

# Algorithms for Local Sensor Synchronization

Lixing Wang, Yin Yang, Xin Miao, Dimitris Papadias, Yunhao Liu

Department of Computer Science and Engineering, Hong Kong University of Science and Technology

Clear Water Bay, Hong Kong

{lxwang, yini, miao, dimitris, liu}@cse.ust.hk

**Abstract—** In a wireless sensor network (WSN), each sensor monitors environmental parameters, and reports its readings to a base station, possibly through other nodes. A sensor works in cycles, in each of which it stays active for a fixed duration, and then sleeps until the next cycle. The frequency of such cycles determines the portion of time that a sensor is active, and is the dominant factor on its battery life. The majority of existing work assumes *globally synchronized WSN* where all sensors have the same frequency. This leads to waste of battery power for applications that entail different accuracy of measurements, or environments where sensor readings have large variability.

To overcome this problem, we propose LS, a query processing framework for *locally synchronized WSN*. We consider that each sensor  $n_i$  has a distinct *sampling frequency*  $f_i$ , which is determined by the application or environment requirements. The complication of LS is that  $n_i$  has to wake up with a *network frequency*  $F_i \geq f_i$ , in order to forward messages of other sensors. Our goal is to minimize the sum of  $F_i$  without delaying packet transmissions. Specifically, given a routing tree, we first present a dynamic programming algorithm that computes the optimal network frequency of each sensor; then, we develop a heuristic for finding the best tree topology, if this is not fixed in advance.

## I. INTRODUCTION

A sensor is a device with a radio and limited computation capabilities, which takes physical measurements such as temperature, light and humidity. A wireless sensor network (WSN) consists of a *base station* that collects measurements and/or aggregate information, and a set of sensor nodes, each of which is able to directly communicate with other sensors (or the base station) within the area of its radio coverage. Users register continuous queries at the base station, which are processed based on the sensor measurements. The transmission of sensor readings to the base station are performed according to the *topology* of the WSN, which can be either tree-based (e.g., [12]) or multi-path (e.g., [17]). This work follows the tree-based paradigm due to its simplicity and high energy efficiency.

In most applications, especially WSN deployed in harsh or difficult to access environments, sensor battery power is the main bottleneck of the entire system. In order to minimize energy consumption, instead of sensors being on continuously, they operate in *cycles*. Specifically, within a cycle, each sensor is active for a fixed duration, during which it (i) collects measurements from the environment, (ii) receives data from other sensors in its network neighborhood, (iii) possibly performs computations<sup>1</sup> on the received and collected

measurements, (iv) broadcasts data to the WSN, and then enters the *sleep* mode until it *wakes* up for the next cycle.

The power consumption of a sensor is dominated by the time that its radio remains on [9][18][21]. Accordingly, since a sensor switches on its radio for a fixed duration in every cycle, its energy cost is proportional to the *frequency* of the cycles. Existing work has focused on query processing in globally synchronized WSN, where all sensors have the same frequency. This leads to waste of battery power for applications that entail different accuracy of measurements (depending on the sensor location), or environments where sensor readings have large variability. To overcome this problem, we propose LS (for *local synchronization*), a novel framework that permits sensors to have different frequencies. For instance, in a factory setting, temperature sensors in engine rooms, or other areas susceptible to fire, must sample much more frequently than sensors in warehouses. In other cases, the different frequencies may be imposed by the variability of measurements, e.g., a sensor that has stable readings for long periods should turn on less often than another whose readings change with a high rate.

Let the *sampling frequency*  $f_i$  of sensor node  $n_i$  be the minimum frequency by which  $n_i$  should turn on in order to take measurements. Given  $f_i$ , the goal of LS is to determine a *network frequency*  $F_i$  of each sensor  $n_i$  so that: (i)  $F_i$  is at least as high as  $f_i$  in order to satisfy the accuracy requirements, (ii)  $n_i$  is active when it needs to forward messages from other sensors to the base station, and (iii) the sum of network frequencies for all sensors in the WSN is minimized. Fig. 1a shows an example subtree of a WSN containing three sensors  $n_1$ - $n_3$  and their sampling frequencies  $f_1$ - $f_3$ . Sensor  $n_1$  is in the path of  $n_2$  and  $n_3$ ; i.e., it must forward messages from these sensors to the base station, implying that  $n_1$  should be active whenever  $n_2$  and  $n_3$  are active. Fig. 1b and Fig. 1c illustrate two solutions towards this. In Fig. 1b, the network frequency  $F_2$  of  $n_2$  increases with respect to its sampling frequency ( $\Delta_2 = F_2 - f_2 = 1$ ), achieving synchronization of  $n_1$  and  $n_2$  ( $n_1$  and  $n_3$  are already synchronized). Consequently,  $n_2$  can sample with higher frequency (3 instead of 2) since its energy consumption is dominated by the radio. In Fig. 1c, the network frequencies  $F_1$  and  $F_3$  become 4, whereas  $F_2 = f_2 = 2$  (i.e.,  $n_2$  wakes up once for every two cycles of  $n_1$  and  $n_3$ ). The first solution is better because the total increase ( $\Sigma \Delta_i$ ) of the network frequency compared to the sampling frequency is

<sup>1</sup> In case of *in-network* query processing, sensor nodes perform local computations instead of simply forwarding raw data; e.g., for a

*max* query, a sensor computes and transmits only the maximum among the received values and the local measurement (instead of all values).

smaller (1 versus 2). Since the sampling frequencies are fixed and given, minimizing  $\Sigma\Delta_i$  is equivalent to minimizing the total network frequencies.

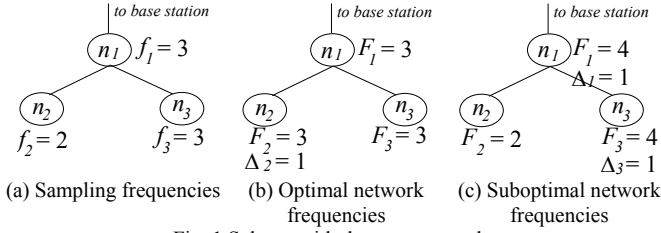


Fig. 1 Subtree with three sensor nodes

Searching for the optimal network frequencies in a WSN is a challenging task, for two reasons. First, there is an infinite search space for the network frequency  $F_i$  of a node  $n_i$ , which can be any real number satisfying  $F_i \geq f_i$  so that  $n_i$  is synchronized with the sensors directly connected to it. Second, locally optimal solutions do not necessarily lead to globally optimal ones. Fig. 2a extends the subtree of Fig. 1a by adding two more sensors  $n_4$  and  $n_5$ , with  $f_4=2$  and  $f_5=4$ . Fig. 2b expands the solution of Fig. 1b to this setting. Given that  $F_1=3$ , the best values for the network frequencies of  $n_4$  and  $n_5$  are  $F_4=F_5=6$  (i.e., a multiple of  $F_1=3$ ), yielding  $\Sigma\Delta_i=7$  ( $\Delta_2+\Delta_4+\Delta_5$ ). On the other hand, by expanding the solution of Fig. 1c, we obtain the network frequencies of Fig. 2c, which are minimal ( $\Sigma\Delta_i=4$ ) for this setting.

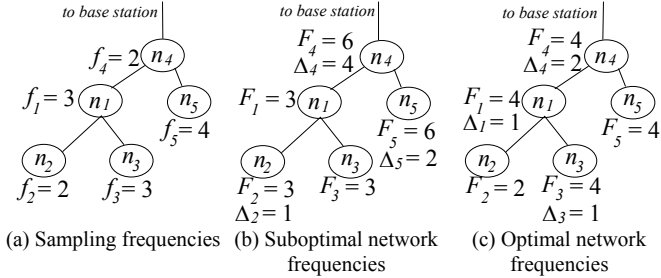


Fig. 2 Subtree with five sensor nodes

The problem is more complex when the tree topology is not given, but it has to be computed so that it minimizes  $\Sigma\Delta_i$ . To our knowledge, the only previous work [11] on locally synchronized WSN requires proxies. On the other hand, LS achieves energy-efficient local synchronization without additional hardware. Specifically, (i) given a tree topology, we compute the optimal network frequencies through a polynomial-time dynamic programming algorithm; (ii) we employ effective heuristics to obtain a high-quality tree topology, if this is not fixed in advance. Extensive experiments demonstrate that LS leads to substantial energy savings compared to the globally synchronized framework.

The rest of the paper is organized as follows. Section II surveys related work. Section III formally defines two key tasks in local synchronization: the computation of the optimal network frequencies and the tree topology. Section IV and V provides efficient algorithms for the above tasks, respectively. Section VI examines the effectiveness of the proposed algorithms using real and synthetic WSN data. Section VII concludes the paper.

## II. RELATED WORK

Section II-A overviews data transmission in WSN. Section II-B surveys WSN query processing methods that minimize energy consumption.

### A. Data Transmission in a WSN

Due to their limited radio range, most sensors in the WSN cannot transmit directly to the base station, but must communicate with the latter through other sensors. These connections are represented by a connectivity graph  $G$  where: (i) every node  $n_i \in G$  corresponds to a sensor (we use the terms *sensor* and *node* interchangeably), and (ii) an edge  $(n_i, n_j) \in G$  denotes that sensors  $n_i, n_j$  are within the radio range of each other<sup>2</sup>. Fig. 3a shows a graph consisting of five sensors  $n_1$  to  $n_5$ , and a base station  $n_0$ ;  $n_1$  and  $n_2$  can communicate directly with the base station  $n_0$ . On the other hand,  $n_3, n_4$  and  $n_5$  must relay their messages through  $n_1$  and  $n_2$ . Since there are multiple choices for  $n_3$ - $n_5$  to reach the base station, the WSN must select a *routing scheme* that determines the routing path for each message.

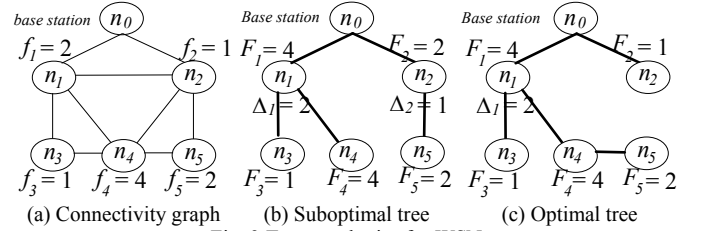


Fig. 3 Tree topologies for WSN

A large class of routing schemes organizes sensors in tree topologies [13]. Specifically, a spanning tree is built from the connectivity graph, with the base station acting as the root. Most approaches only consider *min-hop* trees, in order to minimize packet losses [8]. A spanning tree  $T \subseteq G$  is *min-hop*, if and only if the number of edges between every node  $n_i$  and  $n_0$  is the minimum among all possible paths in  $G$ . For example, the tree in Fig. 3b is *min-hop*, whereas that in Fig. 3c, is not because  $n_5$  needs 3 hops to reach  $n_0$  and there is a shorter path  $n_5-n_2-n_0$  in  $G$ . In multi-path routing schemes [2][17], a message may arrive at the base station through several paths. In the example of Fig. 3a, a data packet from  $n_4$  may be transmitted to both  $n_1$  and  $n_2$ . Compared with routing trees, multi-path schemes are more robust, at the cost of higher energy consumption and duplicate packets received by the base station. Manjhi et al. [15] propose *Tributaries-and-Deltas*, combining tree and multi-path routing.

