tree $T$ **then**
5: Node $v$ establishes a direct route with source $s$; no warehouse is activated.
6: **else**
7: Node $v$'s ancestor $w$ on level $d^*$ of HST $T$ *intercepts* the subscription request. $w$ becomes an active warehouse (nominated by $v$, for the source $s$, in HST $T$).
8: **end if**

The information distribution procedure is shown in Algorithm 2. Communication between users and warehouses, in lines 3 and 4, is done over the HST paths encoded in the initial HST construction of Section II-B. The HST $T'$ selected by warehouse $w$ in line 1 is independent of HST $T$ on which user requests arrived to it; $T'$ is simply the best HST on which to point-to-point route from $w$ to $s$.

**Algorithm 2** information distribution

1: Warehouse $w$ determines the best HST $T' \in \mathcal{T}$ on which to route between itself and source $s$.
2: On behalf of all users subscribing to $w$ for source $s$, $w$ sends a subscription request to $s$ via the $w - s$ route in $T'$.
3: Source $s$ sends information to all subscribed warehouses, via the respective best HSTs computed in line 1.
4: Warehouse $w$ disseminates the data to all of its subscribed users via the subtree paths rooted at $w$ in $T$.

Finally, as users traverse the network, delivery routes need to be reconfigured dynamically due to the mobility of users and link dynamics (see Algorithm 3). The default mechanism is to restart the user subscription procedure of Algorithm 1. Since users may completely lose connectivity with the network (e.g., due to moving through an area with no network coverage), we use a combination of periodic *keep-alive* messages and aging timers to purge inactive users from the routing tables. Observe that a smaller $d^*$ parameter results in shorter reconfiguration time after node mobility; this parameter should be set as small as is feasible, given the expected mobility and clustering patterns of the users.

**Algorithm 3** routing reconfiguration

1: When user $i$'s link with proxy $v$ breaks, restart Algorithm 1.
2: Periodic *keep-alive* messages are sent between a subscribed proxy node $v$ and its designated warehouse $w$. If no messages from $v$ are received in a predetermined interval, user subscription is dropped; if no messages from $w$ are received in a predetermined interval, the HST $T_w$ by which $v$ subscribed to $w$ is deemed unreliable, and $v$ restarts Algorithm 1 on the reduced set of HSTs $\mathcal{T} \setminus T_w$.
3: A warehouse with no active user subscriptions is deactivated.
4: Periodic *keep-alive* messages are sent between a subscribed warehouse $w$ and a source $s$. If no messages from $w$ are received in a predetermined interval, warehouse subscription is dropped; if no messages from $s$ are received in a predetermined interval, the HST $T_s$ by which $w$ subscribed to $s$ is deemed unreliable, and $w$ restarts Algorithm 2 on the reduced set of HSTs $\mathcal{T} \setminus T_s$.

Observe that user mobility is handled without the pitfalls of the route acquisition delay inherent in on-demand multicast protocols. In particular, routes between nodes are encoded in their $O(\log^2 n)$ addresses and can be determined locally at the sender. The use of $\log n$ HSTs provides route diversity as each pair of nodes has $\log n$ independent routes to choose from. However, significant changes in network topology negatively impact routing stretch and should result in reconstruction of $\log n$ HSTs. Note that user mobility does not change network topology as users connect to the network through proxy nodes.

| Method | Distributed | Stretch | Storage (per node) |
|---|---|---|---|
| Shortest-path tree (ex. PIM-SM, DVMRP) | Yes | 1 | $O(n)$ |
| RP multicast | Yes | $O(n)$ | $O(\#RPs)$ |
| S4 | Yes | 3 | $O(\sqrt{n})$ |
| Steiner tree | No | $O(n)$ | $\Omega(1)$ |
| **HST-based Protocol** | **Yes** | $O(1)$ | $O(\log^2 n)$ |

TABLE I: Properties of information delivery algorithms. Note: bounds are given for networks with constant expansion rate.

