A Distributed Algorithm to Construct Multicast Trees in WSNs: An Approximate Steiner Tree Approach

Hongyu Gong
Dept. of Electronic Engineering
Shanghai Jiao Tong University
ann@sjtu.edu.cn

Lutian Zhao
Dept. of Mathematics
Shanghai Jiao Tong University
golbez@sjtu.edu.cn

Kainan Wang
Dept. of Computer Science
Shanghai Jiao Tong University
sunnywkn@sjtu.edu.cn

Weijie Wu
School of Information Security Engineering
Shanghai Jiao Tong University
weijiewu@sjtu.edu.cn

Xinbing Wang
Dept. of Electronic Engineering
Shanghai Jiao Tong University
xwang8@sjtu.edu.cn

ABSTRACT

Multicast tree is a key structure for data dissemination from one source to multiple receivers in wireless networks. Minimum length multica modeled as the Steiner Tree Problem, and is proven to be NP-hard. In this paper, we explore how to efficiently generate minimum length mult wireless sensor networks (WSNs), where only limited knowledge of network topology is available at each node. We design and analyze a simple algorithm, which we call Toward Source Tree (TST), to build multicast trees in WSNs. We show three metrics of TST algorithm, i.e., running and energy efficiency. We prove that its running time is $O(\sqrt{n \log n})$, the best among all existing solutions to our best knowledge. We prove that TST tree length is in the same order as Steiner tree, give a theoretical upper bound and use simulations to show the ratio be only 1.114 when nodes are uniformly distributed. We evaluate energy efficiency in terms of message complexity and the number of forwardin prove that they are both order-optimal. We give an efficient way to construct multicast tree in support of transmission of voluminous data.

1. INTRODUCTION

Wireless Sensor Network (WSN) is a network of wireless sensor nodes into which sensing, computation and communication functions are integrated. Sensors are self-organizing and deployed over a geographical region [1]. Multicasting, i.e., one-to-many message transmission, is one of the most common data transmission patterns in WSNs. Tree is the topology for non-redundant data transmission. To enable efficient multicast, multicast tree has been proposed and widely used. It has not only been used for multicast capacity analysis in wireless networks [2–4], but in practice, multicast supports a wide range of applications like distance education, military command and intelligent system [5].

Many researchers have been working on constructing efficient multicast trees [6–8]. They have proposed a number of algorithms so as to minimize the routing complexity as well as achieve the time and energy efficiency (for details, please refer to the next section), but most of them did not focus on an important performance measure: the tree length. This is a critical metric since larger tree length clearly results in longer delay. To enable the messages to be forwarded farther, sensors have to increase their transmission power, causing more energy consumption as well as more serious interference to neighboring nodes. Besides, as the transmission distance increases, the messages suffer from higher probability of transmission failure.

Recently, GEographic Multicast (GEM), inspired by Euclidean Steiner Tree, was proposed for routing in dense wireless networks [9]. Formally, given a network $G = (V, E)$, the weight of each edges, and a set of terminals $S \subseteq V$, the Steiner Tree Problem is to find a tree in G that spans S with the minimum total weight [10]. This problem has been proven to be NP-hard [11], and has not been visited for a long time. Former forms of its approximate implementation were not appropriate for constructing multicast trees in WSNs for various reasons (details will be discussed in the following section.) In GEM, the authors took the first step to utilize the Steiner tree for constructing multicast trees in WSNs, achieving routing scalability and efficiency. This approach can potentially reduce the tree length, but this very simple form of utilization only considers the hop count in an unweighted graph, but not the total length of the multicast tree in a weighted graph. Further, as for the performance analysis, the statistical properties were all under the assumption that all nodes are uniformly distributed, making it difficult to tell its efficiency under a realistic network environment.

In this paper, inspired by taking the advantage of the Steiner tree property, we design a novel distributed algorithm to construct an approximate minimum-length multicast tree for wireless sensor networks, aiming at achieving energy efficiency, ease of implementation and low computational complexity, at an affordable cost on the sub-

optimality of tree length. In what follows, we call our design Toward Source Tree Algorithm, or TST for short. We quantitatively evaluate TST algorithm performance under general node distribution, and show that TST has the following satisfactory metrics:

- Its running time is $O(\sqrt{n \log n})$, the best among all existing solutions for large multicast groups.
- Its tree length is in the same order as Steiner tree, and simulation shows the constant ratio between them is only 1.114 with uniformly distributed nodes.
- Its message complexity (which we will formally define later) and the number of nodes that participate in forwarding are both order-optimal, yielding high energy efficiency for sensor networks.

The rest of paper is organized as follows. Section 2 states related work. In Section 3, we introduce our network model. In Section 4, we present our Toward Source Tree Algorithm to construct a multicast tree. In Section 5, 6 and 7, we evaluate the performance of our algorithm mainly from three aspects: multicast tree length, running time and energy efficiency separately. In Section 8, we use extensive simulations to further evaluate the performance. Section 9 concludes this paper. Due to page limit, some detailed derivations are omitted in this paper.

2. RELATED WORK

We review related work in two categories: the multicast tree construction and the approximate Steiner tree.

Multicase tree construction. Many studies focus on multicast routing in wireless networks, and useful techniques for routing have been proposed in WSN. Sanchez *et al.* proposed Geographic Multicast Routing (GMR), a heuristic neighborhood selection algorithm based on local geographic information [6]. Later Park *et al.* [15] combined distributed geographic multicasting with beaconless routing. In Localized Energy-Efficient Multicast Algorithm (LEMA), forwarding elements apply the MST algorithm locally for routing [7]. Dijkstra-based Localized Energy-Efficient Multicast Algorithm (DLEMA) finds energy shortest paths leading through nodes with maximal geographical advance towards desired destinations [8]. However, few works have considered minimizing the distance of multicast routing or providing comprehensive quantitative analysis theoretically on the performance of routing policies.

Approximate Steiner Tree. Shortest Path Heuristic (SPH) and Kruskal Shortest Path Heuristic (KSPH) add new nodes to existing subtrees through the shortest path [12]. Average Distance Heuristic (ADH) joins subtrees that contain receivers by a path passing non-receivers with minimal average distance to existing subtrees [13]. Santos *et al.* pushed forward distributed dual ascent (DA) algorithm, achieving good performance in practice [14]. The comparison of these algorithms with our TST algorithm is shown in Table 1. These algorithms were proposed for point-to-point networks. In this paper, we consider the Steiner Tree Problem in wireless sensor networks that are broadcast in nature. In addition, each node has limited computation and storage capability. Devices are usually battery-powered, therefore energy-efficiency is of great importance. Due to these specific features and requirements, existing algorithms for P2P are not suitable for WSN.

To sum up, there have been extensive existing works focusing on multicast tree construction or the approximate Steiner tree problems, but we have not found a perfect adoption of Steiner tree into constructing multicast trees.

3. NETWORK MODEL

Let us first use mathematical model to capture a wireless sensor network. We assume the network consists of n nodes in total (or we call the *network size* is n), distributed independently and identically in a unit square. Each node is assigned with a unique identifier to be distinguished from others. Each time when a source needs to transmit messages, it chooses m receivers randomly. In other words, m is the number of nodes that participate in a multicast transmission, or we call it the *multicast group size*. For our statistical analysis, we focus on the dense network and large multicast group where m and n are both very large, and $m \leq n$. This is particularly suitable to describe a wireless sensor network.

The geographical distribution of nodes is described by a density function $f(\mathbf{x})$ where $\mathbf{x}$ is the position vector. Here we allow $\mathbf{x}$ to be of any dimension; in the rest of this paper we let it be a two-dimensional vector for ease of presentation, but it does not hurt any generality. We assume $f(\mathbf{x})$ is independent of n and m. We also assume that $0 < \epsilon_1 \leq f(\mathbf{x}) \leq \epsilon_2$ where ϵ_1 and ϵ_2 are both constants, i.e., a node has a positive probability to be located at any point of this area.

To ensure the connectivity of the whole network, we set the transmission range $r = \Theta\left(\sqrt{\frac{\log n}{n}}\right)$ [16]. For all nodes, r is the same and fixed. We assume that two nodes u and v can communicate with each other directly if and only if the Euclidean distance between them, d_{uv}, is no larger than r. Every node can obtain its own geographical location, e.g., via the Global Position System (GPS). However, nodes do not know the exact location of other nodes until they receive messages containing that piece of information.

Table 2: System Parameter

n	number of all nodes in the network
m	number of receivers
r	transmission range
r_c	search coverage range
L_V	length of temporary tree
L_M	length of multicast tree

4. ALGORITHM

In this section, we describe our Toward Source Tree algorithm in detail. This algorithm consists of three phases. In the first phase, the source broadcasts a message and wakes up all receivers it chooses. In the second phase, every receiver chooses the closest neighboring receiver that has shorter Euclidean distances to the source node than the receiver node itself, and then a temporary tree can be established among all receivers. However, till the end of this stage cycles might exist. Hence we eliminate these cycles in the third phase. In what follows we describe the process in de-

Table 1: Comparison of Distributed Algorithm for Approximate Steiner Construction

Algorithm	Expected tree length	Expected time	Expected messages	Assumptions
SPH [12]	2-approximation Steiner tree, $O(\sqrt{m})$	$O(m\sqrt{\frac{n}{\log n}})$	$O(mn)$	Shortest paths need to be known; Applied in Point-to-point network.
$KSPG$ [12]		$O(m\sqrt{\frac{n}{\log n}})$	$O(mn)$	
ADH [13]		$O(m\sqrt{\frac{n}{\log n}})$	$O(n\log n + mn)$	
DA [14]	$O(\sqrt{m})$	$O(n^2)$	$O(mn^2)$	No shortest path information required; Applied in Point-to-point network.
TST	$O(\sqrt{m})$	$O(\sqrt{n\log n})$	$O(n)$	No shortest path information required; Applied in Wireless network.

tail, and we will use an example to illustrate how to generate such a tree at the end of this section.

4.1 Phase 1: Identifying Receivers

Each node has a label indicating its role in the multicast tree: "S" stands for the source and "R" for receivers. In this phase, a message containing all receivers' identifiers is sent from the source so that all nodes in the network can be aware whether they are receivers. Upon receiving this message, receivers then wake up, label themselves with "R" and be ready to participate in the multicast routing.The source will also specify its own location in this message.

