2006

Energy-Efficient Distributed Consturctions of Minimum Spanning Tree for Wireless Ad-hoc Networks

Maleq Khan

Gopal Pandurangan

V. S. Anil Kumar

Report Number:
06-019

ENERGY-EFFICIENT DISTRIBUTED CONSTRUCTRIONS OF MINIMUM SPANNING TREE FOR WIRELESS AD-HOC NETWORKS

Maleq Khan
Gopal Pandurangan
V.S. Anil Kumar

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #06-019
October 2006

Energy-Efficient Distributed Constructions of Minimum Spanning Tree for Wireless Ad-hoc Networks

Maleq Khan Gopal Pandurangan V.S. Anil Kumar

Abstract—**The Minimum Spanning Tree (MST) problem is one of the most important and commonly occurring primitive in the design and operation of data and communication networks. While there are distributed algorithms for the MST problem these require relatively large number of messages and time, and are fairly involved, require synchronization and a lot of book keeping; this makes these algorithms impractical for emerging technologies such as ad hoc and sensor networks. In such networks, a sensor has very limited power, and any algorithm needs to be simple, local and energy efficient for being practical. Motivated by these considerations, we study the performance of a class of simple and local algorithms called Nearest Neighbor Tree (NNT) algorithms for energy-efficient construction of MSTs in a wireless ad hoc setting. These employ a very simple idea to eliminate the work involved in cycle detection in other MST algorithms: each node chooses a distinct rank, and connects to the closest node of higher rank. We consider two variants of the NNT algorithms, obtained by two ways of choosing the ranks: (i) Random NNT, in which each node chooses a rank randomly, and (ii) Directional NNT, in which each node uses directional information for choosing the rank. We show provable bounds on the performance of these algorithms in instances obtained by uniformly distributed points in the unit square.**

Finally, we perform extensive simulations of our algorithms. We tested our algorithms on both uniformly random distributions of points, and on realistic distributions of points in an urban setting. The cost of the tree found by the NNT algorithms is within a factor of 2 of the MST, but there is more than a ten-fold saving on the energy and about a five fold saving on the number of messages sent. Also, our algorithms are significantly simpler to implement compared to, for instance, the GHS algorithm, which is essentially optimal with regards to the message complexity. Thus, our results demonstrate the first such tradeoff between the quality of approximation and the energy cost for spanning trees on ad hoc networks, and motivates similar considerations for other important problems.

Index Terms— **Minimum Spanning Tree, Optimization, Wireless Ad-hoc Networks, Sensor Networks, Energy-efficient Algorithms, Distributed Algorithms.**

M. Khan and G. Pandurangan are with the Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA. E-mail: {mmkhan, gopal}@cs.purdue.edu.

V.S. Anil Kumar is with the Basic and Applied Simulation Science (CCS-5) and National Infrastructure and Analysis Center (NISAC), Los Alamos National Laboratory, MS M997, P.O. Box 1663, Los Alamos, NM 87545. E-mail: anil@lanl.gov

I. Overview

A. Introduction and Motivation

The Minimum Spanning Tree (MST) problem is one of the most important and commonly occurring primitive in the design and operation of data and communication networks. For instance, in ad hoc sensor networks, MST can be shown to be the optimal routing tree for data-centric routing [1]. Traditionally, the efficiency of distributed algorithms is measured by running time and number of messages exchanged among the computing nodes, and a lot of research has gone into the design of algorithms that are optimal with respect to such criteria. There are distributed algorithms of that find the optimal MST (for e.g., see [2], [3]) and are essentially optimal in terms of time complexity: they run in $O(Diam(G) + n^{\epsilon})$ time, and there are matching lower bounds. However, these algorithms involve a lot of message transfers and time. The GHS algorithm [4] uses $\Theta(n \log n + |E|)$ messages, and is essentially optimal with respect to the message complexity. Despite their theoretical optimality, these algorithms are fairly involved, require synchronization and a lot of book keeping; such algorithms are impractical for ad hoc and sensor networks. For example, consider sensor networks — an ad hoc network formed by large numbers of small, battery-powered, wireless sensors. In many applications, the sensors are typically "sprinkled" liberally in the region of interest and the network is formed in an ad hoc fashion by local self-configuration. Since each sensor usually knows only its (local) neighbors, the network management and communication has to be done in a *local and distributed* fashion. Additionally, because of battery limitations, energy is a very crucial resource. A distributed algorithm which exchanges a large number of messages can consume a relatively large amount of energy (and also

time) and may not be suitable in an energy-constrained ad hoc wireless sensor network.

Thus it is necessary to develop simple, local, distributed algorithms which are energy-efficient and (preferably also time-efficient), even at the cost of being *sub-optimal.* This adds a new dimension to the design of distributed algorithms for such networks. Thus we can potentially *trade-off* optimality of the solution to work done by the algorithm. In a sensor network, the total energy cost ("energy complexity") of a distributed algorithm typically depends on the number of messages exchanged and the energy needed to transmit the messages over a certain distance (cf. Section I-C). (It can also depend on the time complexity of the distributed algorithm). The (radiation) energy needed to transmit a message is typically assumed proportional to some *work function* f (typically square or some small power) of the distance between the sender and the receiver [5], [6]. Thus it becomes important to measure efficiency of a distributed algorithm in terms of power, energy, besides the number of messages.

We study a class of *simple, local, distributed, approximation* algorithms called the *Nearest Neighbor Tree (NNT) algorithms* to build slightly sub-optimal trees, with low energy complexity. A fundamental step in all existing algorithms for the MST is *cycle detection*: given an edge, one needs to determine whether the edge would form a cycle with the edges already chosen. This deceptively simple operation leads to a big overhead: a significant amount of book keeping and message passing needs to be done in order to maintain the components, and answer such queries. Our NNT algorithms bypass such a step completely by a very simple idea: each node chooses a unique *rank*, a quantity from a totally ordered set, and a node connects to the *nearest* node of higher rank. Observe that this immediately precludes cycles, and the only information that needs to be exchanged is the rank; also, this information does not have to be updated continuously over the course of the algorithm.

B. MST and its Applications

Formally, our focus is the following geometric weighted minimum spanning tree problem: given an arbitrary set N of points (nodes) [1] in a plane [2], find a tree T spanning N such that $\sum_{(u,v)\in T} d^{\alpha}(u,v))$ is minimized where $d(u,v)$ is the distance of an edge $(u,v) \in T$ according to some norm (we use the Euclidean norm in this paper) and α is a small positive number. The motivation for this objective function comes from energy requirements in a wireless communication paradigm (see also next Section): to transmit a signal over a distance r, the required *radiation energy* is proportional to r^{α}, where typically α is 2 and can range up to 4 in environments with multiple-path interferences or local noise [6], [5]. It can easily be shown (e.g., using Kruskal's algorithmic construction) that the MST which minimizes $\sum_{(u,v)\in T} d(u,v)$ also minimizes $\sum_{(u,v)\in T} d^{\alpha}(u,v)$ for any $\alpha > 0$.

Two important applications of the MST in wireless networks are broadcasting and data aggregation. The MST is the optimal broadcast tree to minimize radiation energy consumption since it minimizes $\sum_{(u,v)\in T} d^{\alpha}(u,v)$. In data aggregation, the idea is to combine the data coming from different sources enroute to eliminate redundancy and minimize the number of transmissions and thus saving energy; the common aggregate functions are minimum, maximum, average, etc [7]. One popular paradigm for computing aggregates is to construct a (directed) tree rooted at the sink where each node forwards its (locally) *aggregated* data collected from its subtree to its parent [8], [9], [10], [11]. Again, in such cases, MST is the optimal data aggregation tree, since it works exactly as a reverse broadcast tree [1].

