SREE-Tree: Self-Reorganizing Energy-Efficient Tree Topology Management in Sensor Networks

Debraj De and Sajal K. Das
Department of Computer Science
Missouri University of Science and Technology
Email: ded@mst.edu, sdas@mst.edu

***Abstract*—The evolving applications of Information and Communications Technologies (ICT), such as smart cities, often need sustainable data collection networks. We envision the deployment of heterogeneous sensor networks that will allow dynamic self-reorganization of data collection topology, thus coping with unpredictable network dynamics and node addition/ deletion for changing application needs. However, the self-reorganization must also assure network energy efficiency and load balancing, without affecting ongoing data collection. Most of the existing literature either aim at minimizing the maximum load on a sensor node (hence maximizing network lifetime), or attempt to balance the overall load distribution on the nodes. In this work we propose to design a distributed protocol for self-organizing energy-efficient tree management, called *SREE-Tree*. Based on the dynamic choice of a design parameter, the in-network self-reorganization of data collection topology can achieve higher network lifetime, yet balancing the loads. In *SREE-Tree*, starting with an arbitrary tree the nodes periodically apply localized and distributed routines to collaboratively reduce load on the multiple bottleneck nodes (that are likely to deplete energy sooner due to a large amount of carried data flow or low energy availability). The problem of constructing and maintaining optimal data collection tree (T_{opt}) topology that maximizes the network lifetime ($L(T_{opt})$) is an NP-Complete problem. We prove that a sensor network running the proposed *SREE-Tree* protocol is guaranteed to converge to a tree topology (T) with sub-optimal network lifetime ($L(T)$) such that $\frac{1}{L(T)} - \frac{1}{L(T_{opt})} < (\epsilon + Q)$, where ϵ is the flexible design parameter and Q is a factor that can depend on system configuration. With the help of experiments using standard TinyOS based sensor network simulator TOSSIM, we have validated that *SREE-Tree* achieves better performance as compared to state-of-the-art solutions, for varying network sizes.**

I. Introduction

One of the promising application domains of evolving Information and Communications Technologies (ICT) is the smart city management with sensing, computation, actuation capabilities. Sensor networks have found pervasive applications in various smart city scenarios. But such networks need to employ sustainable communication methodologies to manage various challenges such as resource heterogeneity, network dynamics. The applications may need to deploy a network of heterogeneous (in terms of sensor type, sensing rate, energy capacity, processing capability, etc.) sensor nodes for spatial-temporal multi-modal sensing data collection from the deployment areas. The dynamics in networks can continuously change, due to various factors such as periods of variation in data generation rate of a sensor node (e.g., nodes can be reconfigured remotely for data rate or different types of sensors can be triggered on a node based on varied event activity contexts), variation in energy capacity of a sensor node (e.g., nodes can harvest energy or be added with new batteries), addition/deletion of heterogeneous sensor nodes, etc. These changing dynamics will require automatic in-network self-reconfiguration of data collection topology (for example, tree topology) by the nodes themselves. This is essential in order to maintain network energy efficiency and load balancing among nodes. The data collection topology has to be seamlessly re-organized and improved (for energy efficiency and load balance) in a distributed and autonomous manner, without affecting the ongoing data collection. It should be a smooth gradual transfer of data streams from the previous to the new topology, without any abrupt pause in network collected data. Such abrupt pause may happen due to dropping some packets or buffering them. This is an important need for applications where live sensing and prompt actuation are critical. Such applications/ services will be otherwise disrupted (temporarily) every single time, during dynamic changes in the network.

Most of the existing works in the literature either attempt to minimize the maximum load on nodes for enhancing network lifetime [16], [19], or attain overall balanced load among the nodes [8]. But there is often lack of a single framework that allows the network self-adaptively (during changing network dynamics) and dynamically achieve both network energy efficiency and load balance. This motivates us to design our proposed *SREE-Tree* protocol. More precisely, *SREE-Tree* is a distributed, collaborative, self-reorganizing, energy-efficient tree maintenance protocol for sensor networks, based on a novel and efficient parent switching mechanism for load balancing in the network. The essence of network lifetime and finding (and fixing) the vulnerable bottleneck nodes are targeted in *SREE-Tree* for load balance and pushing the limit of nodes' life of operation. The characteristics of the *SREE-Tree* protocol are as follows. (a) It utilizes a design parameter ϵ to simultaneously achieve (with flexibly chosen trade-off) high network lifetime, as well as reduced and distributed load on nodes (especially on multiple high risk nodes) through in-network self-reorganization of data collection tree topology. (b) Furthermore, among existing protocols for load balanced and energy efficient dynamic tree construction (e.g., [23], [22], [7]), when a change of network status requires tree topology re-organization, most of them either require centralized re-

978-3-901882-70-8

computation or apply distributed optimization in the network. But these solutions are either expensive in terms of message communication costs, or are not dynamically adaptive to changing global optimal. This is particularly important for time varying heterogeneity in sensor networks under consideration. In contrast, the proposed *SREE-Tree* protocol exploits local beacon message payload that already exists in the ongoing routing mechanism (thus saving communication overhead), and utilizes localized communications for collaborative improvements (towards global optimal) in the network using the concept of fundamental cycles in graph theory.