In our work we consider tree topologies. However, in our setting, conventional minimum spanning tree algorithms (i.e., Prim's and Kruskal's) do not necessarily minimize the network frequency. Fig. 3b illustrates a counter-example using Prim's algorithm that starts with a single node and keeps adding the edge with the minimum cost, which is defined as the  $\Delta_i$  incurred in the path from the new node to the sink. We begin with the base station  $n_0$  and insert all edges that connect

<sup>2</sup> We assume that all edges in  $G$  are undirected, which holds in most practical WSNs [14].

it with nodes ( $n_1, n_2$ ) within its range. Assuming that  $n_0$  is always on, these edges have zero cost. Then,  $n_3$  is appended as a child of  $n_1$  also with zero cost. Subsequently, the algorithm adds node  $n_5$ , which increases the frequency of  $n_2$  to 2 ( $\Delta_2 = 1$ ), and finally  $n_4$  with cost  $\Delta_1 = 2$ . The overhead of this tree measured by  $\Sigma \Delta_i$  is 3. On the other hand, the optimal spanning tree in Fig. 3c has cost 2. The complication is due to the fact that edge weights are not constant, but keep changing according to the nodes already inserted in the tree.

### B. Query Processing in WSN

WSN query processing techniques can be classified into three categories, depending on their restrictions on the sensors' sampling and network frequencies. Methods (e.g., [3], [19], [20], [22]) in the first category assume that all sensors have the same sampling frequency  $f$ . The second class consists of techniques (e.g., [12], [13]) that allow different sensors to have different sampling rates, but require that they use the same network frequency  $F = \max\{f_i \mid \forall n_i\}$ . Finally, the methods of [11] provide flexibility on frequency assignment using additional base-station-like devices called *proxies* that form the backbone of the WSN. Specifically, (i) each sensor is assigned to one proxy; (ii) all the sensors assigned to the same proxy have the same frequency; and (iii) sensors assigned to different proxies may have different frequencies. The use of proxies, however, limits the applicability of [11], since they increase the financial costs of the WSN system, as well as the difficulty for its deployment.

Besides the above, the network community has proposed Low Power Listening (LPL) [7], a low-level protocol for *unsynchronized* WSNs that allows sensors to have arbitrary working cycles. Unlike synchronized WSNs, in LPL a sensor  $n_i$  may be asleep when another node  $n_j$  needs it to forward a message. When this happens,  $n_j$  postpones the message, and keeps probing  $n_i$  by continuously sending *preamble signals*, until  $n_i$  wakes up and responds. The two sensors then perform the delayed data transmissions. Due to such delays, LPL may not be suitable for applications with strict responsiveness requirements. Furthermore, the sender of the preamble signals (i.e.,  $n_j$ ) must stay active for long periods, which drains up its battery power. The above problems amplify as the height of the routing tree increases. In contrast, the proposed methods do not incur any result delays or prolonged transmissions of control signals.

### III. PROBLEM DEFINITION

Let  $f_i$  be the *sampling frequency* of sensor node  $n_i$ ;  $f_i$  is a real number determined by the application requirements on accuracy, or the variability of measurements. The proposed LS (*local synchronization*) assigns to each  $n_i$  an individual network frequency  $F_i$ , rather than a global one. In practice, since the work cycle of a sensor must be sufficiently long to complete the necessary sensing, computation and communication tasks, its network frequency must not exceed a constant  $F_{max}$ . Let  $N$  be the total number of sensors excluding the base station. Our goal is to minimize the objective function  $\Phi = \sum_{i=1}^N F_i$ , referred to as the *cost* of the

WSN. This cost determines the total time that sensors are active, and therefore, it dominates the overall energy consumption. The proposed methods can be easily extended to other aggregates such as maximum and weighted sum. We assume that the WSN uses a tree topology  $T$ , which guides the routing of messages between sensors. Table I summarizes frequent symbols.

TABLE I  
SUMMARY OF NOTATIONS

| Symbol         | Meaning                                                                 |
|----------------|-------------------------------------------------------------------------|
| $G, T$         | WSN connectivity graph and routing tree                                 |
| $N$            | Number of sensors in the WSN                                            |
| $\Phi$         | Cost of the WSN                                                         |
| $n_i$          | $i$ -th sensor (if $1 \leq i \leq N$ ), or the base station (if $i=0$ ) |
| $f_i, F_i$     | Sampling and network frequency of $n_i$                                 |
| $F_i^{LB}$     | Lower bound of $F_i$ defined in Lemma 2                                 |
| $F_{max}$      | Maximum possible network frequency                                      |
| $T_i$          | Subtree of $T$ rooted at $n_i$                                          |
| $\Phi_i^{F_i}$ | Minimum cost of subtree $T_i$ for a given $F_i$                         |

Each sensor must be able to communicate with the base station, in order to report its measurements and / or in-network computation results. To model this, we first introduce the concept of local synchronization between two directly connected sensors.

**Definition 1 (Local Synchronization):** Given a routing tree  $T$ , an internal node  $n_i$  and its child  $n_j$ ,  $n_i$  and  $n_j$  are locally synchronized (or simply *synchronized* when the context is clear), if and only if (i)  $n_i$  and  $n_j$  start one of their respective cycles at the same time, and (ii) there exists a positive integer  $k$  such that  $F_i = k \cdot F_j$ .

Intuitively, when a node  $n_i$  is synchronized with its child  $n_j$ ,  $n_i$  is able to process and / or forward every message from  $n_j$  on time. Among the two conditions in the Definition 1, (i) is a low-level networking issue, e.g., the clocks of the two sensors need to be synchronized<sup>3</sup>, whereas (ii) restricts the sensors' network frequencies in LS query processing. Hence, in the following we assume that (i) is always satisfied. We next model our main constraint.

**Lemma 1:** Every sensor is able to communicate with the base station  $n_0$  in each cycle, if and only if every pair of parent-child nodes in  $T$  are locally synchronized.

Based on whether  $T$  is given, we distinguish two different versions of the problem. The first, referred to as Problem 1, takes  $T$  as an input (e.g., determined by an existing algorithm based on link quality considerations [6]), and computes the network frequencies of the sensors that minimize  $\Phi$ . In the second version (Problem 2),  $T$  is unknown; instead, given the WSN connectivity graph  $G$ , LS selects both the optimal topology  $T \subseteq G$  and the network frequencies to minimize  $\Phi$ . Compared with Problem 1, the additional flexibility on  $T$  may

<sup>3</sup> Small differences between  $n_i$ 's and  $n_j$ 's clocks are tolerable, as long as the two sensors can finish message transmissions in one common working cycle. Furthermore, LS requires only local clock synchronization (e.g., [4], [5]), which is less demanding than global clock synchronization of all sensors in the WSN (e.g., [10]).

lead to further reduction of  $\Phi$ , at the expense of an enlarged search space. Formally, we state the above two problems as follows.

**Problem 1:** Given a routing tree  $T$  and the sampling frequency  $f_i$  of each node  $n_i$ , compute the network frequency  $F_i$  of each  $n_i$  so that (i)  $F_{\max} \geq F_i \geq f_i$ , (ii) every pair of parent-child sensors in  $T$  are locally synchronized and (iii)  $\Phi = \sum_{i=1}^N F_i$  is minimized.

**Problem 2:** Given the connectivity graph  $G$  and the sampling frequency  $f_i$  of each node  $n_i$ , find the best routing tree  $T \subseteq G$  that minimizes  $\Phi = \sum_{i=1}^N F_i$ , where the network frequency  $F_i$  of each sensor  $n_i$  is computed according to Problem 1.

Unfortunately, we have the following negative result.

**Theorem 1:** Problem 2 is NP-hard.

**Proof:** (by reduction to set cover) Given an instance of the set cover problem that involves a set  $E$  of elements and another  $S$  of subsets, construct a corresponding connectivity graph  $G$  as follows.  $G$  contains the base station  $n_0$ , as well as two layers of sensors. For each subset  $s \in S$  (resp. element  $e \in E$ ) in the set cover instance, we create a first-layer (resp. second-layer) node in  $G$ . Furthermore, whenever a subset  $s$  contains an element  $e$  in the set cover instance, we add an edge connecting  $n_s$  and  $n_e$  in  $G$ , where  $n_s$  and  $n_e$  are the corresponding nodes of  $s$  and  $e$ , respectively. Additionally, there is an edge in  $G$  between each node on the first layer and the base station  $n_0$ . The sampling frequency of every first-layer (resp. second-layer) node is 1 (resp. 2). Then, we solve Problem 2 with the above  $G$  to obtain the optimal routing tree  $T$ . The set of first-layer nodes in  $T$  that are parent to at least one second-layer node correspond to the optimal solution of the set cover instance. Therefore, Problem 2 is at least as hard as set cover. Moreover, first-layer (resp. second-layer) nodes always require 1 (resp. 2) hops to reach the base station in any spanning tree of  $G$ . Therefore, the routing tree  $T \subseteq G$  is always min-hop, meaning that Problem 2 is still as hard as set cover under the min-hop limitation.  $\square$

In the following, we describe solutions to both problems. In addition, we discuss their extensions to the case when the routing tree (in Problem 1) or the connectivity graph (in Problem 2) changes over time.

#### IV. FINDING NETWORK FREQUENCIES

This section focuses on Problem 1, i.e., optimal network frequency computation with a given routing tree  $T$ . Section IV-A lays down the theoretical foundation of the proposed solution. Section IV-B describes an efficient dynamic programming algorithm, and analyses its space and time complexity. Section IV-C provides further optimizations.

##### A. Theoretical Foundation

We first investigate the range of feasible values for the network frequency  $F_i$  of a node  $n_i$ . Apart from  $f_i$ , there is a tighter lower bound for  $F_i$ , as follows.