C. Energy Model and Work Complexity

To run a distributed algorithm in an ad hoc wireless setting, the following modules of an wireless device are typically involved: a digital unit for processing the signals and performing network protocol functions, and a radio (transceiver) module for communication [12]. Thus one can consider the following three components of energy consumption to run a distributed algorithm.[3] 1) Radiation energy which is proportional to r^{α} to transmit to distance r. 2) A constant (independent of distance) amount of energy e_c required by the radio electronics (transceiver) for each unit of data at the sending and receiving end. Thus energy consumption to transmit a b-bit message to distance r is $be_c + bcr^m$, for some constant c. 3) Energy consumption in digital electronics: even when a node does not receive or transmit a message, the digital electronics and transceiver (in listening or sleeping mode) dissipates power at a constant rate. Let this power is p_c. Thus, if a distributed algorithm exchanges M messages and takes

[1] E.g., these may represent sensors. We assume that these have unique labels or id's.

[2] We consider the 2-dimensional setting for concreteness; our results can be generalized for higher dimensions.

[3] In this paper we do not consider the affects of protocol layers (e.g. the MAC layer), and the overheads resulting from their interaction-thus the focus of this work is to optimize the *static* energy requirements.

t time to complete the algorithm, total energy-cost of the algorithm is

$$E = tp_c + Mbe_c + bc\sum_{i=1}^{M} r_i^{\alpha},$$

considering all messages have same size b and the ith messages travels to distance r_i. Thus time, number of messages, and distance needed to transmit messages all determine the total energy cost.

Motivated by above, in addition to the traditional time and message complexity, we introduce a new complexity term called *work* defined as $w = \sum_{i=1}^{M} r_i^{\alpha}$. Thus total radiation energy is directly proportional to the work done by the algorithm. We show that NNT algorithms perform better in all three: time, number of messages and work. Thus, total energy consumed by the algorithms is less for our algorithms compared to the algorithms that construct (optimal) MSTs.

The quality of a spanning tree T is defined by $Q_\alpha(T) = \sum_{e\in T} |e|^\alpha$; e denotes an edge of T. For $\alpha = 1$, $Q_1(T)$ is simply the sum of the lengths of the edges. Tree with lower $Q_\alpha(T)$ is considered better tree. Our goal is to develop local distributed algorithm with the objective of minimizing total energy E, while keeping the quality of the spanning tree produced to be reasonably close to the optimal.

D. Our Contributions and Results

Our main contribution is detailed theoretical and experimental study of a simple and local class of algorithms, specifically for ad hoc and sensor networks. Our algorithms, called the *Nearest Neighbor Tree (NNT) algorithms* use a very simple idea to avoid cycle formation: each node (independently) chooses a distinct rank, and connects to the closest node of higher rank. Depending on how ranks are chosen we study two types of NNT algorithms: Random-NNT (ranks are chosen randomly) and Directional-NNT (Dir-NNT) (ranks are based on coordinate information). Both are well motivated: when nodes don't know their geometric coordinates[4]. Random-NNT is natural, but if nodes know their coordinate location then Dir-NNT is more suitable.

We theoretically analyze the performance of both these NNT algorithms in the model where n nodes are uniformly distributed in a unit square (this is a popular probabilistic model for ad hoc wireless networks, e.g., see [13]). Our results are enumerated below.

[4]Consider a scenario where sensor nodes are sprinkled randomly in the ocean from a high flying airplane; the nodes typically will not have (accurate) knowledge of their coordinates, unless they have some sort of geographic information locater (e.g., GPS).

- **Quality bounds:** We give asymptotically tight bounds on the cost of the tree found by Random-NNT: for $\alpha \leq 2$, $E[Q_\alpha(RNNT)] = O(n^{1-\alpha/2}\log^{\alpha/2} n)$ and for $\alpha > 2$, $E[Q_\alpha(RNNT)] = O((\log n)^{\alpha/2-1})$, where RNNT is the tree computed by Random-NNT algorithm. Thus, for $\alpha = 1$, this is an $O(\sqrt{\log n})$ approximation and for $\alpha = 2$, this is an $O(\log n)$ approximation. For Dir-NNT, we show that $E[Q_1(DNNT)] = O(\sqrt{n})$ and $E[Q_2(DNNT)] = O(1)$, where DNNT is the tree computed by Dir-NNT algorithm; thus, Dir-NNT is always within a constant factor of the optimal- this shows that at a cost of increased information (i.e., about the coordinates), we can get very good approximations.
- **Message, time, and work complexity:** We show that NNT algorithms has significantly lower message, time, and work complexity compared to other algorithms distributed which compute the optimal MST. We show how NNT algorithms can be implemented efficiently in a wireless setting and show that the work complexities for Dir-NNT and Random-NNT are $O(\log n)$ and $O(1)$ respectively, for $\alpha = 2$. We show that for both NNT algorithms, the message complexity is $O(n)$ and time complexity is $O(\log^2 n)$. We compare the work-complexity of NNT with other distributed algorithms which compute the (optimal) MST. We analyze the work complexity of GHS algorithm [4] — a message-optimal distributed MST algorithm — and show that it is $\Omega(\log^2 n)$, for $\alpha = 2$. Also, the message complexity of GHS is $O(n \log n)$ and the time complexity is $O(n)$. The above bounds hold when the GHS algorithm does not know the coordinate information. When coordinate information of the nodes are known the algorithm can be improved by computing the Yao graph (or the relative neighborhood graph, see e.g., [14]) and then using GHS to find MST from Yao graph. This is essentially optimal with respect to message complexity. To be fair, we compare Dir-NNT, which uses coordinates of the nodes, with GHS algorithm by running on the Yao graph. In a Yao graph, each node has constant degree (at most 6) and thus $|E| = O(n)$. Therefore, message complexity is $O(n \log n)$. We also show that work complexity of GHS to run on a Yao graph is $\Omega(\log n)$, for $\alpha = 2$. In this analysis, the cost of finding the Yao graph is ignored. In fact, the cost for finding the Yao graph is itself is larger than for Dir-NNT. Because, in Dir-NNT, each node needs to find the nearest node on its

right (in the right half plane). Whereas, to compute the Yao graph, each node needs to find the nearest nodes in each of the six cones.

- **Simulation results:** We performed extensive simulations of our algorithms. We tested our algorithms on both uniformly random distributions of points, and on realistic distributions of points in an urban setting obtained from TRANSIMS [15]. Experimental results show that costs for NNT algorithms are significantly smaller than that for an optimal MST algorithm, while the quality NNT is very close to MST. For example, for the TRANSIMS data, we found that the cost of the tree found by the NNT algorithms is within a factor of 2 of the MST, but there is more than a ten-fold saving on the energy and about a five fold saving on the number of messages sent.

E. Organization of the Paper

The rest of the paper is organized as follow. A suboptimal MST called nearest neighbor tree (NNT) is defined and an energy-efficient local distributed NNT algorithm is described in Section II. Theoretical analysis of the quality, and work, messages and time complexity of the algorithms are given in Section III. Simulation results are presented in Section IV.

II. An Energy-efficient Construction

Building minimum spanning tree (MST) in a distributed fashion is highly energy intensive. A distributed algorithm to construct an MST, called GHS algorithm, was proposed in [4]. In the GHS algorithm, initially each node is considered to be a fragment (or a connected component). As the edges are added, the fragments grow by combing smaller fragments. In each "round" of the algorithm, each fragment finds its minimum length outgoing edge (MOE) — which is guaranteed to be in an MST — and uses this edge to combine fragments. Each fragment has two leaders, which are adjacent to the edge added immediately in the previous step. To find the MOE, the leaders send initiate message (relayed by the intermediate nodes) to the members of the fragment. Upon receipt of initiate message, each node tests its adjacent edges by exchanging test/accept/reject messages to check if the node at the other end is in same fragment. Thus, each member node finds its minimum outgoing edge and reports it to the leaders. Upon receipts of reports, the leaders select a new leader — the node which is adjacent to the MOE for the entire fragment and this begins a new round.