### *B. Stretch Analysis*

**Proposition 1** *In a network with constant expansion rate, the user-to-source path defined by the algorithm has constant stretch, with high probability.*

*Proof:* If a user connects directly with a source, as in line 5 of Algorithm 1, then by Corollary 1, the result holds. Suppose on the other hand that a user, using proxy node $v$, connects to source $s$ via warehouse $w$. Let $T$ be the *best* HST between $v$ and $s$ (as computed in line 2 of Algorithm 1), and $T'$ be the *best* HST between $w$ and $s$ (as computed in line 1 of Algorithm 2). Let $d_{\text{opt}}(\cdot,\cdot)$ denote the shortest path distance in the network, and $d_T(\cdot,\cdot)$ and $d_{T'}(\cdot,\cdot)$ denote the distance via the HST $T$ and $T'$ respectively. By Corollary 1, $d_T(v,s) \leq c \cdot d_{\text{opt}}(v,s)$, and $d_{T'}(w,s) \leq c \cdot d_{\text{opt}}(w,s)$, both with high probability, for some constant $c \geq 1$.

Now, since the path between $v$ and $s$ in $T$ passes through warehouse $w$, we have that $d_T(v,w) \leq d_T(v,s) \leq c \cdot d_{\text{opt}}(v,s)$ (w.h.p.) (1). Moreover, we have by the triangle inequality that $d_{\text{opt}}(w,s) \leq d_{\text{opt}}(v,s) + d_T(w,v) \leq (1+c) d_{\text{opt}}(v,s)$ (w.h.p.). Therefore, $d_{T'}(w,s) \leq c(1+c) d_{\text{opt}}(v,s)$ (w.h.p.) (2).

Combining (1) and (2), since the length $l$ of the user-to-source path between $v$ and $s$ computed by our algorithm is given by $l = [d_T(v,w)] + [d_{T'}(w,s)]$, we have that $l \leq [c \cdot d_{\text{opt}}(v,s)] + [c(1+c) d_{\text{opt}}(v,s)] = O(1) d_{\text{opt}}(v,s)$ (w.h.p.). ■

## V. SIMULATION

In Sections II, III and IV, we showed theoretically that our HST-based Protocol achieves three of the four desired properties of an information delivery scheme. Namely, our algorithm is computed in a completely **decentralized** fashion; we guarantee (w.h.p.) a **constant stretch** path between sinks and sources; each node maintains a **routing table of size** $O(\log^2 n)$ in expectation. See Table I for a summary of the properties and comparison to related techniques.

In this section, therefore, our goal is to evaluate the final desired property of our information delivery scheme — **aggregation** — which counts the total number of edges required to join all of the users with all of the sources. For fair evaluation, we compare our algorithm on this metric with shortest-path tree multicast (i.e., PIM-SM [6]) and S4 (compact routing [15]), the two other distributed information delivery schemes that can guarantee a constant stretch source-to-sink path. With only a single information source in the network, the total number of edges in the HST-based connecting subgraph matches or is slightly more than the total number of edges in the connecting subgraph generated by a shortest-path tree or S4 paths. However, the main result of this section is that, as the number of sources increases, the HST method aggregates more efficiently than the other methods. The reason is that the warehouses and HST paths can be **re-used** across multiple sources in our method (a single warehouse can often serve a cluster of users for more than one source); on the other hand, neither shortest-path multicast nor S4 actively reuse paths across multiple sources — the paths to each source are selected essentially independently.

We simulate the connecting subgraphs generated by the HST-based Protocol, the shortest-path tree and S4, on four network topologies: (i) a perturbed grid on 1024 nodes in the unit square; (ii) a random node topology that places 1024 nodes uniformly at random in the unit square; (iii) the previous random node topology but with all the nodes in two disks of radius 0.15 removed (i.e., "small" holes); (iv) the previous random node topology but with all the nodes in two disks of radius 0.2 removed (i.e., "large" holes). In each network, edges were determined by the unit-disk graph model with radius = $1/16$, giving the connected networks an average degree of 10.8, 11.4, 11.0 and 10.7, respectively. Three clusters, each comprising three users, traversed the network with a trajectory generated according to the Reference Point Group Mobility model (RPGM) [10], over 25 discrete time steps[4]. At each time step, users were identified with a proxy node, selected to be the closest fixed node in the network. At each of 100 iterations, we placed 1, 2, 3, 5, 10 or 20 source nodes at random positions in the network.