System model of the research problem includes several practical factors: non-uniform and dynamic node data rates and energy capacity, separate energy costs for wireless data transmission and data reception, radio transmission power control, no data aggregation. The problem of constructing data collection tree with maximum lifetime in the system model is NP-complete ([7], [14], [15]), and so is computationally too expensive to achieve optimal solution. We have proved that our proposed *SREE-Tree* protocol achieves sub-optimal network lifetime within bound. *SREE-Tree* starts with an arbitrary tree topology, then gradually improves it by reducing the load on the *bottleneck nodes* (those that are likely to soon deplete their energy due to either large carried data flow or low remaining energy). *It applies a number of distributed and localized modules for: tree maintenance, network status update, fundamental cycle search, and two improvement rules (with locally collaborative parent node switching and transmission power control)*. These distributed and localized modules or routines are triggered or periodically performed, in order to keep the topology re-adjusted for all the dynamic changes in the network. The main contributions of this work can be summarized as follows: *(i)* Problem formulation of maximum network lifetime tree construction in sensor networks based on assumption-free general communication model (heterogeneous node data rate and energy capacity, transmission power control, energy costs for radio transmission and reception). *(ii)* Novel design of *SREE-Tree* protocol, a distributed self-reorganizing dynamic tree construction protocol for heterogeneous sensor networks. The advantageous inherent property of *SREE-Tree* is that based on flexibly chosen design parameter ϵ, it can simultaneously achieve both notions of network energy efficiency: maximum network lifetime and overall load (combination of both data load and energy capacity) reduction/ distribution among nodes (at least for the class of high risk nodes with high loads, which can be chosen flexibly). *(iii)* Proving that *SREE-Tree* achieves network lifetime $L(T)$ of a designed tree T as follows: $\frac{1}{L(T)} - \frac{1}{L(T_{opt})} < (\epsilon + Q)$, where ϵ is a flexible design parameter, Q is a factor that can depend on network configuration, and $L(T_{opt})$ is the optimal network lifetime. In addition, key analysis is done for the choice of tunable parameter ϵ. *(iv)* Presentation of experimental study in standard TinyOS based sensor network simulator TOSSIM [18], validating that *SREE-Tree* protocol provides considerably higher performance as compared to related state-of-the-art solutions (collection tree protocols based on shortest path, path quality, and node energy) in the literature, with varying network size.

The rest of the paper is organized as follows. Section II discusses the related works in the literature. Section III presents the system model and problem definition. Then section IV describes the proposed *SREE-Tree* protocol in details. Section V analyzes the performance and derives the bounded error (from optimal solution) of network lifetime achieved by *SREE-Tree*, followed by observation of some key property. Section VI evaluates performance of the *SREE-Tree* protocol with the help of experiments. Finally section VII concludes the paper with directions of future research.

II. Related Works

For continuous monitoring applications, a tree based topology construction is often used for collecting network-wide sensing data. There are a number of works investigating tree topology construction for data gathering [12], [9]. Construction of maximum lifetime data gathering tree in sensor networks is studied in [23] which assumes data aggregation/ fusion and constant transmission power. Also it is a centralized algorithm and hence not very useful for scalable sensor networks with changing network dynamics. The works in [2] and [16] have proposed heuristic algorithms for maximum lifetime data gathering tree construction. In CTP (Collection Tree Protocol) [11], each node chooses parent with maximum overall path quality to the sink node. CMAX [17] proposes an algorithm that tries to maximize the network capacity using path computation based on energy capacity of nodes. The work in [20] has designed constant ratio approximation algorithms for various broadcast and convergecast (data collection at sink node) problems.

The problem of constructing data collection tree for maximum network lifetime has been studied by a number of research works which, however, make various assumptions to simplify the problem or for matching it with very specific applications. For example a number of research works have assumed: (i) reduction or fusion of aggregated data i.e., same incoming and outgoing data flow in the node [23], [16]; (ii) constant and uniform transmission power to construct data collection tree [23]. But in majority of the real wireless sensor network applications: (i) instead of data fusion, all the data flow from each of the nodes need to be transferred to the base station (e.g., structural health monitoring, smart building, volcano monitoring network, etc.); (ii) transmission power control for varying the radio range is effective and under-utilized for optimized collection tree construction; (iii) the sensing data generation rate of each node can also be non-uniform through time periods. In this regard we have attempted to solve the problem of constructing data collection tree for maximum network lifetime for non-aggregation data flow, heterogeneous node data rates, heterogenous node energy capacity, and utilizing transmission power control.

There exist some works on probabilistic or randomized parent selection methodology for constructing better tree topol-

ogy. For example, Local-Wiser [22] proposes a distributed probabilistic load-balancing protocol. It divides the nodes in the network according to their levels or depths in the shortest path tree rooted at sink node. Then it makes switching decisions on the sensor nodes to change parent node from current parent to another node (at the same level/ depth of the tree) based on the chosen local cost function. In LOCAL-OPT [7], nodes switch between parents based on the notion of locally optimal tree. MITT [19] finds a min-max weight spanning tree through switching. The Randomized Switching for Maximizing Lifetime (or RaSMaLai) protocol [14] [15] randomly switches some sensor nodes from their original paths to other paths with lower load based on a concept of δ-bounded balanced trees. In contrast, in the *SREE-Tree* protocol the parent switching decision is not probabilistic or randomized, but based on the concept of node classifications considering data load and energy resource. Moreover, parent switching by the nodes happen in locally collaborative manner (utilizing the concept of fundamental cycles in graph theory). This locally collaborative decision making in *SREE-Tree* makes the network converge more towards global optimality, rather than locally optimal state.

In the process of constructing maximum lifetime data collection tree, the MDST (Minimum Degree Spanning Tree) have been the related sub-problems. In [10] the MDST problem was studied and the authors proposed a centralized approximation algorithm. The first approximated distributed algorithm for MDST problem was proposed in [4]. Then in [6] the same authors have proposed a self-stabilizing distributed algorithm which achieves sub-optimal degree (degree of within one from the minimum degree value).

III. System Model and Problem Definition

In this section we first describe the system model adopted, and then we formulate the problem of constructing energy-efficient parent selection and update methodology for sensor nodes.