Thus a relatively large number of messages needs to be exchanged to find MOEs and to perform the combining operations (changing root of the fragment using "change root" messages); thus, the amount of energy consumed in configuring MST can become prohibitively large. Also as fragments grow, parallelism of the operations reduces (more sequential operations) requiring longer running time. The required number of messages can be shown to be $2|E| + 5n \log n$ and time complexity is $O(n \log n)$, where $|E|$ is the number of edges in the connectivity graph and n is the number of nodes. The time complexity was improved to $O(n)$ in [16], [17], but GHS was shown to be optimal in terms of number of messages.

In this section, we propose a local distributed algorithm to construct a nearly-optimal spanning tree, which requires significantly less energy to build than the MST. The proposed algorithm is very simple. It requires no complex synchronization among the nodes and is naturally robust. An abstract form of the algorithm is given below.

1. Each node u chooses a unique rank $rank(u)$.

2. Node u finds the *nearest* node v such that $rank(u) < rank(v)$ and gets connected to v.

We will shortly describe how to choose such a rank. A distributed implementation in a broadcast setting is given in Figure 1. The following definitions are needed to describe the algorithm and its properties.

Definition 1: Available-for-Connection Set or AC-set. If node u is allowed to get connected to node v, we say v is available to u for connection. $\mathrm{AC}(u)$ is the set of all available nodes. We define $v \in \mathrm{AC}(u)$, if and only if $u \prec v$ for some irreflexive and transitive binary relation $\prec$. Such ordering of the nodes ensures that the connections among nodes do not create any cycle.

Next, we describe how ordering of the nodes can be defined such that each node can determine its relative order with respect to its neighbors locally.

One simple ordering heuristic is as follows. Every node generates a random number independently (between say 0 and 1) and broadcasts this number along with its ID, *identification number*. Each node collects random number-ID pairs of its neighbors and determines its order with respect to the neighbors according to the definition below. Let R_u be the random number generated by node u. We assume that every node is given a unique ID before deployment.

Definition 2: Random Order $\prec_r$. For any two nodes u and v, $u \prec_r v$ if and only if either

a) $R_u < R_v$ or b) $R_u = R_v$ and $\mathrm{ID}(u) < \mathrm{ID}(v)$.

Another ordering heuristic called directional order uses the location information of the nodes. We assume that each node knows its relative coordinates in the plane and no two nodes have the same coordinates (If two nodes have the same coordinates, ID can be used to break ties). Let (x_u, y_u) be the coordinates of u.

Definition 3: Directional Order $\prec_d$. For any two nodes u and v, $u \prec_d v$ if and only if either

a) $x_u < x_v$ or b) $x_u = x_v$ and $y_u < y_v$.

It is easy to see that the graph produced by NNT algorithm is a tree. The relations $\prec_r$ and $\prec_d$ defined above are irreflexive and transitive. For any irreflexive and transitive binary relation $\prec$, if each node u gets connected to exactly one node $v \in \text{AC}(u)$, if $\text{AC}(u) \neq \phi$, there is no cycle in the resulting graph. Further, there is exactly one node u such that $\text{AC}(u) = \phi$ and thus there are $n-1$ edges. Therefore, the resulting graph is a tree.

Definition 4: Nearest Neighbor Tree (NNT). When each node u, if $\text{AC}(u) \neq \phi$, connects itself to a nearest node $v \in \text{AC}(u)$, the resulting tree is called a nearest neighbor tree. When random order is used, the tree is called a *Random-NNT* (RNNT). When directional order is used, the tree is called a *Directional-NNT* (Dir-NNT). The name "nearest neighbor tree" comes from the fact that the tree is formed by connecting each node to the nearest node from the available (for connection) neighbors.

Algorithm 1 Distributed NNT algorithm.

/* The algorithm is executed by each node u independently. Messages are written in the format ⟨message name, sender, [recipient], [other information]⟩. When a message is broadcasted, the recipients are not specified. l is the maximum possible distance between any two nodes. A is the area covered by the nodes and n is the number of nodes[5].*/

$i \leftarrow 1$

Repeat

  Set transmission radius (power level)

    $r_i \leftarrow 2^i\sqrt{\frac{A \log n}{n}}$, for Random-NNT

    $r_i \leftarrow i\sqrt{\frac{A}{n}}$, for Dir-NNT

  If $r_i > l$, $r_i \leftarrow l$

  Broadcast ⟨request, $u, rankinfo$⟩

  /* for Random-NNT rankinfo is random number R_u & ID */

  /* for Dir-NNT rankinfo is coordinate (x_u, y_u) */

  Wait for some specified time period

  $i \leftarrow i + 1$

until (receipt of an "available" message) or ($r_i = l$)

For all v, upon receipt of ⟨request, $v, rankinfo$⟩ do

  if $v \prec u$,

    set transmission radius to $distance(u, v)$

    send ⟨available, u, v⟩ to v

Upon receipt of "available" message(s):

  Select the nearest node v from the senders

  Send ⟨connect, u, v⟩ to v

The algorithm consists of exchanging three types of messages: "request", "available", and "connect" among the nodes. Each node begins with broadcasting a "request" for connection message. Considering a unit square (area $A = 1$), each node broadcasts "request" messages successively to the distances $\frac{1}{\sqrt{n}}, \frac{2}{\sqrt{n}}, \frac{3}{\sqrt{n}}, \ldots$, in case of Dir-NNT and $2\sqrt{\frac{\log n}{n}}, 4\sqrt{\frac{\log n}{n}}, 8\sqrt{\frac{\log n}{n}}, \ldots$, in case of Random-NNT until it finds a node with higher rank. The highest ranked node among all the nodes, can never find a node with higher rank. This node stops transmitting "request" message when it reaches the maximum possible distance between any two nodes. "Request" messages carry rank information (coordinates or random number). The other nodes who can hear the message check their relative rank and send back an "available" message if their rank is higher. The sender of the "request" message selects the nearest node from the senders of "available" messages if more than one available message is received and thus it finds the nearest higher ranked node.

When coordinates are not available (e.g., for Random-NNT), senders can include the transmission power levels in the "available" messages and the recipient can determine the relative distances of the senders from these power levels and the signal-strengths of the received messages. Finally, the node sends a "connect" messages to the nearest higher ranked node, that creates an edge between these two nodes.

Two different strategies are used in increasing the broadcast radius successively for Random-NNT and Dir-NNT. For Dir-NNT, each time radius is increased by $\frac{1}{\sqrt{n}}$. For Random-NNT, radius is doubled each time. Also Dir-NNT begins with a smaller radius compared to Random-NNT. Among many other possible strategies, one strategy can be "begin with a constant radius and increase each time by a constant amount". Selecting the best strategy for an NNT algorithm is not obvious. By experimental trials and theoretical analysis, the strategies given in the Algorithm 1 are found to be optimal for Random-NNT and Dir-NNT. In this paper, we present the results for these strategies only.

III. Analysis

In this section, quality of the trees and energy-cost of the algorithms are analyzed theoretically. In this analysis, we assume that n nodes are uniformly distributed in a unit square. In this setting, we measure the quality of the tree produced by NNT, $Q_\alpha(T) = \sum_{(u,v)\in T} d^\alpha(u,v)$, work $w = \sum_{i=1}^{M} r_i^\alpha$, number of messages, and the time complexities of NNT and GHS algorithms. Although our analysis generalizes to any α, for clarity we consider $\alpha = 1$ and 2.

It was shown by Steele [18] that $Q_2(MST)$ is asymptotically constant, $O(1)$. Also it is well known that $Q_1(MST)$ is $O(\sqrt{n})$. We show that for Dir-NNT, $Q_1 = O(\sqrt{n})$ and $Q_2 = O(1)$ giving an approximation factor of $O(1)$ for both of them. For Random-NNT, $Q_1 = O(\sqrt{n \log n})$ and $Q_2 = O(\log n)$ giving approximation factors of $O(\sqrt{\log n})$ and $O(\log n)$ respectively.