This step is necessary for multicast routing since no one except the source knows which nodes the messages are destinated for. In this phase, the broadcast information will notify the nodes who are selected into the multicast group, and all receivers will be awakened.

4.2 Phase 2: Connecting All Receivers

In this phase, we first build a "temporary tree" consisting of only the multicast group members, and then find the minimum-hop shortest path between each pair of members that are directly connected in the "temporary tree". All multicast group members will be connected with the newly added relays.

Step 1: Searching Receivers in the Neighorhood

In this step, each multicast member chooses an appropriate neighbor to connect to. The neighboring member selection criteria is: each member chooses the closest one from the set of members that have shorter Euclidean distances to the source node than this node itself. If no such neighboring member can be found, then this multicast member directly connects to the source.

When a member tries to contact its neighbor members, it is regarded as the sender that sends *request message*. Its form is: <*sender id, sender location, location of previous hop, coverage range* r_c*, node sequence, total hop* H*, path length* p>. *Sender id* is used to identify the multicast members sending the *request message*, and *path length* can be updated with the location of previous hop and current hop. The coverage range r_c sets the range within which the multicast member searches for its neighboring members. The Euclidean distance between the sender and current node can be calculated with sender location, and messages will be discarded if the distance is larger than r_c. *Node sequence* records in order the nodes through which this message has passed, which acts as a guide for response from neighboring receivers so that the response can be routed via the available path. The hop count H is the number of hops the message has passed through, and p is the path length the message have been through when it reaches the current node.

Algorithm 1 Neighbor Request from Multicast Members

```
for all receiver R in a multicast group do
    the number of request session: k ← 0
    coverage range: r_c ← r
    time out interval: T_0 ← Θ(2^k log n)
    set the node sequence as {R}
    total hop: H ← 0
    path length: p ← 0
    forward the request message to its neighborhood
    while no response is received when time is out for the
    k^th request session do
        k ← k + 1
        r_c ← 2^k r
        T_{k+1} ← 2T_k
        set the node sequence as {R}
        H ← 0
        p ← 0
        forward the request message to its neighborhood
    end while
end for
```

In each search session, the member broadcasts the *request message* within search coverage range. The sender sets an appropriate timeout interval. Once the sender receives replies from neighboring nodes, the search session terminates. Then it enters step 2. However, if time runs out and no reply is obtained, it means that no appropriate neighboring members are found. The sender then doubles its search range and initiates another search. In Algorithm 1, we show how a multicast group member connects to their neighboring members or the source.

A node may receive more than one *request message* from the same sender. If it is within *coverage range*, it will choose the one with the fewest hops among all the messages. If the numbers of hops are the same, it picks out the message with the shortest path length. Then it modifies this message. It adds itself to the *node sequence*, increases the *hop count* by

1 and calculate new *path length* given the location of the previous hop. With these information updated, it forwards the message. Algorithm 2 describes how nodes deal with *request messages* in detail.

Algorithm 2 Request Forwarding

```
1: for all node u receiving request message do
2:    dist = || location of u - sender location ||
3:    if dist < r_c then
4:       add u to node sequence
5:       H ← H + 1
6:       newDist = ||location of u - location of previous hop||
7:       p ← p + newDist
8:       forward the request message to its neighborhood
9:       if u is in the multicast group then
10:         nodeSourceDist = || location of u - source location ||
11:         senderSourceDist = || sender location - source location ||
12:         if nodeSourceDist < senderSourceDist then
13:            send respond message back to the sender
14:         end if
15:      end if
16:   end if
17: end for
```

When a multicast member finds it closer to the source than the sender of the *request message*, it might be chosen as the neighbor by the sender. Therefore, this member will choose a path to the sender and respond with the *respond message*. The form of the *respond message* is: *<sender id, respondent id, node sequence, total hop H, path length p>*. The *respond message* can be routed with the path information provided by the node sequence.

Step 2: Connecting to the Nearest Neighbor

With *respond messages*, every member selects the closest neighbor. Once a neighbor is chosen, the *connect message* is forwarded via the minimum-hop shortest path. The *connect message* is used to establish a connection between nodes in the multicast group. At the same time, all relay nodes on the minimum-hop shortest path record this pair of members, previous hop and the next hop on the path. When all receivers send the *connect message*, a "temporary tree" among all mutlicast group members including the source is constructed.

4.3 Phase 3: Eliminating Cycles

In Phase 2, we construct a "temporary tree" made up of multicast group members. However, when other nodes are added to it as relays, cycles might be formed. In particular, when paths connecting different pairs of multicast members share the same relay nodes, such node may receive redundant information, which indicates that cycles come into being. Therefore, we check the existence of cycles in this phase and eliminate them if any.

Suppose a node u acts as a relay for k ($k > 1$) pairs of nodes in the multicast group, which are directly connected in the temporary tree, denoted as (R_{11}, R_{12}), (R_{21}, R_{22}),..., (R_{k1}, R_{k2}). Let us assume that in each pair, R_{i1} is closer to source than R_{i2} ($1 \leq i \leq k$). A relay stores its previous and the next hop of the path from R_{i1} to R_{i2}, and they are denoted as PH_i and NH_i respectively. Then it chooses one pair randomly, say, (R_{j1}, R_{j2}) and keeps the information: (R_{j1}, R_{j2}, PH_j, NH_j). For other pairs (R_{i1}, R_{i2}) where $R_{i1} \neq R_{j1}$, the relay modifies their information as (R_{j1}, R_{i2}, PH_j, NH_i). Define a set Q, where $Q = \{q \mid q =< R_{i1}, R_{i2}, PH_i >, \forall\ R_{i1} \neq R_{j1}\}$. Last, it sends "*Eliminate* message Q" and its previous hops delete unnecessary edges accordingly. In Algorithm 3, we show how to wipe out the cycles.

Algorithm 3 Cycle Elimination

```
1: for all node u engaged in paths between k pairs of members do
2:    choose an integer j such that 1 ≤ j ≤ k
3:    for all i such that 1 ≤ i ≤ k and i ≠ j do
4:       forward Eliminate message Q_i =< R_i1, R_i2, PH_i >
5:    end for
6: end for
7: for all node w receiving "Eliminate message Q_i" do
8:    if w is exactly PH_i then
9:       if w is not in the multicast group then
10:         PH_i ← previous hop of w on path (R_i1, R_i2)
11:         forward the modified Q_i to the previous hop
12:         eliminate information: (R_i1, R_i2, PH_i, NH_i)
13:      end if
14:   end if
15: end for
```

4.4 Proof of Tree Topology

The previous subsections describe how we can connect mutlicast group members using our TST algorithm. Now let us prove that the topology constructed by TST algorithm is exactly a tree. We first show that temporary tree formed in the second phase has a tree topology in Lemma 4.1. But relays are added into the temporary tree to connect receivers, which might result in the existence of cycles. In Theorem 4.1 we show that cycle elimination can in fact guarantee the tree topology.

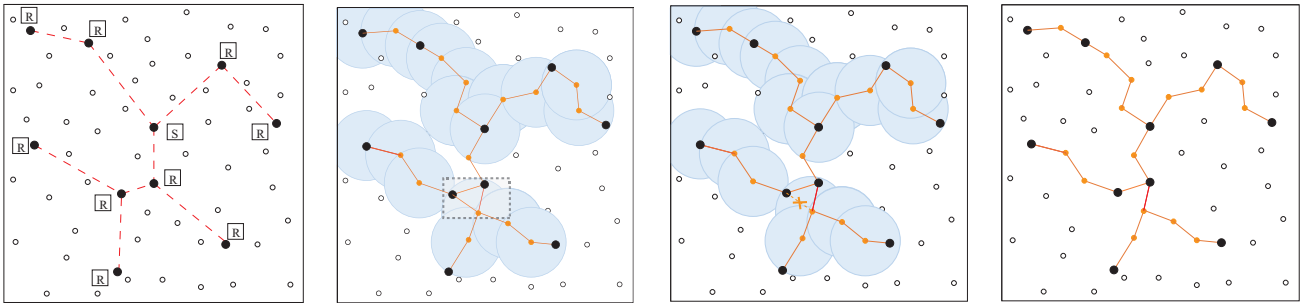
Lemma 4.1. *The temporary tree connecting m receivers has a tree topology.*

PROOF. We prove it by contradiction. Suppose that a cycle exists and k nodes are contained in this cycle, denoted by $n_1, n_2, ..., n_k$. All of them are multicast members. Without loss of generality, we assume n_i selects n_{i+1} as its neighboring member, $i = 1, \cdots, k-1$, and that n_k chooses n_1 as its neighbor. According to the criteria of neighbor selection, we know that n_{i+1} is closer to source than n_i for $i = 1, \cdots, k-1$. Therefore, n_k must be closer to source than n_1. Due to the fact that n_k's selection is n_1, we conclude that n_1 must be closer to source than n_k, which is a contradiction. This concludes the proof. □

Based on Lemma 4.1, we have the following theorem:

Theorem 4.1. *The topology connecting nodes generated by TST algorithm is a tree that spans all multicast group members.*

PROOF. The existence of cycles means that some nodes in the multicast tree may receive redundant messages, i.e., some nodes have more than one previous hop. For these nodes, they send "Eliminate messages" and ensure that they have only one previous hop. When all nodes in the multicast tree have only one previous hop, no cycle exists.



(a) Building a temporary tree spanning multicast group members

(b) Adding relay nodes

(c) Eliminating redundant edges and maintaining the topology of tree

(d) Constructing the multicast tree with relays added

Figure 1: Steps of the TST algorithm

Multiple previous hops also indicate that multiple paths may exist between two nodes. Once some previous hops are unnecessary, the paths involving these hops can also be eliminated. Thus Algorithm 3 can eliminate these unnecessary paths, and this completes our proof. □

4.5 Illustration

We use an example to illustrate our TST algorithm. Nodes are distributed in the unit square as shown in Figure 1(a). Solid nodes represent source nodes labeled by "S", or multicast members labeled by "R". The hollow nodes can be chosen as relays. The first step is to build a temporary tree spanning all multicast members. The dashed lines denote virtual connections between two members. Then nodes on the minimal-hop shortest path are engaged as relays between two neighboring members. They form the topology as shown in Figure 1(b). Note that there exists a cycle marked with dotted rectangular box. The last step is to eliminate unnecessary edges as is done in Figure 1(c). Finally we obtain the multicast tree as is shown in Figure 1(d).