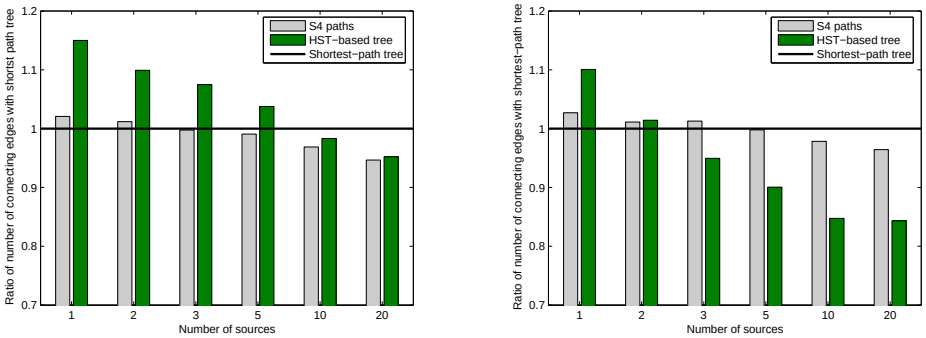
Figure 3 shows the average number of edges in the connecting subgraph (joining each user with each source) for each of the three methods, in the four network topologies. The average is taken over 100 iterations of the 25-step RPGM trajectory (with randomly selected source locations at each iteration). For edges generated by the HST-based Protocol, we select candidate warehouses residing at level $d^* = 4$ of the HSTs[5]. Results for other trajectories are not shown, but give qualitatively similar results.

Observe that, for all topologies considered, with only a single source in the network, the HST-based tree fares worse over the trajectory than shortest-tree multicast and S4. The reason is that, although our method guarantees $O(1)$ stretch w.h.p., shortest-path multicast and S4 guarantee a stretch of 1 and 3, respectively (although S4 generally gives a stretch much closer to 1 in practice)[6]. Our method is geared toward aggregation at warehouse nodes, but when there is only a *single* source in the network, the total size of the connecting subgraph is dominated by the lengths of the source-to-sink

[4]In RPGM, each cluster follows the lead of a reference cluster center; users within a cluster may engage in complex mobility patterns about the center.

[5]HSTs have a height of 11 for networks with 1024 nodes. Selecting candidate warehouses from other levels produced similar aggregation trends as in Figure 3, but $d^* = 4$ performed the best for these topologies.

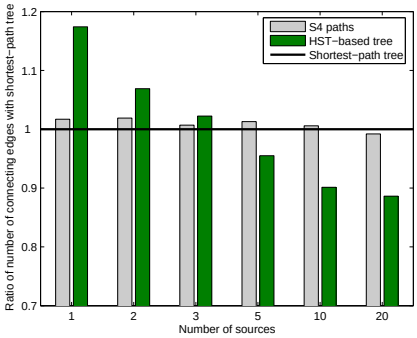
[6]Recall that we achieve $O(1)$ stretch despite requiring only $O(\log^2 n)$ storage per node in expectation, while shortest-path tree multicast and S4 require $\Omega(n)$ and $\Omega(\sqrt{n})$ storage per node, respectively.



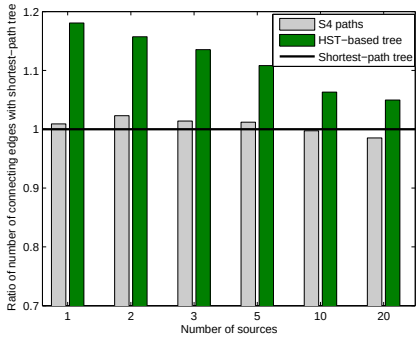

(a) Perturbed Grid




(b) Random Node Topology




(c) Random Node Topology with Small Holes




(d) Random Node Topology with Large Holes

Fig. 3: Average number of edges required to join 9 mobile users with 1, 2, 3, 5, 10 or 20 randomly positioned sources. Three protocols (HST-based Protocol, S4 and shortest-path multicast) are tested for the perturbed grid, the random node topology, and the random node topology with two "small" or two "large" holes. These networks have 1024, 1024, 888 and 770 nodes, respectively. The average number of edges is taken over 100 iterations (with randomly selected source locations) of a 25-step RPGM trajectory. Results are presented as a ratio over the average number of edges generated by the benchmark shortest-path tree multicast method; a reading below 1 indicates a smaller connecting subgraph on average than shortest-path tree multicast. The HST-based tree draws warehouses from level $d^* = 4$.