The work complexities for Dir-NNT and Random-NNT are $O(\log n)$ and $O(1)$, where work for GHS algorithm is $\Omega(\log^2 n)$. In [4], authors showed that message and time complexities are $O(n \log n + |E|)$ and $O(n \log n)$ respectively. We show that for both NNT algorithms, number of messages is $O(n)$ and time complexity is $O(\log^2 n)$.

When coordinate information of the nodes are known, efficiency of GHS algorithm can be improved by computing the Yao graph and then use GHS to find MST from the Yao graph. To be fair, we compare Dir-NNT, which uses coordinates of the nodes, with GHS algorithm with Yao graph. In a Yao graph, each node has constant degree (at most 6) and thus $|E| = O(n)$. Therefore, message complexity is $O(n \log n)$. We also show that work complexity of GHS to run on a Yao graph is $\Omega(\log n)$. In our analysis, we actually favor GHS by ignoring the cost incurred for finding the Yao graph. In fact, the cost for finding the Yao graph is itself is larger than that for Dir-NNT. Because, in Dir-NNT, each node needs to find the nearest node on its right (in the right half plane). Whereas, to compute the Yao graph, each node need to find the nearest nodes in each of the six cones.

The following lemmas and theorems prove the above claims.

A. Random-NNT

Lemma 1: In Random-NNT, a node v connects to the ith nearest neighbor with probability $\frac{1}{i(i+1)}$. Thus, a node gets connected within its k nearest neighbors with probability $1 - \frac{1}{k+1}$.

Proof: Let a and x_i be the random number generated by node v and its ith nearest neighbor. Let $X_i = \{x_k | 1 \leq k \leq i\}$. We define, $a > X_i \iff \forall_{1 \leq k \leq i}(a > x_k)$. Since the random numbers are generated by the nodes independently, $\Pr\{a > X_i\}$ = probability that a is the largest among $i+1$ independent identically distributed random numbers $= \frac{1}{i+1}$. Now, the probability that a node connect to the ith nearest neighbor is

$\Pr\{a > X_{i-1}, a < x_i\}$
$= \Pr\{a > X_{i-1}\} \Pr\{a < x_i | a > X_{i-1}\}$
$= \Pr\{a > X_{i-1}\}[1 - \Pr\{a > x_i | a > X_{i-1}\}]$
$= \Pr\{a > X_{i-1}\} - \Pr\{a > x_i, a > X_{i-1}\}$
$= \Pr\{a > X_{i-1}\} - \Pr\{a > X_i\} = \frac{1}{i} - \frac{1}{i+1} = \frac{1}{i(i+1)}$.

Now Probability that a node is connected within k nearest neighbors is $\sum_{i=1}^{k} \frac{1}{i(i+1)} = 1 - \frac{1}{k+1}$. ■

Theorem 1: Expected work complexity of Random-NNT algorithm is $O(n^{1-\alpha/2} \log^{\alpha/2} n)$ for $\alpha \leq 2$ and $O((\log n)^{\alpha/2-1})$ for $\alpha > 2$.

Proof: Consider an arbitrary node u. First transmission radius for "request" message is $r_1 = 2\sqrt{\frac{\log n}{n}}$ and for the ith transmission, $r_i = 2r_{i-1} = 2^i \sqrt{\frac{\log n}{n}}$. Let m be the maximum number of transmissions. $r_{m-1} < \sqrt{2} \leq r_m$, i.e., $m < \frac{\log n}{2 \log 2} + 1$.

Let C_i be the set of nodes in the circle centered at u with radius r_i and $R_i = C_i - C_{i-1}$. $E[|C_{i-1}|] = \pi r_{i-1}^2 n = \pi 2^{2i-2} \log n$. Using Chernoff bound for lower tail, with high probability $|C_{i-1}| \geq \frac{1}{2} E[|C_{i-1}|] = \pi 2^{2i-3} \log n$.

Now, the probability that node u needs the ith transmission = the probability that u is the highest ranked node in $C_{i-1} = \frac{1}{C_{i-1}} \leq \frac{8}{\pi 2^{2i} \log n}$.

$E[|C_i|] = \pi 2^{2i} \log n$. Again using Chernoff bound for upper tail, with high probability $|R_i| \leq |C_i| \leq \pi 2^{2i+1} \log n$. Consider an arbitrary node $v \in R_i$. $\Pr\{\text{rank}(u) < \text{rank}(v) | u \text{ has the highest rank in} C_{i-1}\} = \frac{1}{|C_{i-1}|+1} \leq \frac{8}{\pi 2^{2i} \log n}$. Thus expected number of "available" messages (reply of "request" message) $\leq \pi 2^{2i+1} \log n \frac{8}{\pi 2^{2i} \log n} = 16$. Counting the "request" and the final "connect" messages there are at most 18 messages that travels the distance $\geq r_{i-1}$ and $\leq r_i$.

Thus the expected total work for n nodes (using linearity of expectation) is

$$E[W] \leq n \left(18 r_1^\alpha + \sum_{i=2}^{m} \frac{8}{\pi 2^{2i} \log n} 18 r_i^\alpha \right) \quad (1)$$

$$= \frac{(\log n)^{\frac{\alpha}{2}}}{n^{\frac{\alpha}{2}-1}} \left(2^\alpha 18 + \frac{144}{\pi \log n} \sum_{i=2}^{m} 2^{i(\alpha-2)} \right).$$

For $\alpha = 2$, $E[W] \leq 106 \log n = O(\log n)$. For $\alpha \neq 2$,

$$E[W] \leq 2^\alpha 18 \frac{(\log n)^{\frac{\alpha}{2}}}{n^{\frac{\alpha}{2}-1}} + \frac{2^{2\alpha} 48}{2^\alpha - 4} (\log n)^{\alpha/2-1}.$$

For $\alpha < 2$, the second term becomes negative, thus $E[W] = O(n^{1-\alpha/2} \log^{\alpha/2} n)$.

For $\alpha > 2$, the second term is dominating, thus $E[W] = O((\log n)^{\alpha/2-1})$. ■

Theorem 2: In Random-NNT, $E[Q_\alpha(RNNT)] = O(n^{1-\alpha/2} \log^{\alpha/2} n)$ for $\alpha \leq 2$ and $O((\log n)^{\alpha/2-1})$ for $\alpha > 2$.

Proof: Similar to the proof of Theorem 1, consider an arbitrary node u, and concentric circles centered at u with radius $r_i = 2^i \sqrt{\frac{\log n}{n}}$ for $i = 1, 2, \ldots, m$. Maximum number of circles $m < \frac{\log n}{2 \log 2} + 1$. Let C_i be the set of nodes in the circle with radius r_i and $R_i = C_i - C_{i-1}$. If

u connects to a node v in the ith ring, i.e., $v \in R_i$, (to get an upper bound) we consider distance $d(u,v) = r_i$.

With high probability (by Chernoff bound), $|C_i| \geq \frac{1}{2}E[|C_i|] = \pi 2^{2i-1} \log n$.

Now $\Pr\{u \text{ connects to any } v \in R_i\} = \Pr\{u \text{ connects to a node in } C_i\} \times \Pr\{u \text{ does not connect to a node in } C_{i-1}\} = \left(1 - \frac{1}{|C_i|}\right) \frac{1}{|C_{i-1}|}$ by Lemma 1. Thus,

$$E[d(u,v)] = \left(1 - \frac{1}{|C_1|}\right) r_1^\alpha + \sum_{i=2}^{m} \frac{1}{|C_{i-1}|} \left(1 - \frac{1}{|C_i|}\right) r_i^\alpha.$$

Keeping the dominating terms and using linearity of expectation, for n nodes,

$$E[Q_\alpha] = nE[d(u,v)] \leq nr_1^\alpha + n\sum_{i=2}^{m} \frac{1}{|C_{i-1}|} r_i^\alpha$$
$$\leq \frac{(\log n)^{\frac{\alpha}{2}}}{n^{\frac{\alpha}{2}-1}} \left(2^\alpha + \frac{8}{\pi \log n} \sum_{i=2}^{m} 2^{i(\alpha-2)}\right).$$

For $\alpha = 2$, $E[Q_\alpha] \leq 6 \log n = O(\log n)$. For $\alpha \neq 2$,

$$E[Q_\alpha] \leq 2^\alpha \frac{(\log n)^{\frac{\alpha}{2}}}{n^{\frac{\alpha}{2}-1}} + \frac{2^{2\alpha}3}{2^\alpha - 4} (\log n)^{\alpha/2-1}.$$

For $\alpha < 2$, $E[Q_\alpha] = O(n^{1-\alpha/2} \log^{\alpha/2} n)$ and for $\alpha > 2$, $E[Q_\alpha] = O((\log n)^{\alpha/2-1})$. ■

Theorem 3: Expected message complexity of Random-NNT algorithm is $O(n)$.