5. LENGTH ANALYSIS

The previous section described our Toward Source Tree algorithm. In the next three sections, we will discuss its performance in terms of tree length, time complexity, and energy efficiency. In this section, we discuss the length of TST. We first obtain the length of temporary tree, first assuming uniform distribution nodes and then extending to a general setting. Next we explore the length of minimal-hop path that connects two receivers. Combining the length of temporary tree and the path, we can derive the upper bound for the multicast tree length.

5.1 Temporary Tree in Uniform Distribution

We start by discussing the tree length of the temporary tree.

Lemma 5.1. *Assume nodes are uniformly distributed in a unit square. The expected length of the temporary tree spanning m receivers is upper bounded by $c\sqrt{m}$, where $c = 5.622$.*

Proof. See Appendix A. □

5.2 Temporary Tree in General Distribution

Based on the conclusions of tree length in uniform distribution, we further study the case that nodes are non-uniformly distributed. We partition the unit square into k small squares, where $m = k^{1+\gamma}$ and $0 < \gamma < 1$. We construct trees among nodes in each square, and then connect nodes in different cells so that all nodes in the network are connected. For each square, the source is outside the square and we still apply the TST algorithm for the tree construction. Lemma 5.2 can estimate the intra-square edge length, and we study the inter-square edge length in Lemma 5.3. With both inter- and intra- square edge estimation, we derive upper bound for temporary tree length in general distribution.

Lemma 5.2. **(Intra-square edges)** *Let m nodes be independently distributed in a unit square with density function $f(\mathbf{x})$. The source S is located outside the square. Let each node connect to the closest neighbor that has shorter Euclidean distance to S than the node itself. If no such receiver exists, it doesn't connect to other nodes. A tree can be constructed among m nodes, and the expected length of such a tree is upper bounded by $c\sqrt{m}$, where $c = 5.622$.*

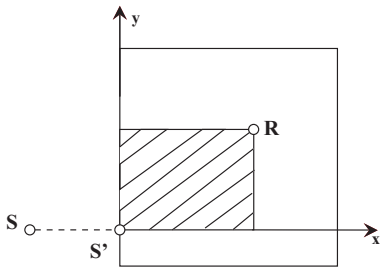


Figure 2: Approximate neighbor region when the source is located outside the square

Proof. For those nodes that not closest to S, they can always find another node to connect to. For the node that is closest to S, it will be connected to by other nodes. We can prove that the topology formed by m nodes is exactly a tree with Lemma 4.1. We denote this tree as T.

There are two differences of this lemma from Lemma 5.1. One is that the source is located outside the square, and

the other is that a node won't connect to others when it can't find another one that has shorter Euclidean distance to the source. Now we find a point S' that is closest to S in the boundary of square region, as is shown in Figure 2. With S' as the source, a temporary tree as mentioned in TST algorithm can be established spanning all nodes in the network. We denote the temporary tree as T'. In the following we demonstrate that the tree length of T' can be used to estimate the upper bound of length of T.

For a node R, we use N_R to denote the regions where nodes might be selected by R as a its neighbor. We use a rectangular region as approximate neighbor region N'_R, and $N'_R \subseteq N_R$. The approximate neighbor region is the region marked with parallel lines in Figure 2. We use the method adopted in the proof of Lemma 5.1 to estimate the length of T.

It can be observed that the approximate neighbor regions are the same in both cases that we take S as the source and that we take S' as the source. There are some details that need to be clarified. Firstly, when we only consider the nodes in approximate neighbor region, the estimated tree length is larger than actual length, because we ignore the nodes that are closer to node R. Secondly, if neighbors exist in the approximate region, the estimations of edge length are the same for both T and T'. Thirdly, if no neighbor is found in approximate region for a node, we assume that it doesn't connect to others in T but it connects to the source in T' in our calculation. From the analysis above, we can conclude that length of T is upper bounded by the length of T'.

Also recall that in our proof of Lemma 5.1, and $5.622\sqrt{m}$ is the upper bound for temporary tree length wherever S' is. In summary, we can directly use estimated tree length in Lemma 5.1 as the upper bound of the tree length of T. This completes our proof. □

Lemma 5.3. **(Inter-square edges)** *Let m nodes be independently distributed in a unit square with density function $f(\mathbf{x})$. The unit square $[0,1] \times [0,1]$ can be partitioned into k square cells with edge length of $\frac{1}{\sqrt{k}}$, where $m = k^{1+\gamma}$ and $0 < \gamma < 1$. The length of inter-square edges connecting k cells in the unit square is $o(\sqrt{m})$.*

Proof. We know that the expected number of nodes in each square cell is greater than $\frac{m}{k}\epsilon_1 = k^{\gamma}\epsilon_1$. To compute the minimal distance between two nodes in adjacent squares, we partition the cell with edge length of $\frac{1}{\sqrt{k}}$ into smaller grids with edge length of $\frac{1}{k^{\alpha}}$, where $\alpha > \frac{1}{2}$.

We claim that if $\alpha - \gamma < \frac{1}{2}$, the minimal length between two adjacent cells is in an order of $o\left(\frac{1}{\sqrt{k}}\right)$. This comes from the observation that we can connect adjacent cells by connecting nodes in adjacent grids whose edge length is $\frac{1}{k^{\alpha}}$, as is shown in Figure 3. In this figure, the yellow and the black squares are two adjacent cells with edge length of $\frac{1}{\sqrt{k}}$. The blue grids contained in them are the smaller squares with edge length of $\frac{1}{k^{\alpha}}$. Green lines are used to show that nodes in the adjacent grids are connected.

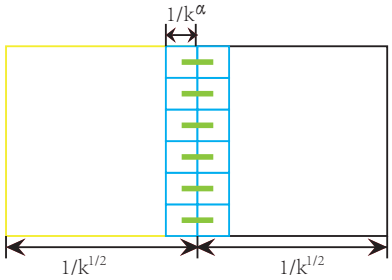


Figure 3: Inter-square edges between nodes in adjacent square cells

As we can see from Figure 3, for two adjacent cells with edge length of $\frac{1}{\sqrt{k}}$, $k^{\alpha-1/2}$ pairs of nodes in adjacent grids might exist. Denote P_1 as the probability that a node exists in a grid with edge length of $1/k^{\alpha}$. Since the area of each square is very small, we can regard nodes in the same square uniformly distributed. We have

$$P_1 = 1 - (1 - \frac{1}{k^{2\alpha-1}})^{\frac{m\epsilon_1}{k}}.$$

Denote P_2 as the probability that nodes exist in both of the adjacent adjacent grids.

$$P_2 = 1 - P_1^2.$$

There are $k^{\alpha-1/2}$ pairs of nodes in adjacent grids, and we denote P as the probability that at least one pair exist. We have

$$P = 1 - P_2^{k^{\alpha-1/2}}.$$

Hence we have

$$P = 1 - (1 - (1 - (1 - \frac{1}{k^{2\alpha-1}})^{\frac{m\epsilon_1}{k}})^2)^{k^{\alpha-1/2}} \quad (1)$$

In order to let k squares connected by inter-square edges, it should hold that $P^k \to 1$. Therefore, we need the following condition

$$1 - k^{-\frac{1}{k^{\alpha-1/2}}} \gg (1 - (1 - \frac{1}{k^{2\alpha-1}})^{\frac{m\epsilon_1}{k}})^2. \quad (2)$$

The expression that $r_1(k) \gg r_2(k)$ means that $r_2(k)/r_1(k) \to 0$ as $k \to \infty$. Condition (2) is equivalent to condition (3).

$$\frac{\log k}{k^{-1/2+\gamma+\alpha\epsilon_1}} \ll \frac{1}{k^{2\alpha-1}}, \quad (3)$$

Condition (3) can be satisfied when $\alpha < \gamma + \frac{1}{2}$. With $\frac{1}{2} < \alpha < \gamma + \frac{1}{2}$, we can evaluate P. By (1) it can be verified that

$$P \sim 1 - \exp(-2\epsilon_1 k^{1/2-\alpha+\gamma} - \frac{\epsilon_1}{\log 2} k^{1/2-\alpha}). \quad (4)$$

It is easy to show that $P^k \to 1$ with the expression (4), which means such pairs of nodes exist for all adjacent cells with high probability. Since $(1-P)$ is exponentially decaying to zero, the expectation of total path length needed to connect k cells is

$$k^{1-\alpha}P^k + k^{1/2}k(1-P) \sim k^{1-\alpha} = o(k^{1/2}) \quad (5)$$

Due to the fact that $k = o(m)$, the expected path length for inter-square connection is in the order of $o(\sqrt{m})$. □

Lemma 5.4. *Let m nodes be independently distributed with density function $f(\mathbf{x})$. The expectation for the total length of temporary tree $E[L_V]$ is smaller than $c\sqrt{m}$. We have $E[L_V] \leq c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}$, where $c \approx 5.622$.*

PROOF. See Appendix B. □

5.3 Path With Minimal Hops

Receivers are connected by the minimal-hop path. In this part, we study the relationship between the path length and Euclidean distance between two nodes.

Lemma 5.5. *Let n nodes be independently and identically distributed over $[0,1]\times[0,1]$ with distribution function $f(\mathbf{x})$. Suppose that the Euclidean distance between two nodes u and v is x. The following properties hold:*

1. *The expectation of fewest relays that are needed to connect u and v converges to $\frac{x}{r}$ as n approaches ∞;*
2. *The length expectation of the path connecting uv and involving the fewest relays converges to x.*

PROOF. See Appendix C. □

5.4 Multicast Tree

We divide the $[0,1]\times[0,1]$ network region into k squares. In each square, we construct a tree and connect nodes with intra-square edges. Adjacent squares are connected by the inter-square edges. All nodes are connected by intra- and inter- square edges, and they can be used to estimate the tree length.