System Model: Let G = (V, E) be the network graph consisting of a set V of nodes and the set E of edges. Here E consists of all possible edges uv from node u to v, where v is any node that can be reached from u, with all possible configurable transmission powers. Thus G is the original network with reachability among the nodes with all possible transmission power levels. The starting energy of sensor nodes can be different. The base station is assumed to be connected to an unlimited power supply (like many practical systems [13] [1]), or the base station has much higher energy capacity as compared to the other sensor nodes in the network. Now each sensor node monitors the environment and sends the data towards the base station (sink node). The time is divided into epochs, and in each epoch a node i generates and sends $u_i.B$ bits of sensor data (B is size of unit data in bits). The tree construction mechanism maintains a tree (say T) where any node i receives $\sum_{j \in C_i} f_j.B$ data flow from its children C_i. This leads to a receiving energy consumption of $\sum_{j \in C_i} f_j.B.P_r$ for node i. Node i has its own generated sensor data flow u_i. Therefore the total data flow from node i to its parent node is given by ($u_i.B$+$\sum_{j \in C_i} f_j.B$). The power consumption of node i due to transmission is then ($u_i.B$+$\sum_{j \in C_i} f_j.B$).P_t. To note that typically P_r is usually constant (only dependent on radio antenna properties of source and destination) and $P_t \propto r^2$ [3]. So P_t (of node i) = $k.r_i^2$, where r_i is the radio transmission range of that node i under the chosen transmission power level. Therefore the lifetime of node i in tree T is defined as:

$$L(T,i) = \frac{E_i}{Bkr_i^2(u_i + \sum_{j \in C_i} f_j) + BP_r \sum_{j \in C_i} f_j} \quad (1)$$

Problem definition: The goal of our tree construction protocol is to construct a tree T_{opt} from original graph G, that will provide the maximum lifetime for the network. Here network lifetime is defined as the time from the network initiation till the first node dies. This is the most commonly used and strict definition of network lifetime. Now the lifetime of any tree T is given by $L(T)$ = $\min_{i \in V} L(T,i)$. Then the goal is to maximize the lifetime of the network, i.e., maximize $\min_{i \in V} L(T,i)$. Now, the lifetime maximization problem is modified to the inverse-lifetime minimization problem as follows: minimize $R(T)$ (where $R(T) = 1/L(T)$), i.e., minimize $\max_{i \in V} 1/L(T,i)$. Therefore the maximum lifetime problem is expressed as:

$$\begin{aligned} R(T_{opt}) &= \min \max_{i \in V} R(T,i) = \min \max_{i \in V} \frac{1}{L(T,i)} \\ &= \min \max_{i \in V} \frac{Bkr_i^2(u_i + \sum_{j \in C_i} f_j) + BP_r \sum_{j \in C_i} f_j}{E_i} \end{aligned} \quad (2)$$

Problem of constructing T_{opt} is NP-Complete: As defined in Equation 2, the construction of T_{opt} aims to solve the lifetime maximization problem. Since the lifetime maximization of data collection trees is known to be NP-complete [7], calculating T_{opt} is NP-complete. Therefore it is needed to design an approximate solution, so the localized and distributed computation in sensor networks lead to sub-optimal network lifetime.

IV. SREE-Tree Protocol

In *SREE-Tree* protocol the nodes in the network start with any initial tree construction, and then with localized decision making, the overall tree topology is improved and reorganized to reduce the maximum inverse lifetime $R(T)$ as explained in Equation 2 (i.e., for increasing the minimum network lifetime). One more key strength of the *SREE-Tree* protocol is that it inherently integrates stability and support for dynamically changing nature of sensor networks such as addition or deletion of nodes, changing node data rate, changing node power levels, etc. If some changes occur to the network, the protocol automatically reconfigures the current tree to an improved tree topology. The improvements (by relieving energy bottleneck nodes through topology change) are performed using two rules/ techniques as follows: *(i)* $Rule1$:

changing the parent of some node i for the flow f_i without increasing the transmission power. This rule does not change transmission power consumption of sender node i, but reduces (resp. increases) the receiving power consumption of old (resp. new) parent; *(ii)* $Rule2$: changing the parent by increasing the transmission power of some node i for the flow f_i. This rule increases transmission power consumption of sender node i, increases receiving power consumption of the receiver (the changed parent), but reduces receiving power consumption of old parent. The core of the *SREE-Tree* protocol comprises of following modules: *(i)* $M1$: a module that maintains the tree; *(ii)* $M2$: a module that computes and shares the maximum inverse lifetime $R(T)$ in the current tree T; *(iii)* $M3$: this is the key module that computes the *fundamental cycle* in the graph with current tree topology, containing the bottleneck node(s) with $R(T)$, and then reduces $R(T)$ directly by applying $Rule1$ and indirectly by applying $Rule2$. Recall that, the *fundamental cycle* in a graph is the cycle generated by adding just one more edge to the spanning tree. In this work we have utilized the property that there is a one-to-one correspondence between fundamental cycles and edges not in the spanning tree. So there is a distinct fundamental cycle for each such edge as described later.

A. Notion of Improvement

Suppose at any moment the constructed tree in the graph G is T. An improved tree T_{next} is constructed such that $R(T_{next}) < R(T)$. To enable this gradual improvement, *two simultaneous classifications* for the nodes are used with the help of $Rule1$ and $Rule2$. In *first classification*, each node belongs to one of the three classes: CL_A, CL_B^1 or CL_C^1. In *second classification*, each node belongs to one of the three classes: CL_A, CL_B^2 or CL_C^2. This node classification approach is motivated by the work in [23]. Now, given a tree T and an arbitrary $\epsilon > 0$, let $\kappa = \lceil \frac{R(T)}{\epsilon} \rceil$, i.e. $(\kappa - 1)\epsilon < R(T) \leq \kappa\epsilon$. *Such classification with tunable parameter ϵ allows the protocol to choose the size of high risk bottleneck sensor nodes, whose load will need to be reduced below a certain level.* The classes are defined below.