Proof: If we consider work needed for every message is 1, i.e., when $\alpha = 0$, the total work is simply the number of messages, M, exchanged in the algorithm. Thus from Equation 1, by putting $r_1^\alpha = r_i^\alpha = 1$ in the right hand side, we get

$$E[M] \leq n\left(18 + \sum_{i=2}^{m} \frac{8}{\pi 2^{2i} \log n} 18\right)$$
$$\leq n\left(18 + \frac{4}{\log n} - \frac{1}{12n}\right) = O(n).$$

■

Theorem 4: Running time of Random-NNT algorithm is $O(\log^2 n)$.

Proof: We assume that transmission of each message take one unit of time and while one node is transmitting a message, no other node in its transmission radius (transmission range) is allowed to transmit.

The radius of the first transmission by each node is $r_1 = 2\sqrt{\frac{\log n}{n}}$. Following the proof of Theorem 1, expected number of nodes within this radius, $E[|C_1|] = 4\pi \log n$ and with high probability, $|C_1| \leq 8\pi \log n$. In the first transmission phase, a node needs to reply to at most $8\pi \log n$ "available" messages. Thus total time for $8\pi \log n$ nodes to complete the first phase is at most $(8\pi \log n)^2 = O(\log^2 n)$.

Now consider an ith transmission phase to distance $r_i = 2^i\sqrt{\frac{\log n}{n}}$. After the $(i-1)$th phase, there can be at most one unconnected node in any circle of radius r_{i-1}, because, otherwise, one node has lower rank than the other and can connect to the that node. Thus expected number of unconnected node within radius r_i is 4 and this number can be at most 24, because distance of any two such nodes is at least r_{i-1}. From the proof of Theorem 1, expected number of reply ("available") messages can be received by one of these nodes is 16. Thus each subsequent transmission phase (other than the first phase) needs constant time. There are at most $\frac{\log n}{2 \log 2} + 1$ phases. Thus total time for Random-NNT algorithm is $O(log^2 n) + O(\log n) = O(\log^2 n)$. ■

B. Dir-NNT

Theorem 5: The expected quality of Dir-NNT, for $\alpha = 1, 2$, and 3, are $O(\sqrt{n})$, $O(1)$, and $O\left(\frac{1}{\sqrt{n}}\right)$ respectively.

Proof: We will upper bound the expected distance that a node needs to connect to some other node. For the purpose of analysis, let us subdivide the unit square into $\sqrt{n} \times \sqrt{n}$ small squares. Length of a side of each small square is $l = \frac{1}{\sqrt{n}}$.

Assume that each node selects the nearest node from a cell which is directly above of it and in the same column or in a column at the right. The probability that a particular cell has at least one node is $p = 1 - \left(1 - \frac{1}{n}\right)^n \geq 1 - \frac{1}{e}$. We further rearrange all the n cells, along with the nodes in it, in a single row - put the cells of first column (bottom cell in the left most position, top cell in the right most position), then the cells of the second column, and so on. In this new arrangement, we are moving the nodes further away and increasing the distances among the nodes; and thus increasing the length of the edges comparing to the original Dir-NNT. As a result, the expected sum of the squared edges in the original Dir-NNT is less than that of Dir-NNT in this new arrangement. All nodes to the right of any node have higher ranks than its own. A node connects to the ith next cell, if there is no node in the next $i-1$ cells and there is a node in the ith next cell. The probability of this event is $p(1-p)^{i-1}$. The distance to the ith next cell is $il = \frac{i}{\sqrt{n}}$. Therefore, if a node u connects to some other v node and $d(u,v)$ is the distance between u and v,

$$E[d^\alpha(u,v)] \leq \sum_{i=1}^{n-1} (il)^\alpha p(1-p)^{i-1}.$$

Using linearity of expectation for n nodes, $E[Q_\alpha(T)] \leq nE[d^\alpha(u,v)]$. As $n \to \infty$, for $\alpha = 1, 2, 3$,

$$E[Q_1(T)] \leq \frac{\sqrt{n}e}{(e-1)} = O(\sqrt{n}),$$
$$E[Q_2(T)] \leq \frac{e(e+1)}{(e-1)^2} = O(1),$$
$$E[Q_3(T)] \leq \frac{e(e^2+4e+1)}{(e-1)^3\sqrt{n}} = O\left(\frac{1}{\sqrt{n}}\right).$$

■

Theorem 6: The energy cost of Dir-NNT algorithm, for $\alpha = 1, 2$, and 3, are $O(\sqrt{n})$, $O(1)$, and $O\left(\frac{1}{\sqrt{n}}\right)$ respectively.

Proof: A node sends its rank info to distance $\frac{1}{\sqrt{n}}, \frac{2}{\sqrt{n}}, \frac{3}{\sqrt{n}}, \ldots$ until it gets a reply back from a higher ranked node. Again we subdivide the area into cells and consider the rearrangement of the cells in a single row as described in Theorem 5. A node sends a message with rank info to the ith next cell, which is at the distance $il = \frac{i}{\sqrt{n}}$, if there is no node in the next $i-1$ cells. The probability of occurring this event is $(1-p)^{i-1}$, where the probability that there is a node in a particular cell is $p \geq 1 - \frac{1}{e}$. The expected number of nodes in a cell is 1. Thus expected number of "accept" messages in response is one and finally at most one connect message sent by this node. Thus, as $n \to \infty$, expected energy cost for one node $\leq \sum_{i=1}^{n-1} 3(il)^{\alpha}(1-p)^{i-1}$. Using linearity of expectation, total expected energy cost,

$$E[C_{\alpha}] \leq n \sum_{i=1}^{n-1} 3(il)^{\alpha}(1-p)^{i-1}. \qquad (2)$$

As $n \to \infty$, for $\alpha = 1, 2, 3$,

$$E[C_1] \leq \frac{3\sqrt{n}e^2}{(e-1)^2} = O(\sqrt{n}),$$
$$E[C_2] \leq \frac{3e^2(e+1)}{(e-1)^3} = O(1),$$
$$E[C_3] \leq \frac{3e^2(e^2+4e+1)}{(e-1)^4\sqrt{n}} = O\left(\frac{1}{\sqrt{n}}\right).$$

■

Theorem 7: Expected message complexity of Dir-NNT algorithm is $O(n)$.

Proof: Again, similar to Random-NNT algorithm, if we consider work for every message is 1, i.e., when $\alpha = 0$, total work is equal to the number of messages M. Thus from Equation 2, by putting $(il)^{\alpha} = 1$ in the right hand side, we get

$$E[M] \leq n \sum_{i=1}^{n-1} 3(1-p)^{i-1} \leq \frac{en}{e-1} = O(n).$$

■

Theorem 8: Running time of distributed Dir-NNT algorithm is $O(\log^2 n)$.