Theorem 5.1. *Let n nodes be independently and identically distributed in a unit square and their distribution satisfies the density function $f(\mathbf{x})$. We construct a multicast tree spanning m receivers as well as the source with TST algorithm. When m and n are both very large, the expected length of the tree is upper bounded by $c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}$, where $c = 5.622$.*

PROOF. Denote $e_{i,j}$ as the edge connecting Receivers i and j in the temporary tree T_V, $l_{i,j}$ as the length of the minimal-hop path between the two receivers. Since redundant edges will be eliminated, $E(L_M) \leq \sum_{e_{i,j}\in T_V} E(l_{i,j})$. And the path length converges to Euclidean distance as network size goes to ∞ according to Lemma 5.5. So we have:

$$E(L_M) \leq c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}. \tag{6}$$

Remark: We derive an upper bound for the Toward Source Tree, but it is not a tight bound. In Section 8, we will show that TST algorithm has even better empirical performance than our theoretical bound. □

Lemma 5.6. *Suppose X_i, $1 \leq i < \infty$, are independent random variables with distribution μ having compact support in $\mathbb{R}^d$, $d \geq 2$. If the monotone function ψ satis es $\psi(x)\ x^\alpha$ as $x \to 0$ for some $0 < \alpha < d$, then with probability 1*

$$\lim_{n\to\infty} n^{-(d-\alpha)/d} M(X_1X_2,...,X_n) = \tag{7}$$

$$c(\alpha,d)\int_{\mathbb{R}_d} f(\mathbf{x})^{(d-\alpha)/d}d\mathbf{x} \tag{8}$$

Here f denotes the density of the absolutely continuous part of μ and $c(\alpha,d)$ denotes a strictly positive constant which depends only on the power α and the dimension d [17].

Given a graph with some nodes and edges, building a minimal length tree spanning a subset of nodes with relays appropriately added is formulated as Steiner Tree Problem. If no relay nodes are allowed, then the tree with minimal length is called minimal spanning tree. However, Steiner tree can only optimize tree length by a constant ratio compared with the minimal spanning tree.

Lemma 5.7. *Let P be a set of n points on the Euclidean plane. Let $l_s(P)$ and $l_m(P)$ denote the lengths of the Steiner minimum tree and the minimum spanning tree on P respectively. The inequality holds: [19]*

$$l_s(P) \geq \frac{\sqrt{3}}{2} l_m(P) \tag{9}$$

Combining the two lemmas above, we can conclude that the length of Steiner tree spanning m receivers is:

$$L_{ST} \geq \frac{\sqrt{3}}{2} c_1\sqrt{m}\int_{[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x} \tag{10}$$

Here c_1 is the constant equal to $c(1,2)$ mentioned in Lemma 5.6. Roberts estimated that $c_1 = 0.656$ [18].

From (6) and (10), we prove that the length of Toward Source Tree is in the same order as that of Steiner tree, and the difference between them is only a constant ratio.

6. TIME ANALYSIS

Now let us derive the time complexity of TST.

Theorem 6.1. *Let n nodes be independently and identically distributed in unit square. The running time of TST algorithm is $O(\sqrt{n\log n})$.*

PROOF. There are three serial phases in TST algorithm, so we discuss the time cost of each phase one by one.

In Phase 1, the messages containing location information of the source are broadcast in the network. The furthest distance between the source and another node is $O(1)$, so at most $O\left(\frac{1}{r}\right)$ relays are needed for a message to reach one node. In expectation, there are $\pi r^2\epsilon_2 n = O\left(nr^2\right)$ nodes within transmission range of a node and hence a node has to wait for $O\left(nr^2\right)$ time slots to transmit a message. The time needed for Phase 1 is:

$$O\left(\sqrt{\frac{n}{\log n}}\right) \leq E(t_1) \leq O\left(nr\right). \tag{11}$$

In Phase 2, the dominant time cost is searching for neighboring receivers. In the k^{th} search session, the coverage range is $2^k r$. We need $O\left(2^k\right)$ relays to forward *request messages* from one receiver to any other nodes within its search coverage range. Since the coverage range does not exceed $\sqrt{2}$, the number of search sessions cannot be more than $\left\lceil \log_2 \frac{\sqrt{2}}{r} \right\rceil$.

$$E(t_2) \leq O\left(\sum_{i=0}^{\left\lceil \log_2 \frac{\sqrt{2}}{r} \right\rceil} 2^i nr^2\right) \leq O(nr). \tag{12}$$

In Phase 3, the worst case is that relays on the path whose length is $O(1)$ form cycles. Time for cycle elimination is

$$E(t_3) = O\left(\frac{1}{r}nr^2\right) = O(nr). \tag{13}$$

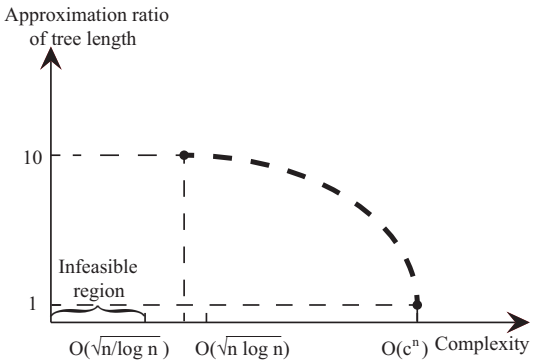


Figure 4: Relationship between tree length and time complexity

The total running time is $E(t) = \sum_{i=1}^{i=3} E(t_i)$, so we have

$$O\left(\sqrt{\frac{n}{\log n}}\right) \leq E(t) \leq O(\sqrt{n \log n}). \tag{14}$$

which completes our proof. □

Remark: For any algorithm to construct a multicast tree among a group of nodes, broadcast in Phase 1 is necessary. Since no node has a knowledge of the multicast group except the source, such information has to be forwarded to every node in the network so that they can know whether they should participate in multicasting. The lower bound of time for multicast tree construction is $O\left(\sqrt{\frac{n}{\log n}}\right)$. Since TST achieves the time complexity upper bounded by $O(\sqrt{n \log n})$, the minimal time cost to construct a multicast tree is also upper bounded by $O(\sqrt{n \log n})$. Hence the time complexity of TST algorithm shares the same upper and lower bounds as the minimal time cost, and the ratio between these two bounds is only $O(\log n)$.

The length of multicast trees have a great influence on communication quality in terms of transmission delay and wireless interference. Construction of minimum-length trees is an NP-hard problem, and takes exponential time. Approximate algorithms achieve larger tree length with lower time complexity. Now we explore the relationship between tree length and time complexity in Figure 4. Since the lower bound of time needed for multicast tree construction is $O\left(\sqrt{\frac{n}{\log n}}\right)$, the region with time complexity smaller than $O\left(\sqrt{\frac{n}{\log n}}\right)$ is infeasible. Accurate solution to Steiner tree problem achieves the approximation ratio of 1 at the cost of exponential time, and our algorithm achieves the ratio of 10. The approximation ratio of other algorithms like those in Table 1 approaches 1 but they have larger time costs.

7. ENERGY EFFICIENCY

Energy is a primary consideration in wireless sensor networks since sensors are battery-powered and their energy is limited. We consider the following factors: 1) the energy consumed to construct such multicast trees; and 2) the energy needed to send messages along the tree constructed by this algorithm. The former one is usually measured by the amount of exchanged messages to run distributed routing algorithms; and the latter directly depends on the number of nodes participating in the transmission. We focus on both aspects.

7.1 Message Complexity

The following theorem quantifies the message complexity in TST.

Theorem 7.1. *Let n nodes be independently and identically distributed in the unit square. The message complexity of TST algorithm is $O(n)$.*

PROOF. See Appendix D. □

Remark: Since each node needs a message telling them whether they are chosen as receivers, the lower bound of message complexity is $O(n)$. Hence TST algorithm is an order-optimal solution in terms of message complexity.

7.2 Number of Forwarding Nodes

Since the transmission range is fixed, the number of transmitters in the tree determines the energy consumption for information propagation. We evaluate the number of forwarding nodes in this subsection.

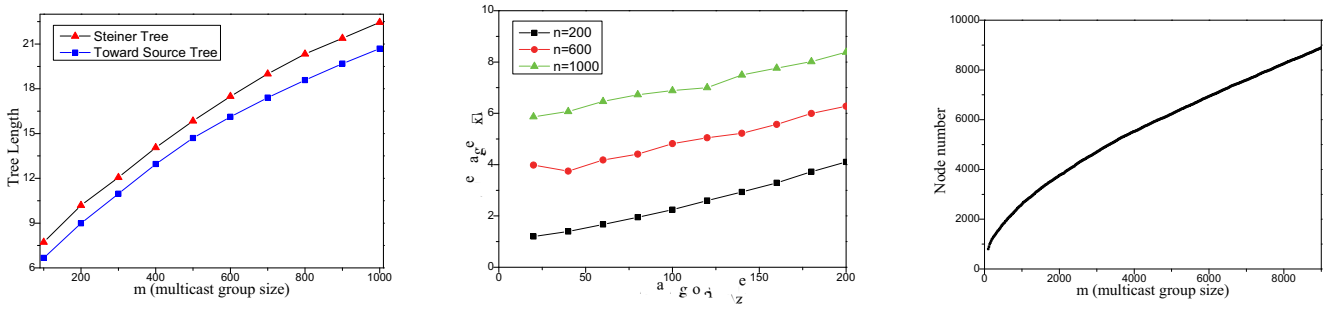
Theorem 7.2. *Let n nodes be independently and identically distributed in the unit square. The number of forwarding nodes in the multicast tree is*

$$N_{TST} = \begin{cases} \Theta\left(\sqrt{\frac{mn}{\log n}}\right), & m = O\left(\frac{n}{\log n}\right); \\ \Theta(m), & m = \omega\left(\frac{n}{\log n}\right). \end{cases} \tag{15}$$

When $m = O(n/\log n)$, the number of forwarding nodes is order-optimal.