❏ CL_A: node i belongs to this class if $(\kappa - 1)\epsilon < R(T, i) < \kappa\epsilon$. This means node i is a *bottleneck node*, and its transmission range or data flow carried cannot be increased. *These nodes are the target for improving the tree topology.*

❏ $CL_B^1(\mu)$: node i belongs to this class (called *blocking node*) with value μ if $(\kappa - 1)\epsilon - \frac{(\mu BP_r + \mu Bkr_i^2)}{E_i} < R(T, i) \leq (\kappa - 1)\epsilon$. *This means if node i receives additional μ units flow, then it will become a bottleneck node* and will fall in class CL_A. The value of μ for nodes is varied in module $M3$, after a *fundamental cycle* search procedure (described later).

❏ CL_C^1: node i belongs to this class if its does not belong to either of CL_A or CL_B^1.

❏ CL_B^2: node i belongs to this class (again called *blocking node*) if $(\kappa-1)\epsilon - \frac{f_i Bkr_i'^2}{E_i} < R(T, i) \leq (\kappa-1)\epsilon$, where r_i' is the increased transmission range, if i increases the transmission range to reach the next reachable neighbor (with increased transmission power level), for the total data flow it is carrying now (f_i). *This means if node i increases the transmission range further for the current flow it is forwarding, it will become a bottleneck node and will fall in class CL_A.*

❏ CL_C^2: node i belongs to this class if its does not belong to either of CL_A or CL_B^2.

The above classifications are utilized as follows. Each node periodically calculates its class (from the data flow carried and own energy capacity). Now in the current tree T, in a distributed manner in module $M3$, nodes find the *bottleneck node* (in class CL_A). Then nodes try to apply $Rule1$ and $Rule2$ to relieve the bottleneck node from class CL_A in order to reduce $R(T)$. This improvement of *bottleneck node* is done with the condition that it does not involve any node from class CL_B^1 or CL_B^2. Otherwise the node from class CL_B^1 or CL_B^2 will become bottleneck node, and the modified tree T_{next} will not be an improvement from T.

B. Modules in SREE-Tree Protocol

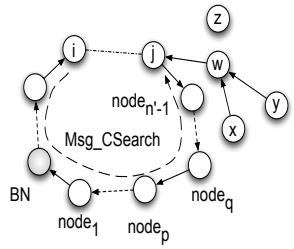
Now let us describe the modules that constitute the proposed *SREE-Tree* protocol.

1) ***Spanning Tree Module*** *($M1$):* After deployment, the network first runs this module to form an initial spanning tree such that each node considers one neighbor as parent that is reachable with minimum possible transmission power level. $M1$ uses the basic tree formation framework similar to that in the Collection Tree Protocol (CTP) [11]. Each node i maintains the variables: $root_i$ (root id known to node i), $parent_i$ (parent node of i), and hop_i (hopcount from i to root). The tree formation procedure finally converges to a state where each node has same known root id (e.g. id of base station) and a valid parent node, thus forming a valid tree topology.

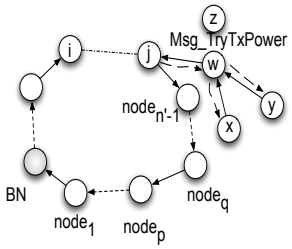
2) $R(T)$ ***Computation and Share Module*** *($M2$):* This module is responsible for periodic calculation and update of the maximum inverse lifetime $R(T)$ in the current tree topology. It could use basic Propagation of Information with Feedback (PIF) protocol [5] for maintaining updated maximum inverse lifetime $R(T)$. But for saving message complexity and network update delay, the *SREE-Tree* protocol integrate locally updated value of $R(T)$ inside the data payload of locally shared regular beacon messages (used with local broadcast to update neighbors). In this way each node i shares and updates the locally known value of $R(T)$ (variable $rmax_i$) and the node id with $rmax$ (variable $rmax\_id_i$) among themselves. In this way the global maximum value $rmax$ and the corresponding node id get disseminated in the network. Each node in the network updates local variables $rmax_i$ and $rmax\_id_i$ accordingly. During improvement in the constructed tree the value of $R(T)$ changes. So the module $M2$ assures that nodes are updated (within a delay bound) with latest value of $R(T)$.

3) ***Improvement Module*** *($M3$): This is the key module of the proposed SREE-Tree protocol.* Addition of at least one edge to a spanning tree topology will create a fundamental cycle. The *SREE-Tree* protocol exploits the property: there is a distinct fundamental cycle for each edge, thus there is

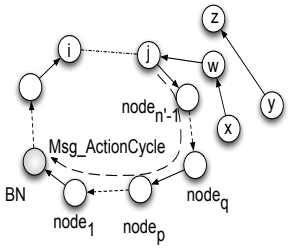
a one-to-one correspondence between the fundamental cycles and the graph edges which are not in the spanning tree. Now during tree topology formation suppose the current tree is T in the graph G. Then if some edge (i, j) (not part of the tree) is included in T, it generates a fundamental cycle. *The goal is to construct a fundamental cycle containing the bottleneck node, and then free this node with reduced data flow by removing some edge on the cycle*. This improvement module ($M3$) consists of the following four procedures: (I) *fundamental cycle generation*, (II) *action on cycle* using $Rule1$ and $Rule2$, (III) *topology change*, and (IV) *action on transmission power*.



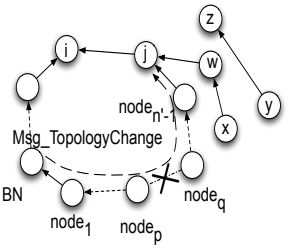
(a) **Fundamental Cycle Generation** procedure: for non-tree edge (i,j), $Msg\_CSearch$ message is generated from node i that reaches node j through tree edges.



(b) **Action on Transmission Power** procedure: *Only if* j is a blocking node, it disseminates $Msg\_TryTxPower$ to children not in a fundamental cycle. In response, node y increases transmission power to reach node z (topology change shown in next step).



(c) **Action on Cycle** procedure: node j (if or no more a blocking node) sends $Msg\_ActionCycle$ that finally reaches (containing information from all intermediate nodes) the bottleneck node BN.