Proof: Stochastically, each node connects to a shorter distance in Dir-NNT than Random-NNT thus requiring transmissions to shorter distances. This allows more simultaneous communications in Dir-NNT than Random-NNT. Moreover, total number of messages for Dir-NNT algorithm is no more than that for Random-NNT. Thus running time for Dir-NNT algorithm $\leq$ running time for Random-NNT = $O(log^2 n)$. ■

C. GHS Algorithm

The authors of GHS algorithm [4] shown the message and time complexity of the algorithm as we discussed earlier in this section. Here we compute the lower bound for work complexity of GHS algorithm.

Lemma 2: Let r_i be the distance of ith nearest neighbor for an arbitrary node. Then $E[r_i] = \frac{ci}{n} = \Theta\left(\frac{i}{n}\right)$, for some constant c and $\frac{1}{\pi} \leq c \leq 2$.

Proof: See appendix. ■

Theorem 9: The expected work complexity of GHS algorithm is $\Omega(\log^2 n)$.

Proof: We analyze work complexity for test/accept/reject messages only. By the end of the algorithm, each node tests all of its adjacent edges by using test/accept/reject messages through these edges one by one. To have a connected graph with high probability the required number of neighbors is $c \log n$ [13], for some constant c. Thus each node send test/accept/reject messages to these $c \log n$ neighbors. We know expected squared distance to the ith nearest neighbor is $E[r_i^2] \geq \frac{i}{n\pi}$ (Lemma 2). Thus expected work by a node is $\geq \sum_{i=1}^{c \log n} \frac{i}{n\pi} = \Omega(\frac{\log^2 n}{n})$. For n nodes, by linearity of expectation, total work $w = n \times \Omega(\frac{\log^2 n}{n}) = \Omega(\log^2 n)$. ■

Theorem 10: The expected work complexity of GHS algorithm to run on Yao graph is $\Omega(\log n)$.

Proof: To find this lower bound, we ignore the energy required to compute the Yao graph. In a Yao graph, each node has most 6 neighbors, thus work cost of the test/accept/rejet messages can be as low as $O(1)$.

Let us consider initiate and report messages. In each level, the leaders of the fragments send the initiate message to all other nodes in the fragment and the member nodes return report messages with the information of minimum outgoing edges (MOE) to the leader. Total number of such messages is $\Theta(n \log n)$, where $\Theta(n)$ messages in each phase (each node need to send/forward these messages) and there are $\log n$ phases. These messages travels through the edges of MST. We know that some of the squared lengths of the $(n-1)$ MST edges is a constant, $O(1)$. Total work complexity for $\Theta(n \log n)$ messages is $\Theta(\log n)$. Since we ignored cost for some messages, this is a lower bound.

Although the initiate message can be broadcasted to the fragment members, the report messages must be transmitted in point-to point basis; because each node sends report to its parent and the parent aggregates data (find min of the MOEs) and forward to its own parent. Therefore, the work is still $\Omega(\log n)$.

However, we can show that even if the initiate message is broadcasted, total work for these messages is also $\Theta(\log n)$. At the ith level, average number of fragments is

$\frac{n}{2^i}$, therefore, there are $\frac{n}{2^i}$ broadcast transmissions by the $\frac{n}{2^i}$ leaders. Again, there are 2^i nodes in each fragments. Thus, in the ith level, the leader need to transmit to at least to the distance of 2^ith nearest neighbor. By using Lemma 2, for $\log n$ phases, work $\geq \sum_{i=1}^{\log n} \frac{n}{2^i} \frac{2^i}{n\pi} = \frac{1}{\pi} \log n$. ■

IV. Simulation Results

We perform extensive simulations of our algorithms to understand their empirical performance. Our experimental setup is the following:

1) **Number of Nodes:** Varying from 50 to 5000.
2) **Node distributions:** Uniformly random distributions in the unit square and several realistic distributions of points in an urban setting obtained from TRANSIMS [15].
3) **Number of Runs:** 50
4) **Measures:** We compare the NNT trees and the MST, with respect to the quality $Q_\alpha(T) = \sum_{(u,v)\in T} d^\alpha(u,v)$ for $\alpha = 1$ and 2. We compare the performance of the NNT algorithms and GHS, with and without the Yao graph information, with respect to the following measures: (i) Number of messages, and (ii) Work, $w = \sum_{i=1}^{M} r_i^\alpha$ for $\alpha = 2$.

In our simulations, we ignore the effects of the MAC layer. Our main results are enumerated below, and validate our theoretical results in earlier sections.

1) The Dir-NNT algorithm always outperforms the Random-NNT algorithm, with respect to the quality, number of messages and the energy.
2) Both Directional and Random-NNT give a very good approximation to the MST- in particular, Dir-NNT is always within about 10% of the MST.
3) For $\alpha = 2, 3$, Random-NNT does not give a very good approximation, but Dir-NNT remains within a factor of 2.
4) The number of messages and the work done by both Directional and Random-NNT is very close, and significantly smaller than that by GHS or GHS with the Yao graph.

A. Quality of the Spanning Trees

We present the simulation results of the quality $Q_\alpha(T)$ for α = 1 and 2. As Figure 1 shows, both Random-NNT and Dir-NNT compare very well with the MST. As shown earlier, the MST cost is $\Theta(\sqrt{n})$ for $\alpha = 1$, and Dir-NNT seems to be within a small constant factor of this value; Figure 2 demonstrates this by showing the values as a fraction of $\sqrt{n}$, and the plot for Dir-NNT is a straight line. For Random-NNT, the asymptotic approximation ratio is $\Theta(\sqrt{\log n})$, and this plot is almost straight.

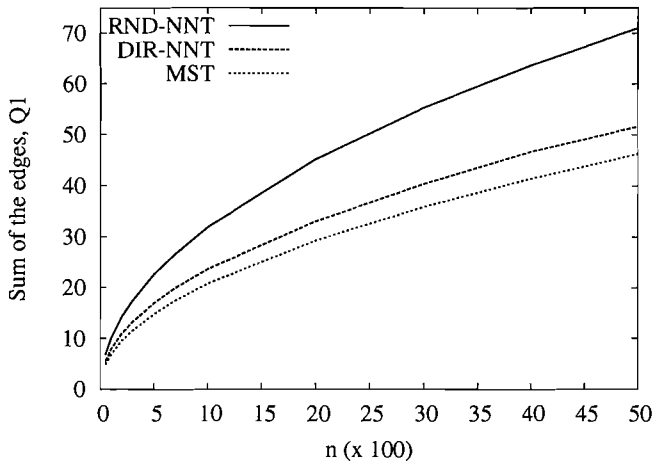

Fig. 1. Sum of the lengths of the edges, $Q_1(T)$, for MST, Random-NNT, and Dir-NNT.

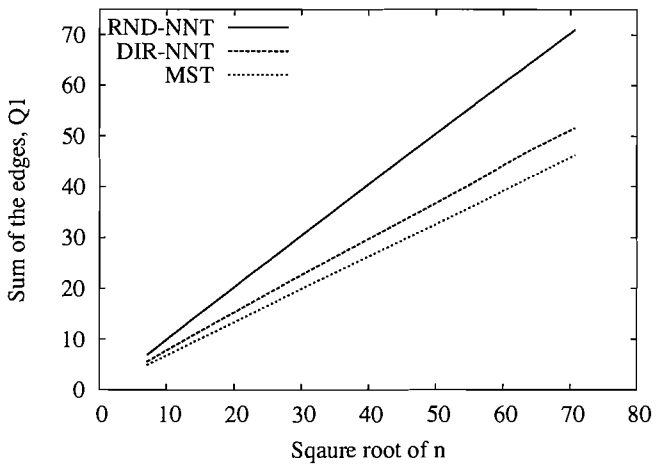

Fig. 2. Sum of the lengths of the edges, $Q_1(T)$, plotted with *sqrtn* for MST, Random-NNT, and Dir-NNT.

Figure 3 shows $Q_2(T)$, the sum of squares of the edge lengths, for the NNT algorithms and the optimum. Both the optimum and Dir-NNT are a constant, and within a factor of 2. However, the value of $Q_2(T)$ for Random-NNT increases with n, as the asymptotic bound is $\Theta(\log n)$- this becomes clear from Figure 4.