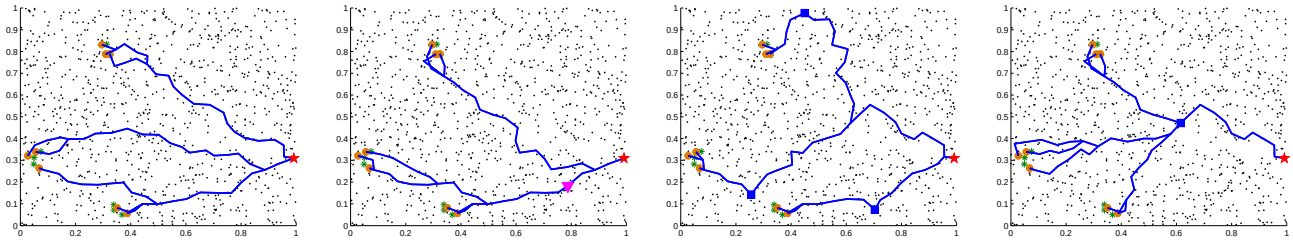
paths rather than any potential path sharing and aggregation.

However, in both Figure 3(a) and 3(b), as the number of sources increases, we observe that our method increasingly outperforms both shortest-path multicast and S4, as the HSTs reuse paths and warehouses more efficiently. With more sources, the contribution of the path length (where our method lags) to the aggregation metric is dampened, while the ability to reuse warehouses and connecting paths (where our method shines) is rewarded. Comparing Figures 3(a) and 3(b), we also observe that our method seems to work better in a setting with high path diversity, outperforming multicast more easily in the random node topology than in the perturbed grid (which comprises a less diverse set of Manhattan-style paths). A high path diversity means that shortest-path routing can reach a given source using a larger selection of independent paths, which results in less aggregation; on the other hand, the HST-based protocol is immune to such path diversity due to the regimented routes dictated by HST edges. Thus, we conclude that our HST infrastructure is particularly well-suited for the multiple source setting; despite our infrastructure requiring only $O(\log^2 n)$ storage per node in expectation (vs. $O(n)$ and $O(\sqrt{n})$ for the other methods), we not only guarantee a constant-stretch path (w.h.p.) between source and sink, but also maintain a smaller connecting subgraph over time than these alternatives.

In Figures 3(c) and 3(d), where we consider the random node topology with two small or two large holes (i.e., nodes removed from two disks of radius 0.15 or 0.2, respectively), observe that as the number of sources increases, the HST-based method improves its aggregation performance relative to shortest-path multicast. This matches our earlier conclusions from the simulations on the perturbed grid and random node topology. However, while the HST-based tree outperforms both multicast and S4 when the topology has "small" holes (beginning at 5 sources), the HST-based tree fails to aggregate more efficiently than either alternative when the topology has "large" holes[7].

The reason for this disparity in performance between Figures 3(c) and 3(d) is similar to the path diversity arguments above. Namely, as the hole radius increases, more and more of the connecting shortest paths will hug the boundary of the holes. When constructing a shortest path, a large hole acts like a magnet, funneling the path into, along and out of the boundary edges of the hole. Whereas previously, a cluster of users may have sent a set of long non-overlapping parallel

[7]The particular radius parameters 0.15 (small) and 0.2 (large) were selected to illustrate the observed phase transition between good and bad relative performance of our method on the topology with holes.