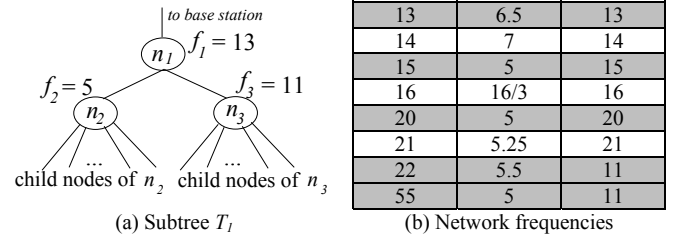
**Lemma 2:** Given a node  $n_i$  and the subtree  $T_i \subseteq T$  rooted at  $n_i$ , a lower bound for the network frequency  $F_i$  of  $n_i$  is

$$F_i^{LB} = \max\{f_j \mid \forall n_j \in T_i\} \quad (1)$$

**Proof:** (by induction) For any node  $n_j \in T_i$ , clearly  $F_j \geq f_j$  according to the problem definition. Suppose that a node  $n_x$  is along the path from  $n_j$  to  $n_i$  whose network frequency  $F_x$  satisfies that  $F_x \geq f_j$ . Let  $n_y$  be the parent node of  $n_x$ . Since  $n_x$  and  $n_y$  are locally synchronized, we have  $F_y \geq F_x \geq f_j$ . By induction,  $F_i \geq f_j$ .  $\square$

For instance, in Fig. 2a, the network frequency  $F_4$  of  $n_4$  must be no smaller than  $F_i^{LB} = \max\{f_1, f_2, f_3, f_4, f_5\} = 4$ . Observe that  $f_4 = 2 < F_i^{LB}$ ; consequently,  $f_4$  in our example *does not have any impact on the choice of  $F_4$* . To avoid complicated notations, in the rest of Section IV-A we assume that for each sensor  $n_i$ ,  $f_i = F_i^{LB}$ ; whenever this is not the case,  $F_i^{LB}$  is used in place of  $f_i$ .

However, besides  $F_{\max}$ , there is no obvious upper bound for  $F_i$ . Fig. 4a shows an example subtree rooted at sensor  $n_1$ , which has two children  $n_2$  and  $n_3$ . The sampling frequencies are  $f_1=13$ ,  $f_2=5$ , and  $f_3=11$ , respectively. Suppose that  $n_2$  (resp.  $n_3$ ) have  $Cl_2$  (resp.  $Cl_3$ ) child nodes, which all have identical sampling frequencies to their respective parent. The marginal cost of increasing  $F_2$  (resp.  $F_3$ ) is then multiplied by  $Cl_2+1$  (resp.  $Cl_3+1$ ), as every child of  $n_2/n_3$  must also increase their network frequency by the same amount to synchronize with their respective parent. Fig. 4b lists several values for  $F_1$  and the corresponding  $F_2$  and  $F_3$  that minimizes the overall cost of  $T_1$ , while satisfying the synchronization requirements. The best  $F_1$  depends on  $Cl_2$  and  $Cl_3$ : when both  $n_2$  and  $n_3$  have numerous (e.g.,  $>50$ ) children, the best  $F_1$  is 55, which in our example is a common multiple of both  $f_2$  and  $f_3$ , and thus, minimizes the cost of the subtrees  $T_2$  and  $T_3$ . Observe that this is far larger than either one of  $f_2$ - $f_3$ . In general,  $f_2$  and  $f_3$  can be non-integer values (e.g., 3.1 and 9.7), in which case the best  $F_1$  may exceed even the product of  $f_2$  and  $f_3$ , depending on the number of children attached to  $n_2$  and  $n_3$ .



(a) Subtree  $T_1$  (b) Network frequencies  
Fig. 4 Example of choosing network frequencies

The above example also demonstrates that there is an infinite search space for a network frequency value ( $F_1$ ), rendering even a brute-force solution rather complicated. We tackle the problem using a *top-down* approach. Two key issues are (i) how to determine the network frequencies of top-level nodes that are directly connected to the base station, and (ii) given the network frequency  $F_i$  of an internal node  $n_i$ , how to choose the network frequencies of its children. We first focus on (ii). In particular, consider a child  $n_j$  of  $n_i$ ; in order to synchronize  $n_i$  and  $n_j$ , their network frequencies must satisfy that  $F_i = k \cdot F_j$  for a positive integer  $k$ , according to Definition 1.

Meanwhile, since  $F_j \geq f_j$ , we have  $k \leq \lfloor F_i/f_j \rfloor$ . Hence, there is a finite number of possible  $F_j$ 's.

**Lemma 3:** Let  $n_i$  be an internal node of  $T$ , and  $n_j$  be a child of  $n_i$ . Given the network frequency  $F_i$  of  $n_i$ , the network frequency  $F_j$  must take one of the following values

$$F_j \in \left\{ F_i/k \mid k=1,2,3,\dots, \lfloor F_i/f_j \rfloor \right\} \quad (2)$$

For example, in Fig. 4, when  $F_i$  is set to 13, according to Lemma 3, there are two possible values (13 and  $13/2=6.5$ ) for  $F_2$ , and only one (13) for  $F_3$ . The question now is how to select the best  $F_j$  for the child  $n_j$  among the above possibilities. Intuitively, since  $F_i$  is already given, the choice of  $F_j$  only affects the descendants of  $n_j$ . Accordingly, the best  $F_j$  is the one that minimizes the subtree  $T_j$  rooted at  $n_j$ .

**Lemma 4:** Given a node  $n_i$ , suppose that its network frequency  $F_i$  is already determined. Let  $\Phi_i^{F_i}$  be the minimum cost of subtree  $T_i \subseteq T$  rooted  $n_i$  with respect to  $F_i$ . Then,  $\Phi_i^{F_i}$  can be computed recursively as follows:

$$\Phi_i^{F_i} = F_i + \sum_{\text{child } n_j \text{ of } n_i} \min \left\{ \Phi_j^{F_i/k} \mid k=1,2,\dots, \lfloor F_i/f_j \rfloor \right\} \quad (3)$$

where  $\Phi_j^{F_i/k}$  is the minimum cost of subtree  $T_j$  given  $F_j=F_i/k$ .

**Proof:** Consider an arbitrary child node  $n_j$  of  $n_i$ . According to the definition of Problem 1, besides  $f_j$ , the value of  $F_j$  only affects the network frequencies of  $n_j$ 's parent (i.e.,  $F_i$  of  $n_i$ ) and children (and, recursively, the descendants of  $n_j$ ). Since  $F_i$  is fixed and given, the choice of  $F_j$  has impact only on the cost of the subtree  $T_j$ . Therefore, the best  $F_j$  should minimize the cost of  $T_j$ , which, combined with Lemma 2, leads to Equation (3).  $\square$

As a special case, when  $n_i$  is a leaf node,  $\Phi_i^{F_i}$  is simply  $F_i$ . Based on Lemma 4, once  $F_i$  is determined, it is straightforward to compute  $\Phi_i^{F_i}$ , as well as the best network frequencies of all nodes in the subtree  $T_i$ , through exhaustive search. For instance, in Fig. 2, given  $F_4=6$ ,  $F_5$  must be 6, since  $f_5=4 > 6/2$ .  $F_1$  can be either 3 or 6. In the former case,  $F_2=F_3=3$ , hence  $\Phi_1^3=3+3+3=9$ . In case that  $F_1=6$ ,  $F_2$  can be 2, 3, or 6,  $F_3$  can be 3 or 6, and the minimum possible cost for  $T_1$  is  $\Phi_1^6=6+2+3=11$ . Since  $\Phi_1^3 < \Phi_1^6$ , the best value for  $F_1$  is 3, and, thus,  $\Phi_4^6=6+\Phi_1^3+6=21$ .

However, the number of possible values for a network frequency increases exponentially with the depth of the node in the tree topology. To reduce the number of potential values, we now describe our most important theoretical result, which entails the recursive equation (3), only if  $F_i$  is a *regular network frequency*, defined as follows.

**Definition 2 (Regular Network Frequency):** A network frequency  $F_i$  is regular, if and only if either (i)  $F_i=f_i$  or (ii) there exists a node  $n_j \neq n_i$  in the subtree  $T_i$  rooted at  $n_i$  such that  $F_i=k \cdot f_j$  for a positive integer  $k$ .

For instance, among the values of  $F_1$  listed in Fig. 4b, only 13, 15, 20, 22 and 55 are regular ones (shown in grey). In general,  $n_i$ 's regular network frequencies are  $f_i=13$  and multiples of  $f_2=5$  (e.g., 15, 20, 55), or  $f_3=11$  (22, 55), since all children of

$n_2$  and  $n_3$  have the same sampling frequencies as their respective parent.

**Theorem 2:** Given an internal node  $n_i$  and a network frequency  $F_i$ , let  $F_i'$  be the highest regular network frequency for  $n_i$  satisfying  $F_i' \leq F_i$ . Then:

$$\Phi_i^{F_i} = \Phi_i^{F_i'} \cdot F_i / F_i' \quad (4)$$

Furthermore, suppose that when  $F_i'$  is assigned to  $n_i$ , the best network frequency for each node  $n_j \in T_i$  is  $F_j'$ . Then, when  $F_i$  is assigned to  $n_i$ , the optimal network frequencies for the nodes in  $T_i$  are:

$$F_j = F_j' \cdot F_i / F_i', \forall n_j \in T_i \quad (5)$$

**Proof:** We first prove that Equation (5) gives a feasible solution for  $T_i$  that leads to a subtree cost of  $\Phi_i^{F_i}$  given in Equation (4). Because when  $F_i'$  is assigned to  $n_i$ ,  $F_j'$  is the optimal network frequency of any node  $n_j$  in  $T_i$ , it must also be a feasible network frequency for  $n_i$ . Hence,  $F_j \geq F_j' \geq f_j$ . Meanwhile, consider any internal node  $n_x \in T_i$  and one of its children  $n_y \in T_i$ . Let  $F_x'$  and  $F_y'$  be the optimal network frequencies of  $n_x$  and  $n_y$ , respectively, when  $F_i'$  is assigned to  $n_i$ . Clearly,  $F_x'$  is a multiple of  $F_y'$ . Let integer  $k_{xy}=F_x'/F_y'$ . According to Equation (5),  $F_x/F_y = F_x'/F_y' = k_{xy}$ . Hence,  $F_x$  is a multiple of  $F_y$ , which enables the local synchronization of  $n_x$  and  $n_y$ . Additionally, we have

$$\Phi_i^{F_i} = \sum_{n_j \in T_i} F_j = \sum_{n_j \in T_i} \frac{F_j' \cdot F_i}{F_i'} = \frac{F_i}{F_i'} \cdot \sum_{n_j \in T_i} F_j' = \frac{\Phi_i^{F_i'} \cdot F_i}{F_i'} \quad (6)$$