(d) **Topology Change** procedure: BN analyzing $Msg\_ActionCycle$, decides to remove edge $(node_p \rightarrow node_q)$ and add edge $(j \rightarrow i)$, releasing load from BN. Then it sends $Msg\_TopologyChange$ towards node j. The edge removal and corresponding edge reversal are done by intermediate nodes, when they receive $Msg\_TopologyChange$.

Fig. 1. Examples to illustrate operational steps of Improvement Module ($M3$) in the proposed *SREE-Tree* protocol.

■ **(I) *Fundamental Cycle Generation*:** This procedure is illustrated with a working example in Figure 1(a). Now for a non-tree edge (i, j) (where $hop_i \leq hop_j$), node i calls the procedure $Cycle\_Search()$ to generate a depth-first search (DFS) in the tree T (but through shared local beacon message payload) that finds a path from node i to node j containing all edges in tree T. Using the $Cycle\_Search()$ process, a message $Msg\_CSearch$ is propagated that accumulates $path$ information with the tuple {node id (id_u), carried data flow (f_u)} of each discovered node u. If the message reaches the bottleneck node (BN), it fills up the field id_{BN}. The search generated by node i for non-tree edge (i, j) terminates, when the propagation of message $Msg\_CSearch$ reaches node j (from some node, say $node_{n'-1}$). Then node j calls procedure $Action\_Bottleneck(id_{BN}, [path], node_{n'-1}, j, [bit\_str])$ if the generated fundamental cycle contains bottleneck node (i.e., $id_{BN} \neq NULL$). Here bit_str is a binary string of n' bits, where n' is the number of hops between the bottleneck node and j. The details of $Cycle\_Search(u)$ process at any node u is described in Algorithm 1. The term $stable_u$ indicates that node u has valid parent node and knowledge of $R(T)$.

Algorithm 1: Fundamental cycle search: **Cycle_Search**(u)

if $stable_u \bigwedge (i == u)$ **then**
  $\forall j \in N_i \bigwedge (nontree_{ij} = true) \bigwedge (hop_i \leq hop_j)$, send DFS propagation of $Msg\_CSearch$ along the tree edges by sending $< Msg\_CSearch, (ij), id_{BN}, [\{i, f_i\}] >$ to $v \in N_i$ s.t. $nontree_{iv} = false$;
else
  On receiving $< Msg\_CSearch, (ij), id_{BN}, [path] >$ by node u from node $v \bigwedge stable_u$
  if $u \neq j$ **then**
    DFS propagate $< Msg\_CSearch, (ij), id_{BN}, [path \cup \{u, f_u\}] >$ on tree edges
  else
    if $nontree_{iu} = true \bigwedge id_{BN} \neq NULL$ **then**
      $Action\_Bottleneck(id_{BN}, [path], v, j, [bit\_str])$
    else
    end
  end
end

■ **(II) *Action on Cycle*:** This procedure is illustrated in Figure 1(c) (the *action on transmission power* in Figure 1(b) is explained later for relevance of explanation). After node j has discovered a fundamental cycle (by receiving $Msg\_CSearch$ generated from node i) containing the bottleneck node, it calls the process $Action\_Bottleneck(id_{BN}, [path], node_{n'-1}, j, [bit\_str])$. Recall that bit_str is initiated at node j with all 1's. This process tries to apply $Rule1$ and $Rule2$ to reduce load on the bottleneck node. Suppose the message $Msg\_CSearch$ has following path of discovered nodes $\{i, .., BN, node_1, node_2, .., node_{n'-1}, j\}$. Then node j tries to find in order if one of the edges $(j \rightarrow node_{n'-1})$, $(node_{n'-1} \rightarrow node_{n'-2})$, .., $(node_2 \rightarrow node_1)$, $(node_1 \rightarrow BN)$ can be removed from the current tree T, then add edge $(j \rightarrow i)$, and with some corresponding edge reversal. This is done by checking in that case, with added receiving and transmission cost of flow, if node j becomes a bottleneck node by being in class CL_A. If so, that edge is not removed. Otherwise the corresponding indicating bit

in bit_str is reset to 0. For example, if j finds that edge $j \rightarrow node_{n'-1}$ can be removed, bit_str is changed from $\{111...1\}$ to $\{011...1\}$ where the bit string has size n'.

There is a special case if j finds that none of the edges can be removed, because it is always becoming a bottleneck node. Then node j calls another procedure *action on transmission power* (illustrated in Figure 1(b)) to release some flow it receives from a node other than $node_{n'-1}$. This process applies $Rule2$ to node $w \in C_j - \{node_{n'-1}\}$ to let them try to change the parent (from node j) to another node by increasing the transmission power. If *action on transmission power* works fine (reducing some flow from children), then j checks again the possibility of any edge removal. In this way j checks the possibility of removal of each of the n' edges, and sends message $Msg\_ActionCycle$ to $node_{n'-1}$ with modified (if any) bit_str. $node_{n'-1}$, after receiving $Msg\_ActionCycle$ applies the same process to check for possibility of removal of edges between itself and BN. Note that nodes between j and BN does not execute process *action on transmission power*. If it finds that an edge (previously with indicating bit 0) cannot be removed (implying that node will become a bottleneck), it sets the bit back to 1. In this way $Msg\_ActionCycle$ traverses from j to BN. Finally $Msg\_ActionCycle$ reaches the bottleneck node BN. Then BN decides by looking at bit_str which edge can be removed to best relieve the node BN from being the bottleneck node.