PROOF. Let T_V be the virtual tree, $e_{i,j}$ be an edge in the virtual tree connecting two receivers i and j, and $d_{i,j}$ be the Euclidean distance between them. When m is small, relay nodes form the dominant part of the forwarding nodes in our multicast tree. The total number of transmitting nodes, N_{TST} in the Toward Source Tree is: $N_{TST} = \Theta\left(\sum_{e_{i,j} \in T_V} \frac{d_{ij}}{r}\right)$ $= \Theta\left(\frac{\sqrt{m}}{r}\right)$. As m grows larger, receivers are close to each other and thus fewer relay nodes are added. Therefore, receivers are dominant in the multicast tree, $N_{TST} = \Theta(m)$. We should discuss the number of forwarding nodes in two cases, and there exists a critical value for m that determines in which case it should be discussed. The critical value satisfies: $\Theta\left(\frac{\sqrt{m_c}}{r}\right) = \Theta(m_c)$, so $m_c = \Theta\left(\frac{1}{r^2}\right)$.

Denote N_{min} as the minimal number of relay nodes that are engaged in propagating the messages from one source to m receivers. [2] gives the lower bound of N_{min} under the assumption that all nodes are uniformly distributed. Now we use its method and explore N_{min} in the case of general distribution. When m is small, the distance between two receivers is large compared with the transmission range. $N_{min} = \Omega\left(\frac{\sqrt{m}}{r}\right)$. This lower bound is achievable with our algorithm, so $N_{min} = \Theta\left(\frac{\sqrt{m}}{r}\right)$. When m is very large, there exist many receivers within the transmission range of one node, so that one transmission can deliver messages to a large number of receivers. In this case, we only need to choose a connected dominating set from m receivers, and



(a) Tree length comparison (b) Message complexity (c) Number of forwarding nodes

Figure 5: Algorithm evaluation when nodes are uniformly distributed

N_{min} is exactly the size of minimum connected dominating set. We will give the definitions of both *connected dominating set* and *minimum connected dominating set*.

Definition 7.1 (***Connected dominating set***). *D is the connected dominating set of a graph G if and only it satis es two properties:*

1. *Any node in D can reach any other node in D by a path that stays entirely within D.*
2. *Every vertex in G either belongs to D or it is adjacent to a vertex in D.*

Definition 7.2 (***Minimum connected dominating set***). *MD is the minimum connected dominating set of graph G if MD is the connected dominating set containing the smallest number of nodes.*

We still need to discuss N_{min} in two cases. There also exists a critical value m_d, and $\Theta(m_d) = \Theta\left(\frac{\sqrt{m_d}}{r}\right)$, so $m_d = \Theta\left(\frac{n}{\log n}\right)$.

$$N_{min} = \begin{cases} \Theta\left(\sqrt{\frac{mn}{\log n}}\right), & m = O\left(\frac{n}{\log n}\right); \\ \Omega\left(\frac{n}{\log n}\right), & m = \omega\left(\frac{n}{\log n}\right). \end{cases} \quad (16)$$

From (15) and (16), we can find when $m = O\left(\frac{n}{\log n}\right)$, the number of forwarding nodes in the multicast tree is optimal in order sense. □

Remark: When $m = \omega\left(\frac{n}{\log n}\right)$, the number of forwarding nodes in TST tree may not be order-optimal. However, in graph theory, finding the minimum connected dominating set of a given graph is proved to be NP-complete [20]. And it also requires global information of network topology. So we consider it an acceptable sacrifice of energy to achieve the feasibility and time-efficiency in practice.

8. PERFORMANCE EVALUATION

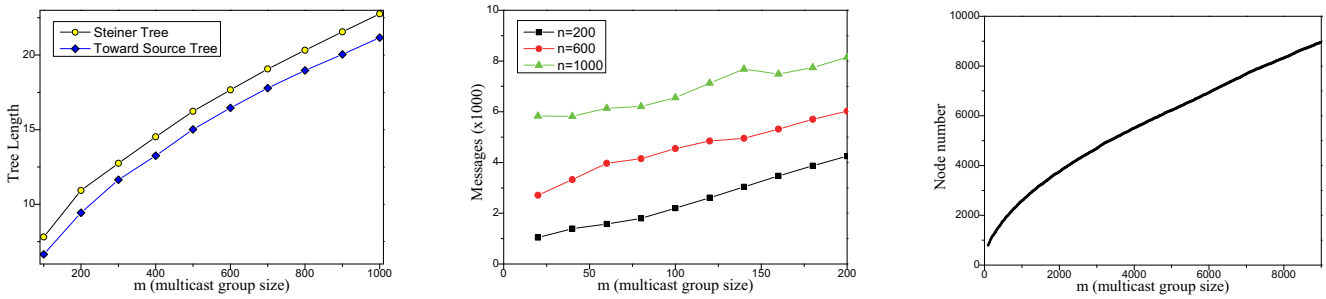
We perform extensive simulations to evaluate the empirical performance of Toward Source Tree algorithm, in terms of the length of the multicast tree, message complexity and the number of forwarding nodes engaged in the tree. We mainly consider two common distribution patterns: uniform distribution and normal distribution.

8.1 Uniform Distribution

We first consider nodes are uniformly distributed in a unit square and transmission range is set to be $r = \sqrt{\frac{\log n}{n}}$. We explore the effect of multicast group size m on the tree length. Assuming that the network size is fixed as 1000, we obtain the lengths of the Stenier tree and TST tree when the value of m varies. The length of the Steiner tree can be obtained via NewBossa in [21]. Two curves in Figure 5(a) describe the relationship between m and the length of the Toward Source Tree as well as the Steiner Tree. It is shown that the length of TST tree is larger than that of the Steiner Tree but quite close to it. According to simulation statistics, the ratio of the tree length achieved by the two algorithms is 1.114 on average. When nodes are uniformly distributed, Toward Source Tree is a good approximation of the Steiner Tree.

Then we evaluate the message complexity in the construction of TST tree, and explore the relationship among the network size n, the multicast group size m and message complexity. We set the network size $n = 200, 600, 1000$ respectively, and record the quantity of exchanged messages when multicast group size m varies. Different curves correspond to different network sizes in Figure 5(b). As can be seen in the figure, the quantity of exchanged messages increases with the multicast group size as well as the network size. It is quite intuitive that the larger network size can result in more exchanged messages. Since the transmission power necessary to maintain the connectivity is less in dense networks than in sparse networks, more relays are engaged in multicasting as the network size increases. Hence messages used to contact nodes and inquire routing information become more.

We find that more messages are exchanged when we fix the network size and add more multicast group members. This result is not so intuitive. On the one hand, multicast group members become closer to each other when the multicast group size increases, so they need to search for the appropriate neighbors in a smaller coverage range. Fewer nodes are inquired within the coverage range, and fewer *request messages* are sent. On the other hand, when a multicast group member looks for neighbors, more other members might find they are closer to the source. Hence more *response messages* might be sent back. Total messages increase as more nodes join the multicast group.



(a) Tree length comparison (b) Message complexity (c) Number of forwarding nodes

Figure 6: Algorithm evaluation when nodes satisfy normal distributed

Finally, we consider the number of forwarding nodes engaged in multicasting. To derive the statistical properties of TST, we set the network size as 100, 000. In Figure 5(c), when the multicast group is small, the first part of the curve indicates that the number of forwarding nodes is $O(\sqrt{m})$. As there are more multicast group members, the number of forwarding nodes grows linearly with the group size.

8.2 Non-uniform Distribution

We randomly choose the location of source, $\mathbf{x_s}$, within the unit square first. For the case of non-uniform distribution, we consider that nodes satisfy the normal distribution: $f(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}-\mathbf{x_s}\|^2}{2}}$, where $\|\mathbf{x} - \mathbf{x_s}\|$ is the Euclidean distance between the node and the source. It is possible that the nodes are scattered outside unit square during simulations. If this happens, we relocate these nodes until they are within this unit square.

We evaluate the TST algorithm still in terms of tree length, message complexity and the number of forwarding nodes. We find the results are quite similar to those in the uniform distribution. In Figure 6(a), the length of TST tree is only a little larger than the optimal length in our simulations. The statistics show that the ratio between them is 1.110 on average. As for the message complexity, Figure 6(b) shows that more messages are exchanged among more multicast group members or in denser networks, and the quantity of messages is still $O(n)$. As is shown in Figure 6(c), the number of transmitting nodes in the multicast tree is linear with $\sqrt{m}$, and becomes linear with m when more nodes participate in the multicast group assuming that the network size keeps unchanged.

9. CONCLUSION

In this paper, we propose a novel algorithm, which we call Toward Source Tree, to generate approximate Steiner Trees in wireless sensor networks. The TST algorithm is a simple and distributed scheme for constructing low-cost and energy-efficient multicast trees in the wireless sensor network setting. We prove its performance measures in terms of tree length, time complexity, and energy efficiency. We show that the tree length is in the same order as, and is in practice very close to, the Steiner tree. We prove its running time is the shortest among all existing solutions. We prove that its message complexity and the number of nodes that participate in forwarding are both order-optimal, yielding high energy efficiency for applications.

10. REFERENCES

[1] M. Segal J. Crowcroft and L. Levin. Improved structures for data collection in wireless sensor networks. In *Proc. of IEEE INFOCOM*, pages 1375–1383, 2014.

[2] H. R. Sadjadpour Z. Wang and J. J. Garcia-Luna-Aceves. A unifying perspective on the capacity of wireless ad hoc networks. In *Proc. of IEEE INFOCOM*, 2008.

[3] X. Y. Li. Multicast capacity of wireless ad hoc networks. *IEEE/ACM Trans. Netw.*, 17(3):950–961, 2009.

[4] X. Liu S. Shakkottai and R. Srikant. The multicast capacity of large multihop wireless networks. *IEEE/ACM Trans. Netw.*, 18(6):1691–1770, 2010.

[5] U. Varshney. Multicast over wireless networks. *Communications of ACM*, 45(12):31–37, 2002.

[6] P. M. Ruiz J. A. Sanchez and I. Stojmenovic. Gmr: Geographic multicast routing for wireless sensor networks. In *Proc. of IEEE SECON*, pages 20–29, 2006.

[7] J. A. Sanchez and P. M. Ruiz. Lema: Localized energy-efficient multicast algorithm based on geographic routing. In *Proc. of 31st IEEE Conference on Local Computer Networks*, pages 3–12, 2006.