Next we prove by induction that  $\Phi_i^{F_i}$  given in Equation (4) is indeed the minimum cost when  $F_i$  is assigned to  $n_i$ . In the base case,  $n_i$  is a leaf node in  $T$ . The theorem holds trivially, since  $\Phi_i^{F_i}=F_i$ , and  $\Phi_i^{F_i'}=F_i'$ . In the induction step, suppose that each child node  $n_j$  of  $n_i$  satisfies the theorem. Specifically, given an arbitrary node  $n_x$ , let the function  $CR_x(F_x)$  that returns the highest regular network frequency for  $n_x$  satisfying that  $CR_x(F_x) \leq F_x$ . As a special case,  $CR_i(F_i)=F_i'$ . The induction assumption is stated as follows.

$$\forall \text{child } n_j \text{ of } n_i, \forall F_j, \frac{\Phi_j^{F_j}}{F_j} = \frac{\Phi_j^{CR_j(F_j)}}{CR_j(F_j)} \quad (7)$$

According to Lemma 2, when  $F_i$  is assigned to  $n_i$ ,  $F_j$  must be  $F_i/k$  for a positive integer  $k \leq \lfloor F_i/f_j \rfloor$ . Similarly, when  $F_i'$  is assigned to  $n_i$ ,  $F_j$  must be  $F_i'/k$  for a positive integer  $k \leq \lfloor F_i'/f_j \rfloor$ . We now prove that  $\lfloor F_i/f_j \rfloor = \lfloor F_i'/f_j \rfloor$  by contradiction. Suppose that  $\lfloor F_i/f_j \rfloor \neq \lfloor F_i'/f_j \rfloor$ . Then, since  $F_i' \leq F_i$ , we have  $\lfloor F_i'/f_j \rfloor < \lfloor F_i/f_j \rfloor$ , and, thus,  $F_i' < \lfloor F_i/f_j \rfloor \cdot f_j \leq F_i$ . However,  $\lfloor F_i/f_j \rfloor \cdot f_j$  a regular frequency for  $n_i$ , as it is an integer multiple of  $f_j$ . This contradicts with the fact that  $F_i'$  is the largest regular network frequency for  $n_i$  satisfying  $F_i' \leq F_i$ .

Substituting  $F_j$  with  $F_i/k$  and  $F_i'/k$  in Equation (7), respectively, we obtain

$$\forall k=1,2,\dots, \lfloor F_i/f_j \rfloor, \frac{\Phi_j^{F_i/k}}{F_i/k} = \frac{\Phi_j^{CR_j(F_i/k)}}{CR_j(F_i/k)} \quad (8)$$

$$\forall k=1,2,\dots,\lfloor F_i/f_j \rfloor, \frac{\Phi_j^{F_i/k}}{F_i'/k} = \frac{\Phi_j^{CR_j(F_i'/k)}}{CR_j(F_i'/k)} \quad (9)$$

Next we prove  $CR_j(F_i/k) = CR_j(F_i'/k)$  by contradiction. Suppose that  $CR_j(F_i/k) \neq CR_j(F_i'/k)$ . Because  $F_i' < F_i$ ,  $CR_j(F_i'/k) < CR_j(F_i/k)$ . Hence,  $F_i' < CR_j(F_i/k) \cdot k \leq F_i$ . Since  $CR_j(F_i/k)$  is a multiple of  $f_x$  of a node  $n_x$  in  $T_j \subseteq T_i$ , and  $k$  is a positive integer,  $CR_j(F_i/k) \cdot k$  is also a multiple of  $f_x$ , meaning that  $CR_j(F_i/k) \cdot k$  is a regular network frequency for  $n_i$  that is closer to  $F_i$  than  $F_i'$ , which contradicts with the definition of  $F_i'$ . Therefore,  $CR_j(F_i/k) = CR_j(F_i'/k)$ , and, consequently, the right hand side of Equation (8) equals that of Equation (9). Their respective left hand side must be equal as well:

$$\frac{\Phi_j^{F_i/k}}{F_i} = \frac{\Phi_j^{F_i'/k}}{F_i'} \quad (10)$$

Let  $k^* \leq \lfloor F_i/f_j \rfloor$  be the positive integer satisfying:

$$\forall k=1,2,\dots,\lfloor F_i/f_j \rfloor, \Phi_j^{F_i/k^*} \leq \Phi_j^{F_i'/k} \quad (11)$$

According to Lemma 3 and the fact that  $\lfloor F_i/f_j \rfloor = \lfloor F_i'/f_j \rfloor$ , we have

$$\Phi_i^{F_i'} = F_i' + \sum_{\text{child } n_j} \Phi_j^{F_i'/k^*} \quad (12)$$

Multiplying both sides of Inequality (11) by  $F_i/F_i'$ , we obtain

$$\forall k=1,2,\dots,\lfloor F_i/f_j \rfloor, \frac{\Phi_j^{F_i'/k^*} \cdot F_i}{F_i'} \leq \frac{\Phi_j^{F_i'/k} \cdot F_i}{F_i'} \quad (13)$$

Combining Equation (10) and Inequality (13), we obtain

$$\forall k=1,2,\dots,\lfloor F_i/f_j \rfloor, \Phi_j^{F_i/k^*} \leq \Phi_j^{F_i'/k} \quad (14)$$

Therefore, according to Lemma 3, we have

$$\begin{aligned} \Phi_i^{F_i} &= F_i + \sum_{\text{child } n_j} \Phi_j^{F_i/k^*} \\ &= F_i + \sum_{\text{child } n_j} \frac{\Phi_j^{F_i'/k^*} \cdot F_i}{F_i'} \\ &= \frac{F_i}{F_i'} \left( F_i' + \sum_{\text{child } n_j} \Phi_j^{F_i'/k^*} \right) \\ &= \frac{\Phi_i^{F_i'}}{F_i'} \cdot F_i \end{aligned} \quad (15)$$

The second line in the above equation is based on Equation (10), whereas the last step is based on Equation (12). Therefore, the theorem also holds for  $n_i$ .  $\square$

In the example of Fig. 4, consider an irregular frequency  $F_i=16$ .  $F_i'=15$  is the highest regular frequency satisfying  $F_i' \leq F_i$ . According to Fig. 4b, when  $F_i=16$  is assigned to  $n_1$ , the optimal values for  $F_2$  and  $F_3$  are  $16/3$  and  $16$  respectively. Meanwhile, when  $F_i'=15$  is assigned to  $n_1$ , the best network frequencies for  $n_2$  and  $n_3$  are  $F_2'=5$  and  $F_3'=15$  respectively. Clearly,  $\Phi_i^{16} / \Phi_i^{15} = F_2/F_2' = F_3/F_3' = 16/15$ .

Theorem 2 indicates that  $\Phi_i^{F_i}$  is a piece-wise linear function of  $F_i$ . Specifically, as  $F_i$  grows from one regular value to the next one,  $\Phi_i^{F_i}$  increases linearly with  $F_i$ ; when  $F_i$  reaches the next regular value,  $\Phi_i^{F_i}$  suddenly drops. Subsequently,  $\Phi_i^{F_i}$

again increases linearly with  $F_i$  until the next regular frequency, possibly with a different slope. Fig. 5 plots  $\Phi_i^{F_i}$  against  $F_i$  in the example of Fig. 4, assuming that  $n_2$  and  $n_3$  have 2 and 3 child nodes respectively. Regular frequencies for  $n_1$  between 13 and 55 are shown on the x-axis. The piece-wise linear pattern, as well as the sudden drops in  $\Phi_i^{F_i}$  at regular values for  $F_i$ , can be clearly observed in the plot.

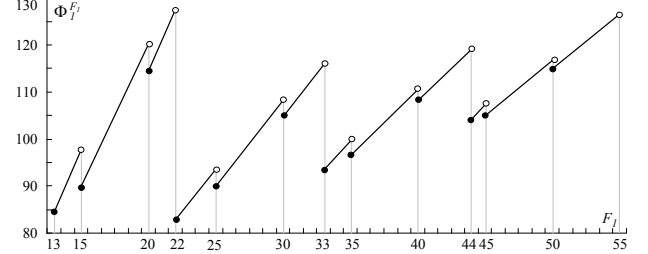


Fig. 5 Example plot of  $\Phi_i^{F_i}$  vs.  $F_i$

Another consequence of Theorem 2 is that an irregular  $F_i$  always leads to a higher subtree cost than the corresponding regular value  $F_i'$  preceding  $F_i$ . Therefore, for each top-level node  $n_i$  for which there is no restriction on its network frequency  $F_i$ , it suffices to consider only its regular values of  $F_i$ .

**Corollary 1:** Given a node  $n_i$  that is directly connected to the base station, the optimal network frequency of  $n_i$  must be regular.

For a node  $n_j$  whose parent is another sensor  $n_i$ , its network frequency is restricted to the values described in Lemma 3. Consequently, given an irregular frequency  $F_j$  of  $n_j$ , the highest regular frequency  $F_j'$  preceding  $F_j$  may not be able to synchronize with  $n_i$ . Hence, the best frequency for  $F_j$  can be irregular. For such nodes, the following algorithm utilizes Theorem 2 to reduce the computation of  $\Phi_j^{F_j}$  to that of  $\Phi_j^{F_j'}$ .

### B. Algorithm

Given the routing tree  $T$  and the sampling frequencies  $f_1, f_2, \dots, f_N$ , LS searches for the optimal network frequencies  $F_1^* - F_N^*$  in two steps. First, it computes the minimum cost  $\Phi^*$  of the entire WSN and gradually completes a hash table  $H$ . Second, guided by  $H$ , LS finds the optimal values  $F_1^* - F_N^*$  that sum up to  $\Phi^*$ . The hash table  $H$  contains entries of the form  $\langle n_i, F_i \rangle \rightarrow \Phi_i^{F_i}$ , where the key is a node  $n_i$  and a regular frequency  $F_i$  for  $n_i$ , and the value  $\Phi_i^{F_i}$  is the minimum cost of the subtree  $T_i$  rooted at  $n_i$ , given  $F_i$ .