■ **(III) *Topology Change*:** This procedure is illustrated in Figure 1(d). When the BN receives $Msg\_ActionCycle$, it first checks the bit_str string to determine which edge to remove from T to reduce its load by maximum possible amount. Next it sends $Msg\_TopologyChange$ to $node_1$, the next node on the fundamental cycle towards node j. Suppose the edge $node_q \rightarrow node_p$ is to be removed. Now, as the message $Msg\_TopologyChange$ reaches $node_q$, it changes the status of the edge from $nontree_{node_p node_q} = false$ to $nontree_{node_p node_q} = true$. Then $Msg\_TopologyChange$ is forwarded to $node_p$ and the edge status is changed again. In addition to forwarding $Msg\_TopologyChange$ further till node j, all the nodes between $node_q$ and j reverse the direction of data flow. In this way a more improved tree topology is created, where the BN is released from being a bottleneck node. But if BN finds all bits of bit_str to be 1, then it calls the process *action on transmission power* (explained next) to reduce its load.

■ **(IV) *Action on Transmission Power*:** This procedure is illustrated with an example in Figure 1(b). If node u calls the process *action on transmission power*, this process applies $Rule2$ to improve the bottleneck node directly or indirectly. More precisely, the node u sends $Msg\_TryTxPower$ to each of its children node v not on the fundamental cycle. After receiving $Msg\_TryTxPower$, each node v tries to increase its transmission power to reach the next possible reachable neighbors with its flow f_v. So v tries to change its parent from node u to another parent with increased transmission power. If successful, node v exchanges information with that node to make sure they do not become bottleneck nodes with the change of data flow. Then its old parent u is notified of the changed topology. But if v finds that it is not possible to choose another parent, it forwards the message $Msg\_TryTxPower$ to its children to do the same process. In this way $Msg\_TryTxPower$ can be propagated down the tree till the bottom or possibly up to some fixed h hops down. The parameter h can be pre-configured.

V. Theoretical Analysis

In this section we derive the absolute performance guarantee or bounded error from optimality of network lifetime achieved by the proposed *SREE-Tree* protocol.

Theorem 1. *The bounded error (from optimal solution) of network lifetime $L(T)$ of the tree T constructed by SREE-Tree is given by: $\frac{1}{L(T)} - \frac{1}{L(T_{opt})} < (\epsilon + Q)$, where ϵ is a flexible design parameter, Q is a configurable factor, and T_{opt} is the optimal tree with maximum network lifetime $L(T_{opt})$.*

Proof. Given a tree configuration T, after applying the *Improvement module ($M3$)*, let some arbitrary spanning tree X be constructed from T. Let $Y_i(T)$ denote $(Bkr_i^2(u_i + \sum_{j \in C_i} f_j) + BP_r \sum_{j \in C_i} f_j)$ for node i in tree T. Let FC denote the fundamental cycle, and let f_{FC} denote the minimum flow in FC that can be reduced from the bottleneck node BN. Thus it is possible to reduce the incoming load of f_{FC} on a set of nodes FC_R that includes $c|FC|$ nodes (including BN), where c is a fraction and $|FC|$ is the number of nodes in the fundamental cycle. Now according to the notion of improvement, for *bottleneck nodes* and *blocking nodes* in class CL_B^1, i.e., for nodes $i \in FC \cap (CL_A \cup CL_B^1(f_{FC}))$, we have $R(T,i) > (k-1)\epsilon - \frac{(f_{FC}BP_r + f_{FC}Bkr_i^2)}{E_i}$. Let FC_B denote the set $FC \cap (CL_A \cup CL_B^1(f_{FC}))$. Also $(k-1)\epsilon < R(T) \leq k\epsilon$. Therefore:

$$\begin{aligned} R(T,i) &> (k-1)\epsilon - \frac{(f_{FC}BP_r + f_{FC}Bkr_i^2)}{E_i} \\ &> R(T) - \epsilon - \frac{(f_{FC}BP_r + f_{FC}Bkr_i^2)}{E_i} \\ &\geq R(T) - \epsilon - \frac{(f_{FC}BP_r + f_{FC}Bkr_{max}^2)}{E_{min}} \end{aligned}$$

Where r_{max} is the maximum transmission range used by any node, and E_{min} is the minimum remaining node energy among nodes (when tree T is the topology in operation). Now for the modified tree X from T we obtain:

$$\begin{aligned} &R(X) \\ &\quad = \max_{i \in V} R(X,i) \\ &\quad \geq \max_{i \in FC_B} R(X,i) \\ &\quad = \max_{i \in FC_B} \frac{Y_i(X)}{E_i} \\ &\quad \geq \frac{\sum_{i \in FC_B} Y_i(X)}{\sum_{i \in FC_B} E_i} \\ &\geq \frac{(\sum_{i \in FC_B} Y_i(T)) - (c|FC|BP_r f_{FC}) - \sum_{i \in FC_R} f_{FC}Bkr_i^2}{\sum_{i \in FC_B} E_i} \end{aligned}$$

$$
\begin{aligned}
&> \frac{(\sum_{i\in FC_B} Y_i(T)) - c|FC|(BP_r f_{FC} + f_{FC} B k r_{max}^2)}{\sum_{i\in FC_B} E_i} \\
&\geq \min_{i\in FC_B} R(T,i) - \frac{c|FC|(BP_r f_{FC} + f_{FC} B k r_{max}^2)}{\sum_{i\in FC_B} E_i} \\
&\geq R(T) - \epsilon - \frac{(f_{FC} BP_r + f_{FC} B k r_{max}^2)}{E_{min}} \\
&\quad - \frac{c|FC|(BP_r f_{FC} + f_{FC} B k r_{max}^2)}{\sum_{i\in FC_B} E_i} \\
&\geq R(T) - \epsilon - \frac{(f_{FC} BP_r + f_{FC} B k r_{max}^2)}{E_{min}} \\
&\quad - \frac{c|FC|(BP_r f_{FC} + f_{FC} B k r_{max}^2)}{E_{min}} \\
&= R(T) - \epsilon - \frac{(1 + c|FC|)(f_{FC} BP_r + f_{FC} B k r_{max}^2)}{E_{min}} \\
&\geq R(T) - \epsilon - \frac{n_{FC}^{max}(f_{FC} BP_r + f_{FC} B k r_{max}^2)}{E_{min}} \\
&\geq R(T) - \epsilon - \frac{n_{FC}^{max}(u_{max} BP_r + u_{max} B k r_{max}^2)}{E_{min}}
\end{aligned}
$$