(a) Shortest-path Tree (b) S4 Paths (c) HST-based Tree, $d^* = 3$ (d) HST-based Tree, $d^* = 4$

Fig. 4: Snapshots of the connecting subgraphs for a single source in the random node topology as generated by (a) shortest-path tree, (b) S4 paths, (c) HST paths with warehouses drawn from level $d^* = 3$, and (d) HST paths with $d^* = 4$. The source is denoted by a red star; each user is given by a green asterisk, while the nearest network proxy nodes to the users are given by brown circles; the routing beacon in S4 is given by a purple triangle; the warehouses in Figures 4(c) and 4(d) are given by blue squares.

paths toward the source, now with a large hole those paths can aggregate together much earlier. As a result, methods like shortest-path tree and S4 (which are based on building shortest paths) perform well in terms of the aggregation metric on topologies with large holes. On the other hand, HST-based paths are less inclined to be heavily aggregated at the hole boundary. The only opportunity for previously parallel paths to be aggregated at a hole boundary is in the first cluster-to-warehouse leg of the journey; once at the warehouse, only a single long path is sent to the source, leaving less opportunity for significant aggregation. Thus, while the HST-based method still improves its relative aggregation performance as the number of sources increases, it is not as effective as shortest-path methods on topologies with large holes.

Regarding the overall performance of S4 across all previous simulations, we comment that S4 generally performs on par with shortest-path multicast, both in terms of stretch and the aggregation metric. This is expected, as S4 selects routes along the shortest path toward *beacon* nodes — where a small amount of aggregation occurs — and are then within just a few hops of the destination.

Figure 4 gives snapshots of the connecting subgraphs with 1 randomly positioned source, at a particular instance in the trajectory. Observe that both multicast and S4 build relatively direct routes to the source; on the other hand, the HST-based tree routes via interior nodes of the HST, but actively seeks to join paths at these aggregation nodes.

## VI. Conclusion

We present a robust, distributed, constant-stretch information delivery algorithm, able to support an arbitrary number of sources and mobile users as they travel through a wireless mesh network. Our sparse infrastructure is based on the Hierarchical Well-Separated Tree construction. Though able to be built in a lightweight, distributed and randomized fashion, HSTs produce a comprehensive spatial clustering of the network, providing bounded-stretch routing and aggregation of delivery paths.

Future work includes optimizing information delivery to users when their specific motion trajectories are known or can be predicted. Given such a prediction, we can seek to improve the set of candidate warehouses available to users, and then optimize the sequence of warehouses that users connect to as they traverse the network.

**Acknowledgment:** The authors gratefully acknowledge the support of NSF grant CNS 0626151, ARO grant W911NF-07-2-0027 for the Army High Performance Computing Research Center at Stanford, ARO grant W911NF-10-1-0037, and a Stanford UPS grant.

## References

[1] *Funkfeuer Free Net*.
[2] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996.
[3] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *STOC*, pages 161–168, 1998.
[4] D. S. J. De Couto, D. Aguayo, J. C. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, pages 134–146, 2003.
[5] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Trans. Comput. Syst.*, 8(2):85–110, 1990.
[6] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An architecture for wide-area multicast routing. In *SIGCOMM '94*, pages 126–135, 1994.
[7] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*, pages 448–455, 2003.
[8] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the USENIX NSDI Conf*, 2005.
[9] J. Gao, L. Guibas, N. Milosavljevic, and D. Zhou. Distributed resource management and matching in sensor networks. In *IPSN*, April 2009.
[10] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *MSWiM*, pages 53–60, 1999.
[11] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *STOC*, pages 741–750, 2002.
[12] H. S. Kim, T. Abdelzaher, and W. H. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *SenSys*, pages 193–204, 2003.
[13] R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA*, pages 798–807, 2004.
[14] S. J. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mob. Netw. Appl.*, 7(6):441–453, 2002.
[15] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proc. of the USENIX NSDI Conf*, 2007.
[16] A. Motskin, I. Downes, B. Kusy, O. Gnawali, and L. Guibas. Network warehouses technical report. Technical report, http://www.stanford.edu/~amotskin/infocom11-tech.pdf, Stanford University, 2009.
[17] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *Proceedings of the USENIX NSDI Conf*, 2007.