The first step involves two functions, namely *CTC* (short for *compute total cost*) and *CSC* (for *compute subtree cost*). Fig. 6 illustrates *CTC*. The method initially computes the lower bound  $F_i^{LB}$  for the network frequency  $F_i$  of each node  $n_i$  (lines 2-4). According to Lemma 2,  $F_i^{LB}$  is a tighter bound than  $f_i$ ; whenever  $f_i < F_i^{LB}$ , the latter is used instead (specifically, in Equations (2), (3), and Definition 2). After that, the algorithm examines each top-level node  $n_i$ , and computes the minimum cost  $\Phi_i^{min}$  of the subtree  $T_i$  rooted at  $n_i$  (lines 5-10). In particular, line 7 enumerates all regular values for  $F_i$  that may possibly lead to the  $\Phi_i^{min}$ , according to Corollary 1. Line 8

invokes the recursive procedure *CSC* (described below) to compute  $\Phi_i^{F_i}$ ; the smallest  $\Phi_i^{F_i}$  among all values of  $F_i$  becomes  $\Phi_i^{min}$ . Finally, line 10 sums up  $\Phi_i^{min}$  for all top-level nodes to obtain the minimum overall cost  $\Phi^*$ .

---

*CTC*( $T, f_1-f_N$ ): returns  $\Phi^*$  // *CTC* for compute total cost  
// Input:  $T$ : routing tree,  $f_1-f_N$ : sampling frequencies of the sensors  
// Output:  $\Phi^*$ : optimal cost of the entire WSN

1. Initialize  $\Phi^*$  to 0, and  $H$  to an empty hash table
2. For each node  $n_i \in T$
3.     Compute  $F_i^{LB}$  according to Equation (1)
4.     If  $F_i^{LB} > f_i$ , use  $F_i^{LB}$  in place of  $f_i$  in the rest of the algorithm
5. For each  $n_i$  that is directly connected to the base station in  $T$
6.     Initialize  $\Phi_i^{min}$  to  $+\infty$
7.     For each regular  $F_i$
8.         Call  $\Phi_i^{F_i} = \text{CSC}(T, f_1-f_N, n_i, F_i, H)$
9.     If  $\Phi_i^{F_i} < \Phi_i^{min}$ , set  $\Phi_i^{min}$  to  $\Phi_i^{F_i}$
10.    Update  $\Phi^*$  to  $\Phi^* + \Phi_i^{min}$
11. Return  $\Phi^*$

---

Fig. 6 Algorithm *CTC*

Fig. 7 shows *CSC*, which recursively computes the optimal cost  $\Phi_i^{F_i}$  of the subtree  $T_i$  rooted at node  $n_i$ , given  $F_i$ . When  $F_i$  is irregular, the algorithm picks the highest regular frequency  $F'_i$  satisfying  $F'_i \leq F_i$ , calculates  $\Phi_i^{F'_i}$ , and applies Theorem 2 to obtain  $\Phi_i^{F_i}$  (lines 1-4). On the other hand, when  $F_i$  is regular, Lemma 4 is used to compute  $\Phi_i^{F_i}$  recursively (lines 6-13). The result is stored in a new entry of the hash table  $H$  with key  $\langle n_i, F_i \rangle$ . When the same parameters  $n_i, F_i$  are passed to *CSC*, the latter looks up  $H$  for the stored  $\Phi_i^{F_i}$  (line 5).

---

*CSC*( $T, f_1-f_N, n_i, F_i, H$ ): returns  $\Phi_i^{F_i}$  // *CSC* for compute subtree cost  
// Input:  $T$ : routing tree  
 $f_1-f_N$ : sampling frequencies of the sensors  
 $n_i, F_i$ : root node of the subtree  $T_i$ , and its network frequency  
//  $H$ : hash table storing subtree costs  
// Output:  $\Phi_i^{F_i}$ : minimum subtree cost of  $T_i$  given  $F_i$

1. If  $F_i$  is irregular with respect to  $n_i$
2.     Let  $F'_i < F_i$  be the highest regular network frequency for  $n_i$
3.     Call  $\Phi_i^{F'_i} = \text{CSC}(T, f_1-f_N, n_i, F'_i, H)$
4.     Calculate  $\Phi_i^{F_i}$  according to Equation (4), and return  $\Phi_i^{F_i}$
5. If  $H$  contains an entry with key  $\langle n_i, F_i \rangle$  and value  $\Phi_i^{F_i}$ , return  $\Phi_i^{F_i}$
6. If  $n_i$  is a leaf node, return  $F_i$
7. Initialize  $\Phi_i^{F_i}$  to  $F_i$
8. For each child  $n_j$  of  $n_i$
9.     Initialize  $\Phi_j^{min}$  to  $+\infty$
10.    For each value of  $F_j$  described in Equation (2)
11.        Call  $\Phi_j^{F_j} = \text{CSC}(T, f_1-f_N, n_j, F_j, H)$
12.     If  $\Phi_j^{F_j} < \Phi_j^{min}$ , set  $\Phi_j^{min}$  to  $\Phi_j^{F_j}$
13.    Update  $\Phi_i^{F_i}$  to  $\Phi_i^{F_i} + \Phi_j^{min}$
14. Add a new entry to  $H$  with key  $\langle n_i, F_i \rangle$  and value  $\Phi_i^{F_i}$
15. Return  $\Phi_i^{F_i}$

---

Fig. 7 Algorithm *CSC*

Fig. 8 describes the function *CNF* (short for *compute network frequencies*), which implements the second step of the

proposed solution, and computes the optimal frequencies top-down. The method starts by examining each top-level node  $n_i$ , testing regular network frequencies  $F_i$  of  $n_i$  (lines 3-7). Since  $F_i$  is regular, and *CTC* has called *CSC* with parameters  $n_i$  and  $F_i$ , the hash table  $H$  already contains an entry  $\langle n_i, F_i \rangle \rightarrow \Phi_i^{F_i}$ . The value of  $F_i$  that minimizes the subtree cost for  $T_i$  becomes the optimal frequency  $F_i^*$  of  $n_i$ . After determining the optimal frequencies for all top-level nodes, the algorithm continues to calculate the frequencies of their descendents. Specifically, given a node  $n_i$  with known  $F_i^*$  and one of its children  $n_j$ , *CNF* enumerates all possible values for  $F_j$ , and computes the minimum cost for subtree  $T_j$ . For a regular  $F_j$ ,  $\Phi_j^{F_j}$  is simply retrieved from  $H$ . When  $F_j$  is irregular,  $\Phi_j^{F_j}$  is obtained using Theorem 2 and  $H$ . In both cases, there is no need for recursive calls, since the computations have already been performed in *CSC*, and the results stored in  $H$ .

---

*CNF*( $T, f_1-f_N$ ): returns  $F_1^*-F_N^*$   
// *CNF* for compute network frequencies  
// Input:  $T$ : routing tree,  $f_1-f_N$ : sampling frequencies of the sensors  
// Output:  $F_1^*-F_N^*$ : optimal network frequencies of the nodes in  $T$

1. Call  $\Phi^* = \text{CTC}(T, f_1-f_N)$
2. Let  $H$  be the hash table used in *CTC*
3. For each node  $n_i$  that is directly connected to the base station  $n_0$
4.     Initialize  $\Phi_i^{min}$  to  $+\infty$ ,  $F_i^*$  to NULL
5.     For each regular  $F_i$
6.         Retrieve  $\Phi_i^{F_i}$  from  $H$  with key  $\langle n_i, F_i \rangle$
7.         If  $\Phi_i^{F_i} < \Phi_i^{min}$ , set  $\Phi_i^{min}$  to  $\Phi_i^{F_i}$ , and  $F_i^*$  to  $F_i$
8. Repeat
9.     For each node  $n_j$  satisfying (i)  $F_j^*$  has not been determined and (ii)  $F_i^*$  corresponding to the parent  $n_i$  of  $n_j$  has been determined
10.        Initialize  $\Phi_j^{min}$  to  $+\infty$ ,  $F_j^*$  to NULL
11.        For each value of  $F_j$  described in Equation (2)
12.            Compute  $\Phi_j^{F_j}$  using  $H$  and Theorem 2
13.            If  $\Phi_j^{F_j} < \Phi_j^{min}$ , set  $\Phi_j^{min}$  to  $\Phi_j^{F_j}$ , and  $F_j^*$  to  $F_j$
14. Until all optimal network frequencies are determined
15. Return  $F_1^*-F_N^*$

---

Fig. 8 Algorithm *CNF*

Fig. 9 illustrates an example of the above algorithms. Assume that  $F_{max}=100$  according to the application requirements. The sampling frequencies are shown alongside their corresponding nodes. Among them,  $f_3$  is smaller than the lower bound  $F_3^{LB}$  given by Lemma 2, which equals the maximum sampling frequency (i.e.,  $f_6=11$ ) in the subtree rooted at  $n_3$ . Therefore, in the computation of the optimal network frequencies, the algorithms simply discard the original value  $f_3=2$ , and proceed as if  $f_3$  were  $F_3^{LB}=11$ .

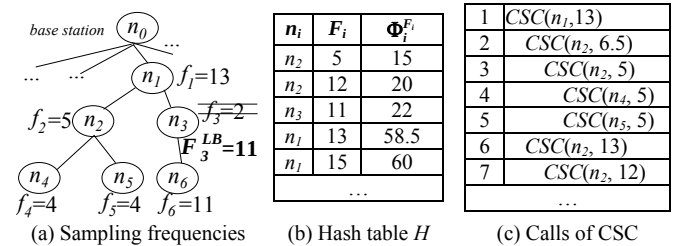


Fig. 9 Example of network frequency computation

Next we elaborate the computation of the optimal  $F_l^* - F_s^*$  in the subtree  $T_l$  rooted at  $n_l$ . Since  $n_l$  connects directly to  $n_0$ , *CTC* enumerates all its regular frequencies, which include (i)  $f_l=13$  and (ii) all multiples of  $f_2=5$ ,  $f_4=f_5=4$ , and  $F_3^{LB}=f_6=11$  below  $F_{max}=100$ . Assume that *CTC* first assigns  $F_l=13$ , and invokes *CSC* to compute  $\Phi_l^{13}$ . Fig. 9c shows the 7 subsequent calls to *CSC*, listing only parameters  $n_i$  and  $F_i$  since the others remain the same for all invocations of *CSC*. In the first call *CSC*( $n_l$ , 13), the hash table  $H$  does not have an entry with key  $\langle n_l, 13 \rangle$ ; thus, *CSC* computes  $\Phi_l^{13}$  recursively by identifying the best value for the network frequencies  $F_2$  and  $F_3$  of  $n_l$ 's child nodes  $n_2$  and  $n_3$ , given that  $F_l=13$ . We first focus on  $F_2$ . According to Lemma 3, there are two possible values for  $F_2$ :  $F_l=13$  and  $F_l/2=6.5$ . Suppose that *CSC* first calculates  $\Phi_2^{6.5}$  by recursively calling *CSC*( $n_2$ ,  $F_2=6.5$ ). However, 6.5 is not a regular frequency for  $n_2$  because it is not equal to  $f_2=5$ , or a multiple of  $f_4=f_5=4$ . Therefore, *CSC* finds the highest regular frequency  $F_2'=5$  below 6.5, and calls *CSC*( $n_2$ , 5).