[8] M. Bartosz B. Musznicki and P. Zwierzykowski. Dijkstra-based localized multicast routing in wireless sensor networks. In *Proc. of IEEE CSNDSP*, pages 1–6, 2012.

[9] G. Morabito L. Galluccio and S. Palazzo. Geographic multicast (gem) for dense wireless networks: protocol design and performance analysis. *IEEE/ACM Trans. Netw.*, 21(4):1332–1346, 2013.

[10] T. Rothvoss J. Byrka, F. Grandoni and L. Sanità. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1):6, 2013.

[11] R. M. Karp. *Reducibility among Combinatorial Problems*. Springer, 1972.

[12] F. Bauer and A. Varma. Distributed algorithms for multicast path setup in data networks. *IEEE/ACM Trans. Netw.*, 4(2):181–191, 1996.

[13] G. Lo. Re L. Gatani and S. Gaglio. An efficient distributed algorithm for generating multicast distribution trees. In *Proc. of IEEE ICPP Workshops*, pages 477–484, 2005.

[14] L. M. A. Drummond M. Santos and E. Uchoa. Distributed dual ascent algorithm for steiner problems in networks. In *Anais do Simpósio Brasileiro de Redes de Computadores*, pages 381–396, 2007.

[15] et al. H. Park, J. Lee. Distributed multicast protocol based on beaconless routing for wireless sensor networks. In *Proc. of IEEE ICCE*, pages 522–523, 2013.

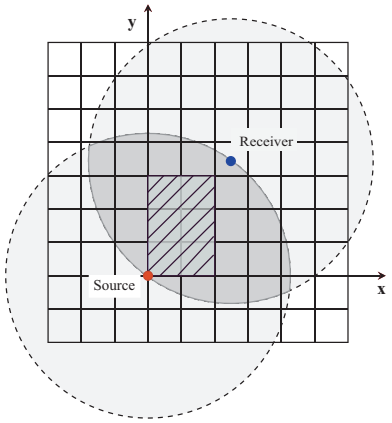


Figure 7: Two-dimensional coordinate system, actual and approximate neighbor regions

[16] M. D. Penrose. A strong law for the longest edge of the minimal spanning tree. *The Annals of Probability*, 27(1):246–269, 1999.

[17] J. M. Steele. Growth rates of euclidean minimal spanning trees with power weighted edges. *The Annals of Probability*, 16(4):1767–1787, 1988.

[18] F. D. K. Roberts. Random minimal trees. *Biometrika*, 55(1):255–258, 1968.

[19] F. Hwang D. Du. A proof of the gilbert-pollak conjecture on the steiner ratio. *Algorithmica*, 7(1):121–135, 1992.

[20] R. G. Michael and S. J. David. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, 1978.

[21] Dept. of CS at Princeton. Bossa. `http://www.cs.princeton.edu/~rwerneck/bossa/`, July 2014.

APPENDIX

A. TREE LENGTH IN THE UNIFORM CASE

When we analyze the temporary tree made up of the source and m receivers, transmission range can be ignored since it does nothing with the temporary tree construction. We establish a two-dimensional coordinate system shown in Figure 7. Let the source be the origin, and X-axis as well as Y-axis parallel to the square edge. The whole network is divided into m square cells, and the edge length of each cell is $\frac{1}{\sqrt{m}}$. In the coordinate system, the edge length is normalized to be 1, so the intersections in the network have integer coordinates. We use the coordinate of the vertex that is farthest from the origin in the square cell as the the coordinate of this cell, and hence cells also have integer coordinates.

In Figure 7, the red nodes is the source node S and the blue node is the receiver node R whose coordinate is (x, y). Without loss of generality, suppose that x and y are both positive. Set the radius to be the Euclidean distance between S and R, and draw two circles with the center in S and in R respectively. The intersected region of the unit square and two circles' overlapping area is the shaded region in Figure 7. According to criteria for neighboring receiver selection, the shaded region in figure is where the possible neighbor receivers of R are located. If no receiver lies in this shaded area, R can only connect to S directly.

Now we illustrate how the neighbor is selected by the receiver in a more detailed way. Suppose that the coordinate of the possible neighboring receiver is (a, b) (a and b don't need to be integers). And there are three conditions that should be satisfied:

$$\begin{cases} a^2 + b^2 < x^2 + y^2, \\ (x-a)^2 + (y-b)^2 < x^2 + y^2 \\ (a,b) \textit{ is within the unit square} \end{cases} \tag{17}$$

Let N_R denote all the nodes whose coordinates satisfy the conditions above, i.e., $N_R = \{(a, b) \mid$ node (a,b) might be connected to by the receiver $R\}$. For simplicity of analysis, we reduce the size of the set N_R and get a new set N'_R, an approximate neighbor region which is the rectangular region marked with parallel lines, which is shown in Figure 7.

$$N'_R = \{(a,b) | 0 \le a \le \lfloor x \rfloor, 0 \le b \le \lfloor y \rfloor\} \tag{18}$$

Now nodes in N'_R are nodes we consider possible to be chosen as neighbors by R in our analysis later. Note that in our algorithm, it is possible that some nodes in other cells with coordinates not in N'_R might also be connected to by the receiver R. With the constraints above, the estimated length of temporary tree among m receivers is larger than that of the tree built with our algorithm since we ignore some nodes that are close to receiver R. Upper bound of the temporary tree length built with our algorithm can thus be derived.

Besides shrinking the neighbor region, we also rearrange the nodes in each cell. Suppose that the coordinate of a node is (a, b), then we move it to the point with coordinate $(\lfloor a \rfloor, \lfloor b \rfloor)$. All nodes are now farther from receiver R after rearrangement. Let M_R be the set of all receivers and $p_{i,j}$ be the probability that the node in the cell whose coordinate is (i, j) is chosen by the receiver R as a neighbor to connect to. Let p_S be the probability that the receiver directly connects to the source.

W assume that the nodes in cell (i, j) is chosen if all cells (i', j') are empty, where $i \le i' \le \lfloor x \rfloor$ and $j \le j' \le \lfloor y \rfloor$ (but $i = i'$ and $j = j'$ are not satisfied at the same time). This assumption can also make the edge length estimation larger than the actual edge length. Since some nodes that are closer to the receiver might also be chosen as a neighbor receiver but they are not taken into consideration under this assumption, the estimated probability that one node is chosen is larger than actual probability.

$$\begin{aligned} p_{i,j} &\le p(\textit{all } (i',j')\textit{s are empty}) \\ &\quad - p(\textit{all } (i',j')\textit{s and } (i,j) \textit{ are empty}) \\ &\le \left(1 - \frac{(\lfloor x \rfloor + 1 - i)(\lfloor y \rfloor + 1 - j) - 1}{m}\right)^{m-1} \\ &\quad - \left(1 - \frac{(\lfloor x \rfloor + 1 - i)(\lfloor y \rfloor + 1 - j)}{m}\right)^{m-1} \end{aligned} \tag{19}$$

We define new variables a and b as: $a = \lfloor x \rfloor + 1 - i$ and $b = \lfloor y \rfloor + 1 - j$. Note that a and b are both integers. We can obtain the following expression with (19).

$$p_{i,j} = \begin{cases} c_1^{m-1} - c_2^{m-1} (0 < c_2 < c_1 < 1), & ab = \Theta(m); \\ e^{-\frac{m-1}{m}ab}(e^{\frac{m-1}{m}} - 1), & ab = o(m). \end{cases} \tag{20}$$

Because $\sqrt{m}(c_1^{m-1} - c_2^{m-2}) \to 0$, we can omit the case that

$ab = \Theta(m)$ when we calculate the length expectation.

$$p_S \leq (1 - \frac{\lfloor x \rfloor \lfloor y \rfloor}{m})^{m-1}. \tag{21}$$

The expected length of temporary tree is:

$$E(L_V) \leq \frac{1}{\sqrt{m}} \sum_{R \in M_R} E\,(\sum_{i=1}^{\lfloor x \rfloor} \sum_{j=1}^{\lfloor y \rfloor} p_{i,j} \sqrt{(x-i+1)^2 + (y-j+1)^2} + p_S \sqrt{x^2+y^2}\,) \tag{22}$$

Combine (20), (21) and (22), and we have

$$E(L_V) \leq \frac{1}{\sqrt{m}} \sum_{R \in M_R} E\,(\sum_{a=1}^{\lfloor x \rfloor} \sum_{b=1}^{\lfloor y \rfloor} e^{-\frac{m-1}{m}ab}(e^{\frac{m-1}{m}} - 1) \sqrt{(a+1)^2 + (b+1)^2} + e^{-\frac{m-1}{m}\lfloor x \rfloor \lfloor y \rfloor} \sqrt{x^2+y^2}\,) \tag{23}$$

For the first part in (22), we use the integration to evaluate its pattern.

$$\begin{aligned}
&\sum_{a=1}^{\lfloor x \rfloor} \sum_{b=1}^{\lfloor y \rfloor} e^{-\frac{m-1}{m}ab} \sqrt{(a+1)^2 + (b+1)^2} \\
&\leq \int_{(a,b) \in [1,x] \times [1,y] \setminus [1,2] \times [1,2]} e^{-\frac{m-1}{m}ab}(a+b)dadb \\
&+ \sum_{b=2}^{\lfloor y \rfloor} e^{-\frac{m-1}{m}b} \sqrt{2^2 + (b+1)^2} + \sum_{a=2}^{\lfloor x \rfloor} e^{-\frac{m-1}{m}a} \sqrt{2^2 + (a+1)^2} \\
&+ e^{-\frac{m-1}{m}} \cdot 2\sqrt{2} + e^{-4\frac{m-1}{m}} \cdot 3\sqrt{2} \qquad (24) \\
&\leq 2\eta - \eta^x - \eta^y - \frac{\eta^x}{x} - \frac{\eta^y}{y} + \frac{\eta^{xy}}{x} + \frac{\eta^{xy}}{y} \\
&+ w_1(x,y) + 2\sqrt{2}\eta + 3\sqrt{2}\eta^4 - \int_1^2 \int_1^2 e^{-\frac{m-1}{m}ab}(a+b)dadb. \qquad (25)
\end{aligned}$$