B. Energy-cost to Construct the Spanning Trees

In this section, we compare work w for $\alpha = 2$ and number of messages needed by the algorithms. The NNT algorithms are compared with GHS, both with and without the Yao graph. The input to the GHS algorithm must be a connected a graph to obtain MST. To have a connected graph, with high probability, in a wireless ah-hoc

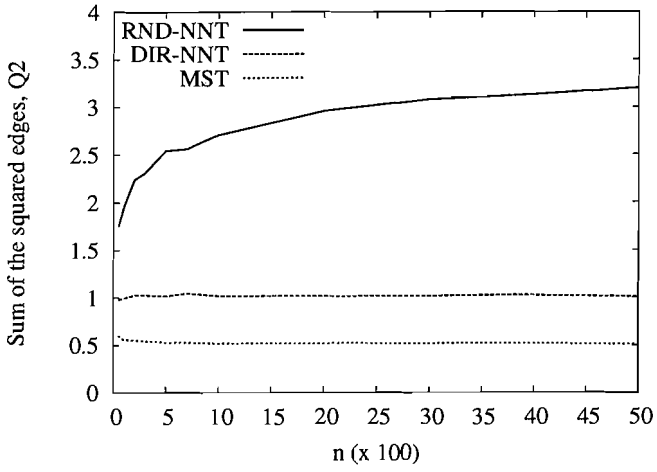

Fig. 3. Sum of the squares of the edge lengths, $Q_2(T)$ for MST, Random-NNT, and Dir-NNT.

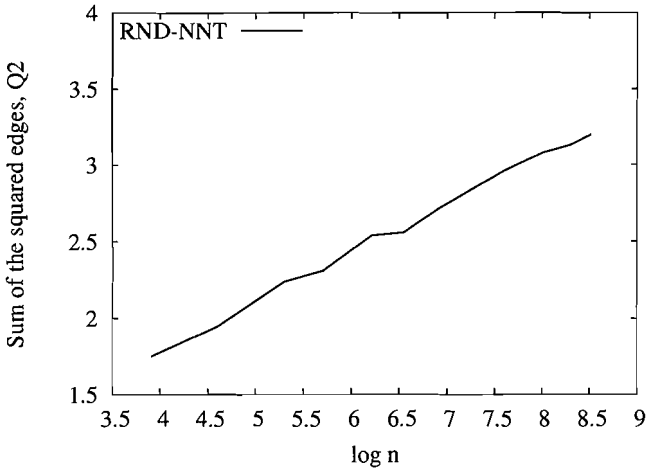

Fig. 4. Sum of the squares of the edge lengths, $Q_2(T)$ for Random-NNT with respect to $\log n$.

network, when the nodes are uniformly distributed, each node must be connected to the nodes which are within distance $\Theta(\sqrt{\frac{\log n}{n}})$ [13]. We consider the radius of the neighborhood to be $1.6\sqrt{\frac{\log n}{n}}$, the minimum required for connectivity. Since each node sends at least one message to each of its neighbor (test message - to check if the neighbor is in the same fragment), cost of GHS algorithm increases as the number of neighbors of the nodes increases.

To determine the neighbors, each node can broadcast a message to distance $1.6\sqrt{\frac{\log n}{n}}$ and consider another node as a neighbor if the node can hear the message from the other node. However, we did not incur any cost on GHS algorithm to find the neighbors (thus favoring GHS). We assumed that each node knows its neighbors and their distances.

In addition, we also simulate GHS on the Yao graph. Each node finds its Yao neighbors first, then executes GHS algorithm.

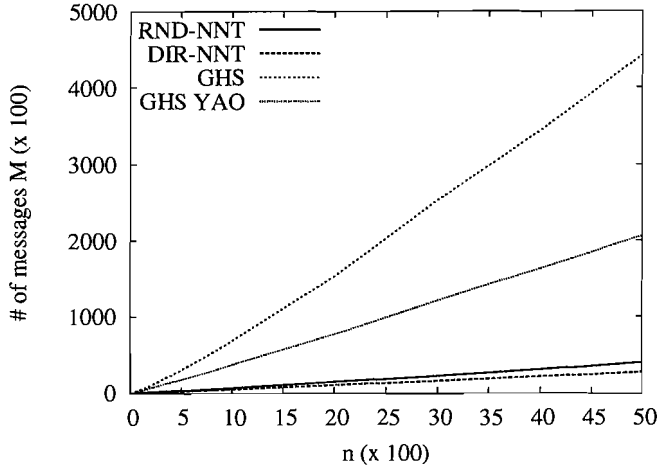

Fig. 5. Number of messages needed to construct the spanning trees.

Fig. 5 depicts the number of messages needed to construct the tree. We see that the number of messages for NNT algorithms is significantly smaller than GHS. Moreover, the number of messages for NNT algorithms increases linearly. On the other hand, the number of messages for GHS increases at a slightly higher rate. In fact, message complexity for GHS is $O(n \log n)$.

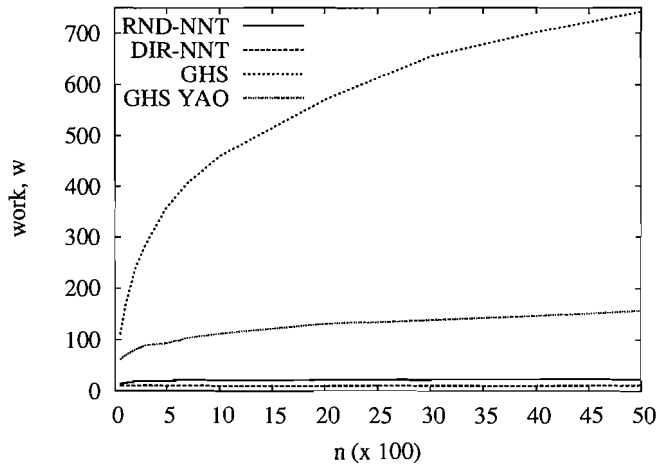

Fig. 6. Work done by the algorithms.

Required work for NNT algorithms is also significantly less than that of GHS algorithm (Fig. 6). In addition, with the number of nodes, energy for NNT algorithms increases in a lower rate than GHS. In terms of both number of messages and work, GHS with Yao graph is more efficient than GHS without Yao graph. However, cost is still much larger than NNT algorithms.

Analytically, we know that the work complexity for Dir-NNT, Random-NNT, and GHS and GHS-YAO are

$O(1)$, $O(\log n)$, $\Omega(\log^2 n)$, and $\Omega(\log n)$ respectively. We can also observe these results from experimental data. Let work $w = c\log^a n$. Then $\log w = \log c + a \log\log n$. Thus if we plot $\log w$ vs. $\log\log n$ the graph is an straight line and the slope of the line is a, the power of log. In Fig. 7, the slope for GHS is greater than 2. For GHS-YAO, the slope is about 1 and for Random-NNT, it is less than 1. For Dir-NNT the slope is 0 which indicates work is $O(1)$.

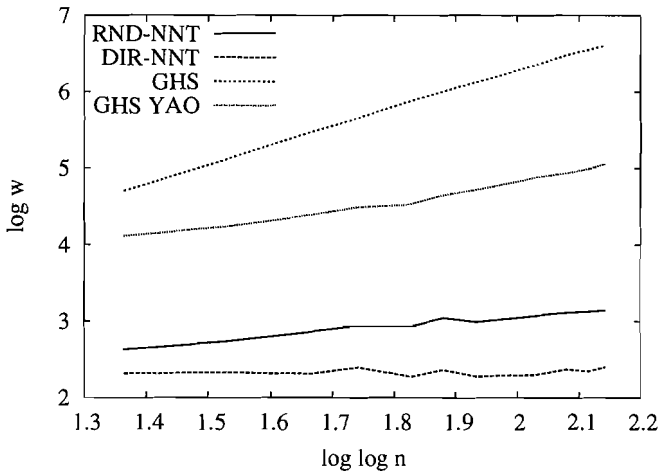

Fig. 7. Slope of the lines indicate the powers of log in work complexity.