To compute  $\Phi_2^5$ , which has no corresponding entry in  $H$ , *CSC* must determine the best  $F_4$  and  $F_5$  given  $F_2=5$ . According to Lemma 3, there is only one possible value 5 for both  $F_4$  and  $F_5$ . In the next two steps, *CSC* calls itself with parameters ( $n_4$ , 5) and ( $n_5$ , 5) respectively, which return  $\Phi_4^5 = \Phi_5^5 = 5$ . With this information, the third call to *CSC* (i.e., with parameters  $n_2$  and 5) returns  $\Phi_2^5 = 5 + \Phi_4^5 + \Phi_5^5 = 15$ , after adding a new entry  $\langle n_2, 5 \rangle = 15$  to  $H$ . Subsequently, the second call to *CSC* with parameters  $n_l$  and 6.5 then returns  $\Phi_2^{6.5} = \Phi_2^5 \cdot 6.5/5 = 19.5$ , according to Theorem 2. Next, the first call *CSC*( $n_l$ , 13) tests the other possible  $F_2=13$  for  $n_2$  by calling *CSC*( $n_2$ , 13), which leads to another invocation *CSC*( $n_2$ , 12) with the highest regular network frequency 12 below 13. The latter eventually returns  $\Phi_2^{12} = 20$ , and after *CSC*( $n_2$ , 13) we have  $\Phi_2^{13} = \Phi_2^{12} \cdot 13/12 = 65/3$ . Because  $\Phi_2^{6.5} < \Phi_2^{13}$ , the minimum cost for subtree  $T_2$  given  $F_l=13$  is  $\Phi_2^{6.5} = 19.5$ . The first call *CSC*( $n_l$ , 13) then computes the minimum cost for subtree  $T_3$  rooted at  $n_3$ , which is 26 when  $F_3=F_6=13$ . Accordingly,  $\Phi_l^{13} = 13 + 19.5 + 26 = 58.5$ , concluding the computation of *CSC*( $n_l$ , 13).

The outer function *CTC* then computes minimum subtree costs of  $T_l$  with other regular values of  $F_l$ , e.g., 15, 16, 20, 22, etc. Finally, *CTC* establishes that  $\Phi_l^{13} = 58.5$  is the minimum cost for subtree  $T_l$  among all values of  $F_l$ . After the termination of *CTC*, *CNF* computes the optimal network frequencies top-down. *CNF* enumerates all regular values of  $F_l$  and finds the one with the minimum  $\Phi_l^{F_l}$  using the hash table  $H$ . Having determined that  $F_l^*=13$ , *CNF* continues to calculate  $F_2^*$ ,  $F_3^*$ , and subsequently  $F_4^* - F_6^*$  based on the optimal subtrees stored in  $H$ .

**Theorem 3:** The space and time complexity of the algorithmic framework is  $O(N^2 \cdot C)$  and  $O(N^2 \cdot C^2)$ , respectively, where  $C = \lfloor F_{max}/f_{min} \rfloor$ , and  $f_{min} = \min \{f_i | 1 \leq i \leq N\}$ .

**Proof:** We first prove that algorithm *CTC* takes  $O(N^2 \cdot C)$  space and  $O(N^2 \cdot C)$  time. Its space consumption is dominated by the storage of the hash table  $H$ , which consists of entries of the

form  $\langle n_i, F_i \rangle \rightarrow \Phi_i^{F_i}$ , where  $n_i$  is an arbitrary node,  $F_i$  is a regular network frequency for  $n_i$ , and  $\Phi_i^{F_i}$  is the minimum cost of the subtree  $T_i$  rooted at  $n_i$ , when  $F_i$  is assigned to  $n_i$ . According to Definition 2,  $F_i$  must be a multiple of the sampling frequency  $f_j$  of a node  $n_j \in T_i$ . Since  $F_i \leq F_{max}$  and  $f_j \geq f_{min}$ , the number of possible regular network frequencies for  $n_i$  contributed by  $n_j$  is upper bounded by  $C = \lfloor F_{max}/f_{min} \rfloor$ . Since  $n_j$  can be any of the  $O(N)$  nodes in  $T_i$ , the total number of regular network frequencies for  $n_i$  is  $O(N \cdot C)$ . Considering that  $n_i$  can be any node in the WSN, the number of entries in  $H$  is  $O(N^2 \cdot C)$ .

Regarding time complexity, the dominating factor is the time consumed by subroutine *CSC*, which computes the optimal subtree cost for a given node  $n_i$  and its network frequency  $F_i$ . The cases when  $F_i$  is irregular, or when  $H$  contains an entry with key  $\langle n_i, F_i \rangle$  take negligible time to handle. Hence, it suffices to count the invocations of *CSC* with a regular  $F_i$  and a combination of  $\langle n_i, F_i \rangle$  not processed before. In each such invocation, *CSC* recursively calls itself to compute the minimum subtree cost corresponding to each child node  $n_j$  of  $n_i$ . Specifically, the algorithm considers all possible network frequencies for  $n_j$  given  $F_i$ , which is bounded by  $C$ . Let  $Cl_i$  be the number of the children of  $n_i$ , *CSC* finishes in  $O(Cl_i \cdot C)$  time. Counting all calls to *CSC* with fresh and regular inputs, the total time complexity is  $\sum_i O(C \cdot Cl_i \cdot N \cdot C) = O(N^2 \cdot C^2)$ .

Next we focus on *CNF*. Clearly, its space complexity is also dominated by the storage of  $H$ , which is bounded by  $O(N^2 \cdot C)$  as in *CTC*. Concerning time, after calling *CTC*, *CNF* enumerates all possible network frequencies of every node exactly once, which takes  $O(N^2 \cdot C)$  time. Hence, its time complexity is also  $O(N^2 \cdot C^2)$ .  $\square$

### C. Discussion

The value of  $C$  in Theorem 3 grows with  $F_{max}$ . Next we present optimizations to limit the impact of  $F_{max}$ , and significantly speed up network frequency computation. In the example of Fig. 9, one feasible solution is to set the network frequency of each sensor in  $n_1 - n_6$  to the highest sampling frequency  $f_l=13$  among the 6 sensors, leading to a total cost of 78 for subtree  $T_l$ . Clearly, setting  $F_l > 78$  will always result in a higher subtree cost for  $T_l$ . Furthermore, since each sensor  $n_i$  must have a network frequency  $F_i \geq F_i^{LB}$  according to Lemma 2, when  $F_l > 43$ , even when  $n_2 - n_5$  use their respective minimum possible network frequency, the total cost for  $T_l$  still exceeds that achieved with the above simple solution (i.e., 78). Therefore, it suffices to examine only regular frequencies for  $F_l$  not exceeding 43. Note that this is a much tighter bound than  $F_{max}=100$ . In addition, after obtaining  $\Phi_l^{13}=58.5$ , the upper bound for  $F_l$  can be further tightened to 23.5, since any higher value, plus the minimum possible network frequencies of  $n_2 - n_5$ , would result in a subtree cost for  $T_l$  higher than the current best 58.5.

Based on the above observations, we modify algorithm *CTC* as follows. In line 7 of Fig. 6, the regular frequencies are enumerated in increasing order. Let  $|T_i|$  be the number of



nodes in  $T_i$ ; an upper bound  $F_i^{UB}$  for  $F_i$  is initialized to  $\max\{f_j | n_j \in T_i\} \cdot |T_i| - \sum_{j \in i, n_j \in T_i} F_j^{LB}$ , and incrementally maintained as  $\Phi_i^{min} - \sum_{j \in i, n_j \in T_i} F_j^{LB}$  while the minimum subtree cost  $\Phi_i^{min}$  for  $T_i$  gets updated. The loop of lines 7-10 terminates as soon as  $F_i$  exceeds  $F_i^{UB}$ . Let  $F_{UB}$  be the maximum value of  $F_i^{UB}$  for all nodes after CTC finishes. If  $F_{max} > F_{UB}$ , the latter replaces the former in the complexity analysis, since it is the actual upper bound of all network frequencies used in our algorithms.

A similar optimization applies to algorithm CSC, when computing the minimum subtree cost  $\Phi_j^{min}$  (lines 10-12 in Fig. 7). Specifically, an upper bound  $\Phi_j^{min} - \sum_{l \in j, n_l \in T_j} F_l^{LB}$  for  $F_j$  is incrementally maintained, and the loop stops when  $F_j$  exceeds it. In our running example, the 6<sup>th</sup> call of CSC is eliminated, since  $F_2=13$  plus  $f_4$  and  $f_5$  already exceeds the previously computed subtree cost  $\Phi_2^{5.5}=19.5$ , when  $F_2$  is set to 6.5.

Finally, we discuss the adaptation of the above algorithms to changes of the routing tree  $T$ . A straightforward solution is to re-compute the network frequencies of all sensors whenever  $T$  is modified. The main challenge is to ensure that after the transition to the new network frequencies, all sensors remain locally synchronized, while satisfying their respective sampling frequency requirements. Specifically, immediately after the new tree is applied, all sensors in the WSN wake up simultaneously, start a new cycle, and switch to their new network frequencies thereafter. This method assigns the optimal network frequency to each sensor at all times, but incurs considerable cost during transitions. An alternative approach reduces the cost of transitions as follows. First, based on historical data, we partition the sensors into *dynamic nodes*, which change their respective parents frequently, and *static* ones with stable parent nodes. Then, for each minimal subtree  $T_i$  containing at least one dynamic node, we apply global synchronization to its corresponding subtree  $T_i$ , i.e., all nodes in  $T_i$  share the same network frequency.  $T_i$  is then treated as a single node in algorithms CTC, CSC and CNF, with a weight proportional to the number of nodes in  $T_i$ . Accordingly, network frequencies need to be re-computed only when at least one static node changes its parent in  $T$ , which happens infrequently.

## V. FINDING ROUTING TREES

This section focuses on Problem 2, i.e., finding the best routing tree from a given connectivity graph. Following the common practice in the WSN literature, we restrict the search space for routing schemes to *min-hop routing trees* [1]. Unfortunately, even under this additional restriction, Problem 2 is still intractable. Therefore, we resort to heuristic methods.