Here $\eta = e^{-1+1/m}$ in (25). Inequality (24) holds because when $a \geq 2$ and $b \geq 2$ (a, b does not equal to 2 at the same time), $\sqrt{(a+1)^2 + (b+1)^2} \leq a + b$. Then we take the components for summation at $a = 1$ and $b = 1$. We must add the value at $a = b = 2$ since (24) does not hold at this point. The substraction in (25) comes from the fact that we have already add this value at $a = b = 2$, so we must subtract the corresponding term in this part. For clarity, we replace some expressions with functions.

$$\begin{aligned}
&J(x,y) = 2\eta - \eta^x - \eta^y - \frac{\eta^x}{x} - \frac{\eta^y}{y} + \frac{\eta^{xy}}{x} + \frac{\eta^{xy}}{y}, \\
&w_1(x,y) = \sum_{b=2}^{\lfloor y \rfloor} \eta^b \sqrt{2^2 + (b+1)^2} + \sum_{a=2}^{\lfloor x \rfloor} \eta^a \sqrt{2^2 + (a+1)^2}, \\
&w_2(x,y) = \eta^{\lfloor x \rfloor \lfloor y \rfloor} \sqrt{x^2+y^2}, \\
&c_3 = 2\sqrt{2}\eta + 3\sqrt{2}\eta^4 - (2e+1)(e-1)^2/e^4.
\end{aligned}$$

The expected length of temporary tree can be expressed as:

$$\begin{aligned}
E(L_V) \leq \frac{e^{\frac{m-1}{m}} - 1}{\sqrt{m}} \sum_{R \in M_R} E(J(x,y) + w_1(x,y) + c_3) \\
+ \frac{1}{\sqrt{m}} \sum_{R \in M_R} E(w_2(x,y))
\end{aligned} \tag{26}$$

The coordinate (x,y) of a receiver is dependent on the location of source. The farther the receiver is from the source, the larger the value of $J(x,y) + w_1(x,y)$ is. We can obtain the maximum of $E(J(x,y)+w_1(x,y))$, when the source is located at one of the square's four vertices. Thus the receiver coordinate ranges from $(0,0)$ to $(\sqrt{m}, \sqrt{m})$.

For $E(w_1(x,y))$, there exist constraints for both x and y. That is, $x \geq 2$ and $y \geq 2$. So the integral of $w_1(x,y)$ is written as:

$$\frac{1}{\sqrt{m}} \sum_{R \in M_R} E(w_1(x,y)) \leq \frac{1}{\sqrt{m}} \int_2^{\sqrt{m}} \int_2^{\sqrt{m}} w_1(x,y)dxdy. \tag{27}$$

We use discrete summation to evaluate the expression on the right side of the inequality (27).

$$\begin{aligned}
&\int_2^{\sqrt{m}} \int_2^{\sqrt{m}} w_1(x,y)dxdy \leq 2\sum_{a=2}^{4} \eta^a \sqrt{(1+a)^2+4} \\
&+ \sum_{a=5}^{x} \eta^a (a + 4/3) + \sum_{b=5}^{y} \eta^b (b + 4/3)
\end{aligned} \tag{28}$$

It's not hard to check the summation on the right side of (28). We find that

$$\begin{aligned}
&\frac{1}{\sqrt{m}} \int_2^{\sqrt{m}} \int_2^{\sqrt{m}} w_1(x,y) \\
&\leq 2(\sum_{i=2}^{4} e^{-i} \sqrt{(1+i)^2+4} + \frac{19e-16}{3e^4(e-1)^2})\sqrt{m}.
\end{aligned} \tag{29}$$

Wherever the source is, the part, $\frac{1}{\sqrt{m}} \sum_{R \in M_R} E(w_2(x,y))$, can be ignored. This is because

$$\begin{aligned}
&\frac{1}{\sqrt{m}} \sum_{R \in M_R} E(w_2(x,y)) \\
&\leq \frac{1}{\sqrt{m}} \int_0^{\sqrt{m}} \int_0^{\sqrt{m}} e^{-\frac{m-1}{m}\lfloor x \rfloor \lfloor y \rfloor}(x+y)dxdy \approx \frac{2}{e\sqrt{m}} \to 0
\end{aligned} \tag{30}$$

In the end, we consider the part $E(J(x,y))$ in $E(L_V)$.

$$\begin{aligned}
&\frac{1}{\sqrt{m}} \sum_{R \in M_R} E(J(x,y)) \\
&\leq \frac{1}{\sqrt{m}} \int_0^{\sqrt{m}} \int_0^{\sqrt{m}} J(x,y)dxdy \sim \frac{2}{\sqrt{m}}(\sqrt{m}-1)^2\eta \sim \frac{2}{e}\sqrt{m}
\end{aligned} \tag{31}$$

With (29), (30) and (31), we can derive the upper bound of $E(L_V)$ by (26).

$$E(L_V) \leq 5.622\sqrt{m}. \tag{32}$$

This completes our proof.

B. TREE LENGTH IN THE GENERAL DISTRIBUTION

We partition the unit square into k square cells with edge length of $\frac{1}{\sqrt{k}}$. $m = k^{1+\gamma}$ and $0 < \gamma < 1$. In each cell, the expected number of points is approaching infinity (actually is greater than $m\epsilon_1/k$).

First, we prove an assertion for a scaled uniform distribution. With Lemma 5.1, we know that length of temporary

tree spanning m receivers is upper bounded by $c\sqrt{m}$, when nodes are uniformly distributed in a unit square. In a scaled square region with edge length of $\frac{1}{\sqrt{k}}$, the expected length of temporary tree is smaller than $\frac{c}{\sqrt{k}}\sqrt{m}$.

We first construct a tree consisting of intra-square and inter-square edges. Intra-square edge means that a receiver chooses to connect to another receiver located in the same square, while inter-square edge means that a receiver connects to another located in a different square.

Let's first consider the intra-square edges. Source is located inside one of k cells, and outside $(k-1)$ cells. For the cell containing the source, we build a temporary tree among all nodes in it. For other cells, we let each node in it connect to the closest neighbor that has shorter Euclidean distance to source and it doesn't connect to any nodes if no such neighbor exists. As is proved in Lemma 4.1 and Lemma 5.3, trees can be established within these k cells. Then we consider the inter-square edges. For any cell, there is always an adjacent cell that is closer to source.

We let a node from each cell connect to another node in adjacent cells and the inter-square edge between them has the minimal distance among all node pairs. So all squares are connected by inter-square edges. With intra-square edges and inter-square edges, all of m receivers form a tree topology. We denote this tree as T_a.

We consider the length of intra-square edges within the unit square. Suppose that Sq_i is one of k cells in the set $\{Sq_1, \cdots, Sq_k\}$ and its edge length of $\frac{1}{\sqrt{k}}$. According to Lemma 5.1 and Lemma 5.3, we can conclude that total length of intra-square edges within Sq_i is is upper bounded by $\frac{c}{\sqrt{k}}\sqrt{m\int_{Sq_i} f(\mathbf{x})d\mathbf{x}}$ with scaling method. Expected length of intra-square edges within the whole network can be obtained by the summation: $\sum_{i=1}^{k} c\frac{1}{\sqrt{k}}\sqrt{m\int_{Sq_i} f(\mathbf{x})d\mathbf{x}}$. Since the integration over the small square is smaller than $\frac{1}{k}\max_{\mathbf{x}\in Sq_i} f(\mathbf{x})$. We take the square root of this expression and by the definition of Riemann-Stiejies integration, we know that the sum is

$$\sum_{i=1}^{k} c\frac{1}{\sqrt{k}}\sqrt{m\int_{Sq_i} f(\mathbf{x})d\mathbf{x}} \le c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}. \quad (33)$$

The total length of inter-square edges within the unit square is $o(\sqrt{m})$ as we have proved in Lemma 5.3. So the expected length of T_a is upper bounded by

$$c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}.$$

Based on neighbor selection criteria in TST algorithm, each node always connects to the closest node that has shorter Euclidean distance to the source. However, for intra-square edges in T_a, the node are also required to connect to the neighbor in the same square. More limitations are imposed on neighbor selection, so the edge length between two nodes might be larger than that in temporary tree. Therefore, the length of T_a can be regarded as the upper bound of that of temporary tree.

The expected length of temporary tree is:

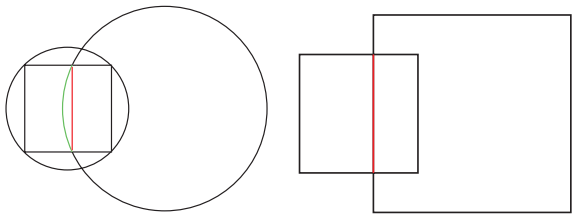
$$E(L_V) \le c\sqrt{m}\int_{\mathbf{x}\in[0,1]^2}\sqrt{f(\mathbf{x})}d\mathbf{x}. \quad (34)$$

This completes our proof.

C. MINIMUM HOP PATH

From work [16], we know that transmission range is set as $r = \Theta\left(\sqrt{\frac{\log n}{n}}\right)$, in order to ensure that network is connected. There are about $\Theta(\log n)$ nodes within transmission range of one node. Finding the path with minimal hops between two nodes can be regarded as a connectivity problem, which is a Poisson process.

Since the expression for the area where two circles intersect is hard to estimate, we estimate it in the infinite norm without affecting the results. The transmission range is set



(a) 2-norm (b) infinite-norm

Figure 8: We estimate the probability of points on the red line instead of green arc

as: $r = \sqrt{\frac{2}{\epsilon_1}}\sqrt{\frac{\log n}{n}}$. There are two receivers u and v, and the left circle in Figure 8(a) shows the transmission range of u. The circumscribed square is embedded properly in this left circle. The right circle with center in v has a radius of x. The intersected part of square and the right circle is the green arc. For simplicity, we use the red line rather than the green arc, and this will overestimate the path length. We can use ∞-norm in Figure 8(b) for estimation instead of the $2-$norm.