Where n_{FC}^{max} is the maximum size (number of nodes) of the fundamental cycles searched, and u_{max} is maximum of the data generation rates of the nodes. X is an arbitrary tree that is improved from T. Now the optimal tree T_{opt} with maximum lifetime will be the improved tree X or even better than it. Therefore, $R(T_{opt}) > R(T) - \epsilon - \frac{n_{FC}^{max}(u_{max} BP_r + u_{max} B k r_{max}^2)}{E_{min}}$. So, $L(T) > \frac{1}{R(T_{opt}) + \epsilon + \frac{n_{FC}^{max}(u_{max} BP_r + u_{max} B k r_{max}^2)}{E_{min}}} = \frac{1}{\frac{1}{L(T_{opt})} + \epsilon + Q}$ where $Q = \frac{n_{FC}^{max}(u_{max} BP_r + u_{max} B k r_{max}^2)}{E_{min}}$. Q can be a configurable factor by allowing certain limits for terms like n_{FC}^{max} and others. The fundamental cycles will then be checked accordingly to decide whether to operate on it or not. Hence, $\frac{1}{L(T)} - \frac{1}{L(T_{opt})} < (\epsilon + Q)$ which provides the absolute performance guarantee or bounded error of *SREE-Tree* from optimality for the inverse network lifetime measure, and therefore for the optimal network lifetime. □

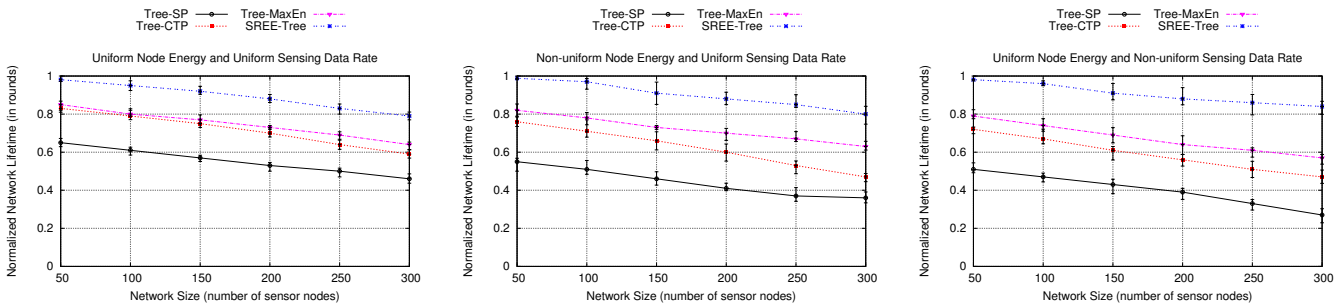
Corollary 1.1. *The parameter ϵ presents a design choice: lower ϵ means lower bounded error ($\epsilon + Q$) from network lifetime optimality but also lower number of nodes in class CL_A (target bottleneck nodes for load reduction). On the other hand, higher ϵ means more nodes in class CL_A (for load reduction) but larger bounded error from network lifetime optimality.*

VI. Experimental Results

We have evaluated the performance of our *SREE-Tree* protocol using the standard TinyOS based sensor network simulator TOSSIM [18]. Since the protocol implementations are done in TinyOS, it is easily portable to a real sensor network deployment. The *SREE-Tree* protocol is compared with following state-of-the-art protocols in the literature: *Tree-SP*, *Tree-CTP* and *Tree-MaxEn*. The *Tree-SP* is the tree based data collection topology where each node chooses parent with shortest hopcount to the sink node (base station). The *Tree-CTP* is the tree data collection topology used in the Collection Tree Protocol [11]. In CTP based tree formation, each node chooses parent with maximum overall path quality to the sink node. Based on *CMAX* [17], *Tree-MaxEn* is the tree based data collection topology where each node chooses a parent that is a neighbor node with maximum remaining energy and fewer hopcount to the sink node.

Simulation Setup: The network lifetime (in rounds) is evaluated in three scenarios. In *setup I* the nodes have uniform starting energy capacity and uniform sensing data generation rate. In *setup II* the nodes have non-uniform starting energy capacity but uniform sensing data generation rate. Finally, in *setup III* the nodes have uniform starting energy capacity but non-uniform sensing data generation rates. Each of these network scenarios are used with network sizes of 50, 100, 150, 200, 250 and 300. For each network size, the nodes are deployed randomly in the area so that the minimum nodes separation is at least 5 meters. The sink node is at one corner of the deployed area. Each experimental setup combination is run for 10 times for achieving the confidence interval of the target performance parameter. For calculation of energy consumption we have used standard Tmote Sky (TelosB [21]) sensor node parameters (such as 250 Kbps data rate, 19.5 mA current consumption in radio transmission mode when using default radio power level, 21.8 mA current consumption in radio receive mode), and data packet size of 48 bytes is used. Periodically broadcasted local beacon message among the nodes are utilized to piggyback the control overhead. In uniform starting energy setup, nodes are assumed to start with standard 2200 mAh energy capacity. In non-uniform starting energy setup, nodes are assumed to start with energy capacity varied in the range (1100 mAh, 2200 mAh). In uniform data rate setup, nodes are assumed to generate sensing data at 5 packets per second. In non-uniform data rate setup, nodes are assumed to generate sensing data at a rate in the range (1 packets per second, 10 packets per second). The term ϵ is selected as very small fraction of inverse lifetime $R(T)$ in the network, so that $\kappa = \lceil \frac{R(T)}{\epsilon} \rceil$ is 10 times the network size for instance. However, the choice of ϵ is subject to further study and analysis. For the performance parameter, we have calculated the network lifetime in terms of rounds. For understanding of relative performance, the normalized network lifetime is illustrated in the figures of experimental results.