Based on the min-hop property, LS partitions nodes into *layers*, so that the  $l$ -th layer contains nodes requiring at least  $l$  hops to reach  $n_0$ . Clearly, in a min-hop tree, the parent of a layer- $l$  node must reside at layer  $l-1$ . LS starts from the layer  $l_{max}$  of nodes that need the most hops to reach  $n_0$  (i.e., the leaves of the routing tree  $T$ ) and builds  $T$  bottom-up. Specifically, whenever LS examines a node  $n_j$  at level  $l$ , the subtree  $T_j$  rooted at  $n_j$  containing nodes from levels  $l+1$  to  $l_{max}$ , has already been constructed. LS then extends  $T_j$  by adding an

appropriate parent  $n_i$  of  $n_j$ . Let  $P_j$  denote the set of *candidate parents* of  $n_j$ , which consists of layer  $l-1$  nodes that are connected to  $n_j$  in  $G$ . The selection of  $n_i$  is performed according to the following heuristics.

**Heuristic 1:** Given a node  $n_j$ , the parent  $n_i \in P_j$  of  $n_j$  should satisfy that  $f_i \geq F_j^{LB}$  (Equation 1). If no such node exists in  $P_j$ , the parent of  $n_j$  is the node  $n_i$  with the highest  $f_i$  among all nodes in  $P_j$ .

**Heuristic 2:** Given a node  $n_j$ , if multiple nodes in  $P_j$  have sampling frequencies no less than  $F_j^{LB}$ , the parent of  $n_j$  is the node  $n_i \in P_j$ , that satisfies  $f_i \geq F_j^{LB}$  and minimizes  $f_i / \lfloor f_i / F_j^{LB} \rfloor$ .

Intuitively, Heuristic 1 aims at minimizing the value  $\Delta_i = F_i - f_i$  for the parent node  $n_i$ . In particular, in order to synchronize  $n_i$  and  $n_j$ ,  $F_i$  must be no less than  $F_j$ , which, in turn, is lower bounded by  $F_j^{LB}$  according to Lemma 2. Hence, the heuristic tries to pick a node  $n_i \in P_j$  satisfying  $f_i \geq F_j^{LB}$ , in which case it is possible that  $\Delta_i = 0$ . When there is no such node in  $P_j$ ,  $\Delta_i$  is lower bounded by  $F_j^{LB} - f_i$ . Thus, the node  $n_i$  with the highest  $f_i$  (i.e., minimal  $F_j^{LB} - f_i$ ) is chosen as  $n_j$ 's parent. Heuristic 2, on the other hand, tries to minimize the network frequency  $F_j$  for the child node  $n_j$ . Specifically, since  $F_i$  must be a multiple of  $F_j$ , the lowest  $F_j$  is obtained, when  $F_i = f_i$ , and  $F_j = F_i / \lfloor f_i / F_j^{LB} \rfloor = f_i / \lfloor f_i / F_j^{LB} \rfloor$ . Thus, Heuristic 2 chooses the parent  $n_i$  that minimizes this value. Fig. 10 shows an algorithm that applies the above heuristics. *Find\_Routing\_Tree* first partitions all nodes into layers (line 1), and builds  $T$  bottom-up. If a node  $n_i$  is directly connected to the base station  $n_0$  in  $G$ ,  $n_0$  becomes the parent of  $n_i$ . Otherwise, the best parent  $P_i$  of  $n_i$  is chosen according to Heuristic 1 (line 11) or 2 (line 12).

---

```

Find_Routing_Tree( $G, f_1, \dots, f_N$ ): returns  $T$ 
// Input:  $G$ : connectivity graph,  $f_1, \dots, f_N$ : sampling frequencies
// Output:  $T$ : routing tree of the WSN
1. Partition sensor nodes into layers
2. Let layer  $l_{max}$  be the deepest layer
3. Initialize  $T$  with nodes  $n_1, \dots, n_N$ , and no edges
4. For  $l = l_{max}$  DownTo 1
5.   For each node  $n_j$  on layer  $l$ 
6.     If  $l=1$ , add edge  $\langle n_0, n_j \rangle$  to  $T$ 
7.     Else
8.       Compute  $F_j^{LB}$  according to Equation (1)
9.       Let  $P_j$  be the set of candidate parents of  $n_j$ 
10.      If there does not exist  $n_i \in P_j$  such that  $f_i \geq F_j^{LB}$ 
11.        Choose  $n_i \in P_j$  according to Heuristic 1
12.      Else, choose  $n_i \in P_j$  according to Heuristic 2
13.      Add edge  $\langle n_i, n_j \rangle$  to  $T$ 
14. Return  $T$ 

```

---

Fig. 10 Algorithm *Find\_Routing\_Tree*

The algorithm takes  $O(N^2)$  time (for choosing the best parent for each node) and  $O(N^2)$  space (for storing  $G$ ), where  $N$  is the number of sensors. Since the network frequency assignment module has the same space and time complexities with respect to  $N$  according to Theorem 3, the proposed solution to Problem 2 takes  $O(N^2)$  time and space overall, meaning that it easily scales to large WSNs. Furthermore, our experiments, shown next, demonstrate that *Find\_Routing\_Tree* usually

identifies high-quality trees that lead to significant energy savings.

Finally, changes in the connectivity graphs are handled as follows. Whenever at least one link in the current routing tree  $T$  is broken, we re-compute  $T$  using *Find\_Routing\_Tree*, as well as the network frequencies of the sensors. Otherwise, *Find\_Routing\_Tree* is invoked periodically, and the transition process starts when a better routing tree is detected.

## VI. EXPERIMENTAL EVALUATION

We have implemented the proposed methods in C++, and carried out all experiments on a Core 2 Duo 2.6GHz PC with 2GBytes of memory. We use two real datasets, *Greenorbs* (94 sensors), *Intel Lab* (52 sensors), and a synthetic one.

**GreenOrbs :** *GreenOrbs* [16] is deployed by our group in a forest area in China. It contains 94 active sensors measuring temperature, humidity and light per minute. In our experiments, we use the temperature readings. The WSN follows the collection tree protocol (CTP) [6]. Specifically, sensors are organized into a routing tree, which is periodically adjusted based on the current link quality conditions. We observed that during a period of 12 hours, there is only a small number of timestamps when the tree changed. Furthermore, a large part of the tree, which forms the backbone of the network remained stable throughout the testing period. Fig. 11a shows the geographic locations of the sensors and the base station, as well as the connectivity graph of the entire WSN.

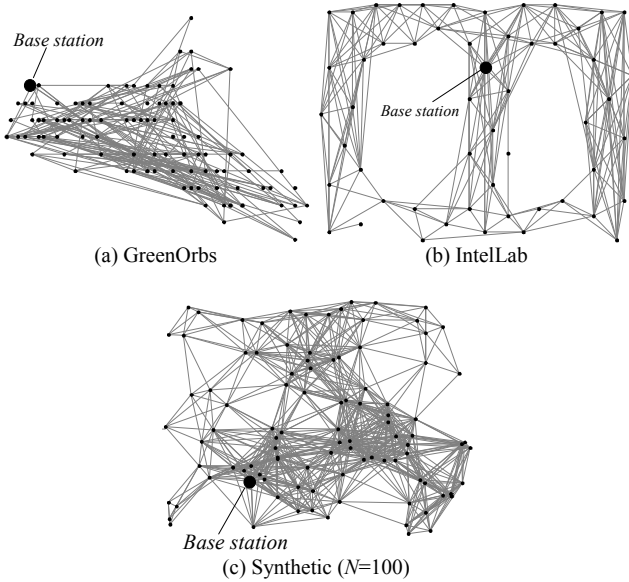


Fig. 11 Connectivity graphs of the datasets used in the experiments

We set the sampling frequency of each sensor  $n_i$  based on the change rate of  $n_i$ 's readings. The intuition is that a sensor with stable readings should sample less frequently than another whose readings change with a high rate. Specifically, suppose that node  $n_i$  has collected  $m$  samples  $s_{i,1}, s_{i,2}, \dots, s_{i,m}$  at  $m$  timestamps  $t_{i,1}, t_{i,2}, \dots, t_{i,m}$ . The change rate  $cr_i$  of  $n_i$  is calculated by:

$$cr_i = \frac{\sum_{j=1}^{m-1} |s_{i,j+1} - s_{i,j}|}{\sum_{j=1}^{m-1} t_{i,j+1} - t_{i,j}} \quad (16)$$

Let  $f_{max}$  be the highest sampling frequency required by the application. The sampling frequency  $f_i$  of a node  $n_i$  is then:

$$f_i = \frac{cr_i \cdot f_{max}}{\max_{j=1}^N cr_j} \quad (17)$$

In our experiments, we use  $f_{max}=20$ . Note that the specific value of  $f_{max}$  does not affect the relative performance of the proposed methods. The maximum possible sampling frequency  $F_{max}$  is set to  $10 \cdot f_{max}=200$ . In all experimental settings, during the search for the optimal network frequencies, the algorithms always terminate earlier (i.e., using the optimizations described in Section IV-C) without testing the last regular frequency before  $F_{max}$ .

**IntelLab:** *IntelLab*<sup>4</sup> includes data from 54 sensors that measure temperature, humidity, light and voltage every 31 seconds for a month. Similarly to *GreenOrbs*, we use temperature readings in the experiments. The sampling frequencies of the sensors are computed using Equations (16) and (17). The dataset does not include the connectivity graph  $G$ ; instead, it lists the probability for each sensor  $n_i$  to successfully deliver a message directly to a surrounding node  $n_j$ . Accordingly, we add an edge  $\langle n_i, n_j \rangle$  to  $G$ , whenever the probability of successful packet delivery between  $n_i$  and  $n_j$  is above 20%. Except for one sensor (ID=5), all others are connected in  $G$ . Meanwhile, the dataset does not mention the location or the connectivity of the base station. Hence, we simply treat sensor with ID 1 as the base station. Fig. 11b shows the resulting connectivity graph. Finally, there is no information on the routing tree.

**Synthetic:** We generate sensor locations uniformly within a  $[0,1] \times [0,1]$  square. In the connectivity graph  $G$ , there is an edge  $\langle n_i, n_j \rangle$ , if and only if, the distance between nodes  $n_i$  and  $n_j$  does not exceed 0.25. Fig. 11c displays the connectivity graph of 100 sensors. Additionally, the sampling frequencies are random numbers in the range  $[1, 100]$ , which follow the Zipf distribution. Similar to the real datasets, the maximum possible network frequency  $F_{max}$  is set to  $10 \cdot f_{max}=1000$ .

We investigate the energy savings and computational cost of LS, under four parameters: (i) number of nodes, (ii) distribution of the sampling frequencies, (iii) shape of the routing tree, and (iv) the number of dynamic nodes (i.e., with frequently changing parents). Sections VI-A and VI-B present results on real and synthetic datasets, respectively.