Let A be the event that next point exists with distance s, and B be the event that there's a point within the transmission region. It's easy to see that

$$\begin{aligned} P(A|B) &= \frac{e^{-2n\epsilon_1(r-s)r}(1-e^{-2n\epsilon_1 sr})}{1-e^{-2n\epsilon_1 r^2}} \\ &= \frac{e^{-2n\epsilon_1 r^2}}{1-e^{-2n\epsilon_1 r^2}}(e^{2n\epsilon_1 sr}-1) \end{aligned} \quad (35)$$

Denote $E_r(x)$ as the expected number of relay nodes on the path with minimal hops to a receiver with distance x. Here we consider that r in the ∞ norm are contained by $\sqrt{2}r$ in the $2-$norm, so we have the following functional equation when plugging in $r = {\epsilon_1}^{-\frac{1}{2}}\sqrt{\frac{\log n}{n}}$:

$$\begin{aligned} E_r(x) &= \int_0^r (1+E_r(x-s))\frac{2n\epsilon_1 re^{-2n\epsilon_1 r^2}}{1-e^{-2n\epsilon_1 r^2}}e^{2n\epsilon_1 sr}ds \\ &= \frac{2n\epsilon_1 r}{n^2-1}\int_0^r (1+E_r(x-s))e^{2nsr}ds \\ &= 1+\frac{2n\epsilon_1 r^2}{n^2-1}\int_0^1 E_r(x-\alpha r)e^{2n\epsilon_1 r^2\alpha}d\alpha \\ &= 1+\frac{2\log n}{n^2-1}\int_0^1 E_r(x-\alpha r)n^{2\alpha}d\alpha \end{aligned} \quad (36)$$

Clearly the lower bound of hop count is x/r. As for the upper bound of hop count, $E_r(x)$, we can also prove that

$E_r(x) \le d(x/r)+1$ for a fixed $d>1$ and a large enough n by induction. We know that $E_r(x)=1$ when $x\in(0,r)$, so it is satisfied that $E_r(x)\le d(x/r)+1$. Suppose that $E_r(x-\alpha r)\le d\frac{x-\alpha r}{r}+1$, we can show

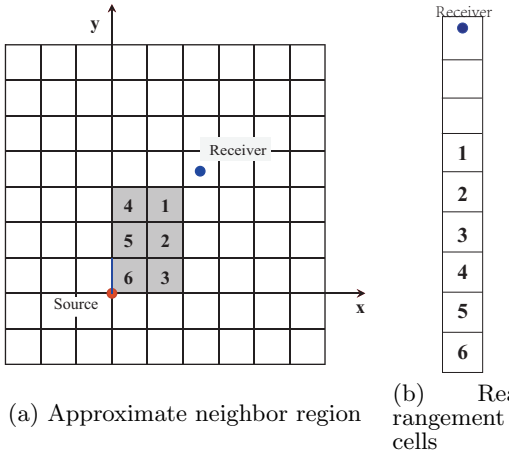
$$\begin{aligned} E_r(x) &= 1+\frac{2\log n}{n^2-1}\int_0^1 E_r(x-\alpha r)n^{2\alpha}d\alpha \\ &\le 2+\frac{2\log n}{n^2-1}\int_0^1 d\frac{x-\alpha r}{r}n^{2\alpha}d\alpha \\ &= 2+d\frac{x}{r}-d\frac{n^2}{n^2-1}+\frac{d}{2\log n} \\ &\le d(x/r)+1 \end{aligned} \tag{37}$$

Inequality (37) holds for

$$d\ge\frac{2\log n(n^2-1)}{n^2(2\log n-1)+1}. \tag{38}$$

We can conclude that $E_r(x)$ converges to x/r uniformly when n is very large. The first property has been proved. And because the transmission range is r, the total tree length converges to x. So the second property also holds.

D. MESSAGE COMPLEXITY



(a) Approximate neighbor region (b) Rearrangement of cells

Figure 9: Shrinking neighbor region and rearranging cells

We only consider the approximate neighbor region shown by the shaded area in Figure 9(a) as in length analysis. We rearrange the cells in the feasible region along with the nodes in these cells, and put them in a column. We first compare x-coordinates of the cells, and cells with larger x-coordinates will be put at the top of the column. If they have the same x-coordinates, cells with larger y-coordinates will be put at the top. Figure 9(b) illustrate this arrangement. We have $\lfloor x\rfloor\times\lfloor y\rfloor$ cells rearranged in a column. These cells are numbered from 1 to $\lfloor x\rfloor\times\lfloor y\rfloor$. After rearrangement, it is easy to see that the actual distance between nodes in the i^{th} cell and the receiver R must be smaller than $\frac{i+3}{\sqrt{m}}$.

There are five types of messages are exchanged in TST algorithm, and their quantities are denoted as Msg_i respectively ($1\le i\le 5$). In phase 1, all nodes participate in message propagation, so $E(Msg_1)=n$.

In Phase 2, three types of messages are sent during this process: *request message*, *respond message* and *connect message*. We rearrange the cells as in Figure 9(b). The i^{th} cell is represented by g_i, and the probability that one node is located in the i^{th} cell is assumed as $p(g_i)$. The probability that a node in the i^{th} cell is chosen as a neighbor by the receiver (x,y) is denoted as p_i. We have

$$\begin{aligned} p_i &= (1-\sum_{j=1}^{i-1}p(g_i))^{m-1}-(1-\sum_{j=1}^{i}p(g_i))^{m-1} \\ &= e^{-(m-1)\sum_{j=1}^{i-1}p(g_j)}-e^{-(m-1)\sum_{j=1}^{i}p(g_j)} \\ &\le e^{-\frac{m-1}{m}(i-1)\epsilon_1}-e^{-\frac{m-1}{m}i\epsilon_2} \end{aligned} \tag{39}$$

When no appropriate receivers are found in this region, receiver R connects to the source directly. Its probability is denoted as p_S.

$$p_S\le(1-\frac{\lfloor x\rfloor\lfloor y\rfloor}{m}\epsilon_1)^{m-1} \tag{40}$$

If the receiver in the i^{th} cell is selected by receiver R, k search sessions are needed in total for receiver R, where $k=\left\lceil\log_2\frac{i+3}{\sqrt{m}r}\right\rceil$.

$$\begin{aligned} E(Msg_2^R)\le &\sum_{i=1}^{\lfloor x\rfloor\lfloor y\rfloor}p_i\sum_{k=0}^{\left\lceil\log_2\frac{i+3}{\sqrt{m}r}\right\rceil}\pi(2^kr)^2\epsilon_2 n \\ &+\sum_{k=0}^{\left\lceil\log_2\frac{\sqrt{x^2+y^2}}{\sqrt{m}r}\right\rceil}p_S\pi(2^kr)^2\epsilon_2 n \end{aligned} \tag{41}$$

Combining (39), (40) and (41), we have

$$\begin{aligned} E(Msg_2^R)\le &O\left(\frac{n}{m}\sum_{i=1}^{\lfloor x\rfloor\lfloor y\rfloor}(e^{-\frac{m-1}{m}(i-1)\epsilon_1}-e^{-\frac{m-1}{m}i\epsilon_2})(i+1)^2\right) \\ &+O\left(\frac{n}{m}e^{-\frac{m-1}{m}\lfloor x\rfloor\lfloor y\rfloor\epsilon_1}(x^2+y^2)\right) \end{aligned} \tag{42}$$

It is easy to prove that

$$\sum_{i=1}^{\lfloor x\rfloor\lfloor y\rfloor}(e^{-\frac{m-1}{m}(i-1)\epsilon_1}-e^{-\frac{m-1}{m}i\epsilon_2})(i+1)^2=O(1), \tag{43}$$

$$e^{-\frac{m-1}{m}\lfloor x\rfloor\lfloor y\rfloor\epsilon_1}(x^2+y^2)=O(1). \tag{44}$$

Combining (42), (43) and (44), we can obtain

$$E(Msg_2^R)=O\left(\frac{n}{m}\right), \tag{45}$$

so the expected number of *request messages* is: $E(Msg_2)=\sum_{R\in M_R}E(Msg_2^R)=O(n)$.

Next comes *respond message*. If the neighbor search ends up in the k^{th} session, the expected number of responding receivers is at most $m\frac{\frac{3}{4}\pi(2^kr)^2\epsilon_2}{1-\frac{1}{4}\pi(2^kr)^2\epsilon_2}\le\pi(2^kr)^2\epsilon_2 m$. If the response is from the receiver in the i^{th} cell, as we have proved, there are $O(\frac{i+1}{\sqrt{m}r})$ relays on the minimal-hop path connect-

ing two receivers.

$$
\begin{aligned}
E(Msg_3^R) &\leq O\left(\sum_{i=1}^{\lfloor x\rfloor\lfloor y\rfloor} p_i \left(\pi(2^k r)^2 \epsilon_2 m\right) 2^k \Big|_{k=\left\lceil \log_2 \frac{i+3}{\sqrt{m}r}\right\rceil}\right) \\
&+ O\left(p_S \left(\pi(2^k r)^2 \epsilon_2 m\right) 2^k \Big|_{k=\left\lceil \log_2 \frac{\sqrt{x^2+y^2}}{\sqrt{m}r}\right\rceil}\right) \\
&\leq O\left(\sum_{i=1}^{\lfloor x\rfloor\lfloor y\rfloor} \left(e^{-\frac{m-1}{m}(i-1)\epsilon_1} - e^{-\frac{m-1}{m} i\epsilon_2}\right) \frac{(i+3)^3}{\sqrt{m}r}\right) \\
&+ O\left(e^{-\frac{m-1}{m}\lfloor x\rfloor\lfloor y\rfloor\epsilon_1} \frac{(x^2+y^2)^{\frac{3}{2}}}{\sqrt{m}r}\right) \\
&\leq O\left(\frac{1}{\sqrt{m}r}\right). \qquad (46)
\end{aligned}
$$

The total number of *respond messages* is:

$$E(Msg_3) = \sum_{R\in M_R} E(Msg_2^R) = O\left(\frac{\sqrt{m}}{r}\right)$$

. The last type of message in the second phase is *connect message*, it can not be larger than *respond messages*. $E(Msg_4) \leq O\left(\frac{\sqrt{m}}{r}\right)$.

In phase 3, cycle elimination, all relays participate in forwarding *eliminate message* for the worst case.

$$E(Msg_5) \leq O(n). \qquad (47)$$

The total message complexity is

$$E(Msg) = O(n). \qquad (48)$$

This completes our proof.