Experimental Results: The results of comparative performance evaluation are presented in Figures 2(a), 2(b) and 2(c). These figures present the normalized network lifetime with different network sizes. It can be observed that our proposed *SREE-Tree* protocol achieves much higher network lifetime compared to *Tree-SP*, *Tree-CTP* and *Tree-MaxEn*, especially when nodes have non-uniform starting energy and non-uniform sensing data generation rates. The performance of *Tree-SP* is the least because it is not aware of node energy capacity and node data load. *Tree-MaxEn* performs slightly better than *Tree-CTP* in the used network scenarios due to local



(a) setup I: Uniform Node Energy and Uniform Sensing Data Rate

(b) setup II: Non-uniform Node Energy and Uniform Sensing Data Rate

(c) setup III: Uniform Node Energy and Non-uniform Sensing Data Rate

Fig. 2. Network Lifetime performance for different network size.

energy awareness in choosing parent node. But the *SREE-Tree* protocol achieves best performance because of its awareness of load (both in terms of energy capacity and data load) balancing of nodes not in the local neighborhood, but in context of the whole network. This is achieved through the use of bottleneck node search (followed by topology re-adjustments) on the fundamental cycles in the graph.

VII. Conclusion

In this work we have designed *SREE-Tree*, a novel distributed dynamic tree management protocol for sensor networks. It enables sustainable and energy efficient topology for prolonged data collection in self-reorganizing sensor networks. *SREE-Tree* can simultaneously achieve high network lifetime and reduced load on the numbers of high risk bottleneck nodes. Our future work includes the design of multi-sink multi-tree based topology management protocol for large-scale sensor networks. The intra-tree and inter-tree switching of nodes (for parent selection) will be designed based on the *SREE-Tree* protocol methodologies.

References

[1] S. W. Arms, J. H. Galbreath, A. T. Newhard, and C. P. Townsend. Remotely Reprogrammable Sensors for Structural Health Monitoring. In *Structural Materials Technology (SMT): NDE/NDT for Highways and Bridges*, pages 331–338, 2004.

[2] G. S. Badrinath, P. Gupta, and S. K. Das. Maximum Lifetime Tree Construction for Wireless Sensor Networks. In *Proceedings of the 4th international conference on Distributed computing and internet technology (ICDCIT'07)*, pages 158–165, 2007.

[3] C. A. Balanis. *Antenna Theory: Analysis and Design*. John Wiley and Sons Inc, 1997.

[4] L. Blin and F. Butelle. The First Approximated Distributed Algorithm for the Minimum Degree Spanning Tree Problem on General Graphs. In *17th International Parallel and Distributed Processing Symposium (IEEE IPDPS'03)*, Apr. 2003.

[5] L. Blin, A. Cournier, and V. Villain. An improved snap-stabilizing pif algorithm. In *Proceedings of the 6th international conference on Self-stabilizing systems (SSS'03)*, pages 199–214.

[6] L. Blin, M. G. Potop-Butucaru, and S. Rovedakis. Self-stabilizing minimum-degree spanning tree within one from the optimal degree. In *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS'09)*, pages 1–11, May 2009.

[7] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'05)*, pages 1747–1757 vol. 3, 2005.

[8] D. De, W.-Z. Song, S. Tang, and D. Cook. EAR: An Energy and Activity Aware Routing Protocol for Wireless Sensor Networks in Smart Environments. *The Computer Journal*, 55(12):492–1506, 2012.

[9] M. Enachescu, A. Goel, R. Govindan, and R. Motwani. Scale-free aggregation in sensor networks. *Theoretical Computer Science*, 344(1):15–29, Nov. 2005.

[10] M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17:409–423, 1994.

[11] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (ACM SenSys'09)*, pages 1–14, 2009.

[12] A. Goel and D. Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *In Proceedings of the 14th Symposium on Discrete Algorithms (ACM SODA'03)*, pages 499–505, 2003.

[13] G. Hackmann, C.-L. Fok, G.-C. Roman, and C. Lu. Agimone: Middleware Support for Seamless Integration of Sensor and IP Networks. In *DCOSS'06*, 2006.

[14] S. K. A. Imon, A. Khan, M. Di Francesco, and S. K. Das. Rasmalai: A randomized switching algorithm for maximizing lifetime in tree-based wireless sensor networks. In *IEEE INFOCOM'13*, pages 2913–2921, April 2013.

[15] S. K. A. Imon, A. Khan, M. Di Francesco, and S. K. Das. Energy-efficient randomized switching for maximizing lifetime in tree-based wireless sensor networks. *IEEE/ACM Transactions on Networking*, 2014.

[16] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, Aug. 2003.

[17] K. Kar, M. Kodialam, T. V. Lakshman, L. Tassiulas, and R. Tassiulas. In *22nd Annual Joint Conference of the IEEE Computer and Communications (IEEE INFOCOM'03)*, pages 673–681.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems (ACM SenSys'03)*, pages 126–137, 2003.

[19] J. Liang, J. Wang, J. Cao, J. Chen, and M. Lu. An efficient algorithm for constructing maximum lifetime tree for data gathering without aggregation in wireless sensor networks. In *IEEE INFOCOM'10*, pages 1–5, Mar 2010.

[20] Z. Nutov and M. Segal. Improved approximation algorithms for maximum lifetime problems in wireless networks. *Theoretical Computer Science*, 453(0):88–97, 2012.

[21] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *IPSN'05*, Aug. 2005.

[22] Y. Wang, Y. Wang, H. Tan, and F. C. M. Lau. Maximizing network lifetime online by localized probabilistic load balancing. In *ADHOC-NOW'11*, pages 332–345, 2011.

[23] Y. Wu, S. Fahmy, and N. B. Shroff. On the Construction of a Maximum-Lifetime Data Gathering Tree in Sensor Networks: NP-Completeness and Approximation Algorithm. In *27th Conference on Computer Communications (IEEE INFOCOM'08)*, Apr. 2008.