C. Experiments on Real Data

We consider a distribution of points in a section of downtown Portland, OR, measuring $2900m \times 2950m$ approximately 9 square KM. The distribution of points, corresponding to cars on the roadway, was obtained from the TRANSIMS simulation [15], which does a very detailed modelling of urban traffic, combining a variety of data sources, ranging from census data to activity surveys to land use data. We use three snapshots, at one minute intervals. The number of nodes (or cars) in these snapshots is different, because there are cars moving in and out of this section all the time. The distribution of nodes at one of the snapshots is shown in Fig. 8. Experimental results on these three snapshots are given in Table I, II, and III. Where the original data was in meters, we converted into KM. Work is computed for $\alpha = 2$.

We see that work and number of messages are significantly larger for GHS algorithm. Work is about 10 times larger and number messages is about 5 times larger than NNT algorithms. On the other hand, both quality for Dir-NNT is within 2-approximation. Although approximation for Q_2 in Random-NNT is large, for Q_1, Random-NNT also provides a close approximation. In this experiment, we only considered the YAO graph assuming that

Fig. 8. The distribution of nodes at one of the snapshots.

TABLE I
EXPERIMENT RESULTS FOR SNAPSHOT 1

Algorithm	Q_1	Q_2	Work	Messages
Dir-NNT	38.72	6.77	90.54	4832
Rnd-NNT	50.75	14.13	131.42	5241
GHS-YAO	33.16	3.73	1271.11	20592

TABLE II
EXPERIMENT RESULTS FOR SNAPSHOT 2

Algorithm	Q_1	Q_2	Work	Messages
Dir-NNT	39.39	8.18	92.28	4647
Rnd-NNT	52.97	20.12	137.91	5250
GHS-YAO	33.52	3.82	1083.99	20417

TABLE III
EXPERIMENT RESULTS FOR SNAPSHOT 3

Algorithm	Q_1	Q_2	Work	Messages
Dir-NNT	38.32	6.25	83.42	4668
Rnd-NNT	52.57	18.47	148.88	5229
GHS-YAO	33.27	3.78	1083.99	20417

the nodes know their coordinates. If the coordinates are not available, for GHS algorithm input need to be a complete graph (each node is a neighbor of the others) to make sure connectivity since the points does not follow any particular (say uniform) distribution. Thus GHS would incur large work and messages. In that case, Random-NNT will still be a good choice over GHS, by sacrificing quality.

REFERENCES

[1] M. Khan, G. Pandurangan, and B Bhargava, "Energy-efficient routing schemes for sensor networks," Tech. Rep. CSD-TR 03-0013, Department of Computer Science, Purdue University, July 2003.

[2] M. Elkin, "Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem," in *Proceedings of Symposium on Theory of Computing (STOC)*, June 2004.

[3] D. Peleg, *Distributed Computing: A Locality Sensitive Approach*, SIAM, 2000.

[4] R. Gallager, P. Humblet, and P. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66–77, January 1983.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[6] K. Delin and S. Jackson, "Sensor web for in situ exploration of gaseous biosignatures," in *Proceedings of IEEE Aerospace Conference*, March 2000.

[7] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *International Workshop on Distributed Event-Based Systems*, July 2002.

[8] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[9] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and G. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *18th ACM Symposium on Operating Systems Principles*, October 2001.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion:a scalable and robust communication paradigm for sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom)*, August 2000.

[11] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*, July 2002.

[12] W. Heinzelman, "Application-specific protocol architectures for wireless networks," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, June 2000.

[13] F. Xue and P. Kumar, "The number of neighbors needed for connectivity of wireless networks," *Wireless Networks*, vol. 10, no. 2, pp. 169–181, March 2004.

[14] D. Eppstein, "Spanning trees and spanners," Tech. Rep. 96-16, Computer Science Department, UC Irvine, May 1996.

[15] "transims.tsasa.lanl.gov," .

[16] B. Awerbuch, "Optimal distributed algorithms for minimum-weight spanning tree, counting, leader election, and related problems," in *Proceedings of 19th ACM Symposium on Theory of Computing*, 1987, pp. 230–240.

[17] J. Garay, S. Kutten, and D. Peleg, "A sublinear time distributed algorithm for minimum-weight spanning trees," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 302–316, February 1998.

[18] J. Steele, "Asymtotics for euclidian minimal spanning trees on random points," *Probability Theory and Related Fields*, vol. 92, pp. 247–258, 1992.

[19] R. Graham, D. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science, Second Edition*, Addison-Wesley Publishing Company, Inc., 1989.

APPENDIX

Proof of Lemma 2

We will show that $\frac{i}{n\pi} \leq E[r_i^2] \leq \frac{2i}{n}$. To get the lower bound, we assume that a node u exists at the center of the square (otherwise we can do a translation and the following argument still holds). It is easy to see that the distance to the ith nearest neighbor of u is stochastically less than or equal to that of any other node (e.g., compared to a node at the boundary). Consider a circle centered at u with radius R such that $\pi R^2 = 1$. The probability that some other node is within distance r from node u is $\frac{\pi r^2}{\pi R^2} = \frac{r^2}{R^2}$. Then the probability that there are at least i nodes within distance r,

$C_i(r) = 1 - \sum_{k=0}^{i-1} \binom{n-1}{k} \left(\frac{r^2}{R^2}\right)^k \left(1 - \frac{r^2}{R^2}\right)^{n-k-1}$.

The probability density function, $P_i(r) = \frac{d}{dr} C_i(r)$

$= -\sum_{k=0}^{i-1} \binom{n-1}{k} k \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^{k-1} \left(1 - \frac{r^2}{R^2}\right)^{n-k-1}$

$+ \sum_{k=0}^{i-1} \binom{n-1}{k} (n-k-1) \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^{k} \left(1 - \frac{r^2}{R^2}\right)^{n-k-2}$.

Let T_k = the first term inside the above sum $= \binom{n-1}{k} k \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^{k-1} \left(1 - \frac{r^2}{R^2}\right)^{n-k-1}$.

Then $T_{k+1} = \binom{n-1}{k+1}(k+1) \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^{k} \left(1 - \frac{r^2}{R^2}\right)^{n-k-2} = \binom{n-1}{k}(n-k-1) \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^{k} \left(1 - \frac{r^2}{R^2}\right)^{n-k-2}$.

Now $T_0 = 0$, thus $P_i(r) = -\sum_{k=0}^{i-1}(T_k - T_{k+1}) = T_i$.

$E[r_i^2] \geq \int_0^R r^2 P_i(r) dr$

$= iR^2 \binom{n-1}{i} \int_0^R \frac{2r}{R^2} \left(\frac{r^2}{R^2}\right)^i \left(1 - \frac{r^2}{R^2}\right)^{n-i-1} dr$

$= iR^2 \binom{n-1}{i} \sum_{k=0}^{i} \binom{i}{k} (-1)^k \frac{1}{k+n-i}$. Since $n - i > 0$, using the identity $\sum_{k=0}^{n} \binom{n}{k} \frac{(-1)^k}{k+x} = x^{-1} \binom{x+n}{n}^{-1}$ (page 188 in [19]),

$$E[r_i^2] \geq iR^2 \binom{n-1}{i} \frac{1}{(n-i)\binom{n}{i}} = \frac{iR^2}{n} = \frac{i}{n\pi}.$$

To get the upper bound, we consider a node v at a corner. Let us consider another circle centered at v and with radius $R' = \sqrt{2}$, the maximum possible distance between any two nodes. If we redistribute the nodes in this circle uniformly, the average distance to the ith nearest neighbor can only increase. Thus, $E[r_i^2] \leq \frac{iR'^2}{n} = \frac{2i}{n}$.