### A. Results for GreenOrbs and IntelLab

*GreenOrbs* and *IntelLab*, like most WSN currently under deployment, are restricted to relatively few nodes (under 100), for which the computational overhead of LS is negligible. Therefore, we defer the discussion on this cost for the

<sup>4</sup> Available at <http://db.csail.mit.edu/labdata/labdata.html>

synthetic data, and focus on the energy overhead. Specifically, let  $t_w$  be the number of time units that a sensor is active per cycle, and  $P$  be the energy consumed per time unit. Given the sum of network frequencies  $\Phi$ , the energy consumed by all sensors is  $\Phi \cdot P \cdot t_w$ , which is proportional to  $\Phi$ . Hence, in the following we simply report the value of  $\Phi$  as the overall energy cost. We compare LS against the traditional globally synchronized (GS) approach, which sets the network frequency of each sensor to  $f_{\max} = \max\{f_1, f_2, \dots, f_N\}$ . Clearly, the energy cost of GS is  $\Phi = N \cdot f_{\max}$ . We include two versions of LS: *LS-Problem1* and *LS-Problem2*. The former considers a given routing tree  $T$ , whereas the latter computes  $T$ .

We first evaluate the impact of the number of nodes  $N$  on the total energy cost  $\Phi$ , with a static tree  $T$ . Specifically, for *GreenOrbs*, *LS-Problem1* uses a real routing tree  $T_{\text{real}}$ , taken at an arbitrary timestamp. To vary  $N$ , we randomly remove leaf nodes from  $T_{\text{real}}$  (and recursively, entire subtrees). *IntelLab*, on the other hand, does not include real routing trees. Hence, we evaluate only *LS-Problem2*, and vary  $N$  by removing random nodes. The sampling frequency of each node  $n_i$  is fixed to the value computed based on the real change rate of  $n_i$ 's readings, as described in the beginning of this section. Fig. 12 plots  $\Phi$  as a function of  $N$ . Clearly,  $\Phi$  grows with  $N$  for both LS and GS, since more sensors naturally lead to higher energy consumption. Comparing GS with LS, the former consumes significantly more energy, and the difference increases with  $N$  (note the logarithmic scale on the vertical axis). This is because when a new node  $n_i$  is added, GS always assigns  $F_i = f_{\max}$ ; LS, on the other hand, usually sets  $F_i$  to be far lower than  $f_{\max}$  since  $n_i$  only needs to locally synchronize with its adjacent nodes. *LS-Problem2* outperforms *LS-Problem1*, indicating that the proposed solution indeed generates better routing trees.

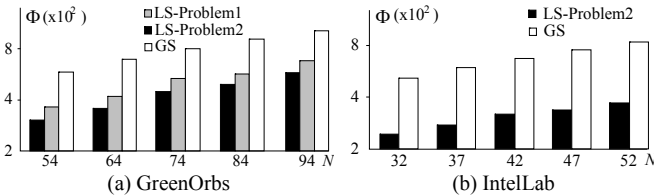


Fig. 12 Energy cost  $\Phi$  vs. number of nodes  $N$

Next, we fix  $N$  to its maximum value, and study the effect of the sampling frequencies in the range  $[1, 100]$ , generated according to Zipf distribution with skewness factor  $\alpha$ . *LS-Problem1* (resp. *LS-Problem2*) uses the real routing tree (resp. tree computed by the proposed heuristics) as before. Fig. 13 demonstrates the effect of  $\alpha$  on  $\Phi$ . Given the small node cardinality, a growing value of  $\alpha$  decreases the probability that  $f_{\max}$  reaches its maximum value 100. Consequently, the cost of all methods drops. LS takes better advantage of this fact since it can isolate the effect of the high frequencies in their respective subtrees.

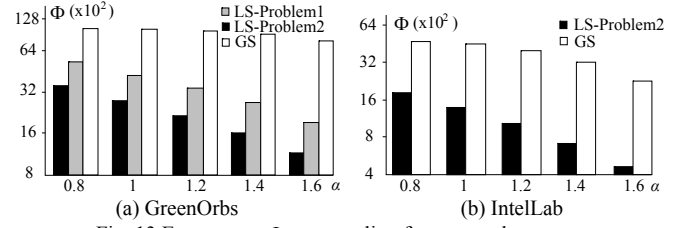


Fig. 13 Energy cost  $\Phi$  vs. sampling frequency skewness  $\alpha$

Finally, we evaluate the impact of changing routing trees. Our implementation of LS is based on the second method discussed at the end of Section IV, which minimizes re-computations of network frequencies. In the *GreenOrbs* dataset, we identified 41 dynamic nodes among the real routing trees, which reside at the lowest levels of the tree. On average, network frequency re-computations are performed once every 40 times when  $T$  changes, which itself happens infrequently. Overall, LS using real routing trees achieves 34% energy savings compared to GS. To further investigate the effect of routing trees with different volatility, we randomly mark a number  $N_D$  of nodes as dynamic in a bottom-up fashion, and evaluate the energy efficiency of LS. The total number of nodes  $N$  is fixed to its maximum value, and the sampling rates are derived from real readings. Fig. 14 demonstrates  $\Phi$  against varying  $N_D$ . The energy consumption of LS generally increases with  $N_D$ , as more nodes are merged and share the same network frequencies. When the majority of sensors are dynamic, LS reduces to GS. Nevertheless, LS achieves considerable energy savings, even for relatively large values of  $N_D$  ( $\sim 50\%$  of  $N$ ).

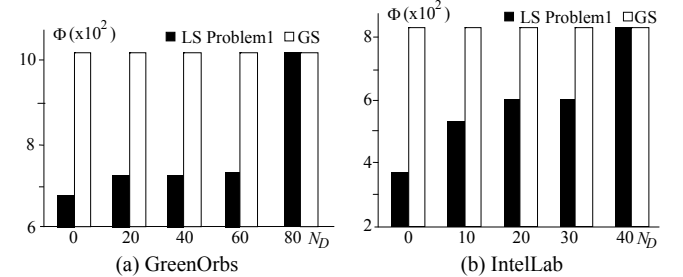


Fig. 14 Energy cost  $\Phi$  vs. number of dynamic nodes  $N_D$

## B. Results for Synthetic Datasets

We repeat our experiments on a much larger synthetic dataset described in the beginning of this section. In addition to energy cost, we report the CPU time and the memory required by the proposed algorithms. We focus on static routing trees; the results for changing trees lead to similar conclusions as in *GreenOrbs* and *IntelLab*, and are omitted. Fig. 15 shows the effect of the number of nodes  $N$ , after fixing the skewness factor  $\alpha$  of the sampling frequencies to 0.8. Similar to the real datasets, the energy overhead of all methods increases with  $N$ , and LS consistently outperforms GS. The CPU and memory overhead of LS grows quadratically with  $N$ , as predicted by Theorem 3. Nevertheless, even for  $N=10000$ , these costs are very low (i.e., less than 320 milliseconds and 7Mbytes), which confirms that the proposed algorithms could be utilized for WSN much larger than the ones currently deployed.

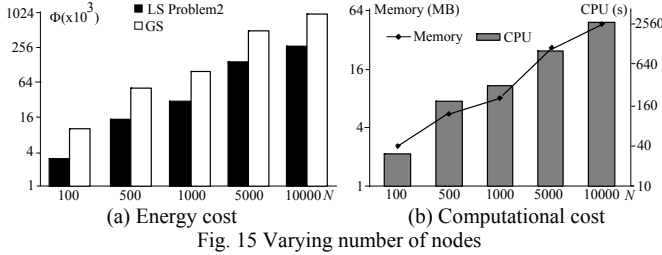


Fig. 15 Varying number of nodes

Fig. 16 fixes  $N=5000$ , and varies the skewness factor  $\alpha$  of the sampling frequencies. Observe that, unlike the case of real datasets, the energy consumption of GS remains the same for all values of  $\alpha$  because, due to the large number of nodes, there is always one that has the maximum sampling frequency 100. On the other hand, as  $\alpha$  grows, the consumption of LS decreases for the reasons explained in the context of Fig. 13. The computational overhead of LS also drops because a lower  $\Phi$  leads to a more restrictive upper bound on a node's network frequency. Consequently, the optimization of Section IV-C becomes more effective, leading to an earlier termination of the algorithm.

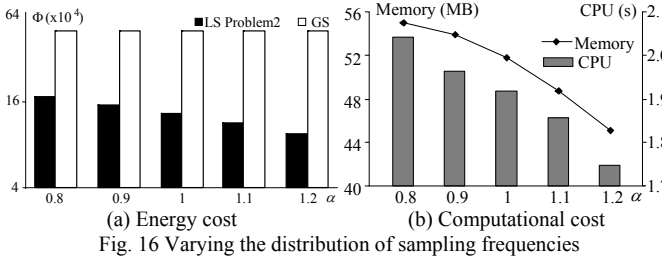


Fig. 16 Varying the distribution of sampling frequencies

Fig. 17 investigates the impact of the routing tree on LS-Problem1. Specifically, we generate random trees with a given fanout  $fan$  for each internal node, and present the average of their results, after setting  $N=5000$  and  $\alpha=0.8$ . Since  $fan$  has no effect on GS, we report the ratio between the energy cost of LS and that of GS. The energy savings of LS increase with  $fan$  due to the fact that the height of the tree decreases; thus, a node  $n_i$  with high  $f_i$  influences fewer ancestors. The computational overhead also increases with  $fan$  because a higher fanout leads to a larger subtree for each internal node, and, consequently, more regular network frequencies to examine.

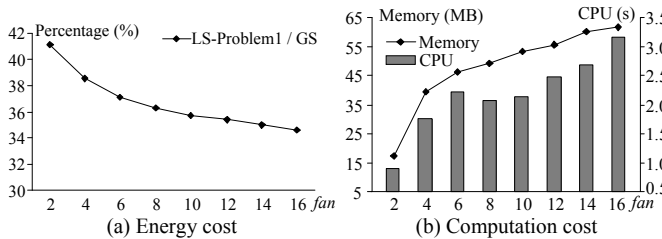


Fig. 17 Varying the shape of the tree

## VII. CONCLUSION

This paper presents a novel framework that allows sensors to sample at different rates, while ensuring timely routing of packets. We focus on two versions of the problem, depending on whether the WSN topology is fixed or not. Extensive experiments, using real and synthetic datasets, demonstrate the effectiveness of local synchronization in both versions. In the future we plan to investigate the computation of routing trees that minimize the network frequencies and at the same time minimize the packet losses. Finally, another interesting direction is the extension of the proposed techniques to multi-path topologies.

## REFERENCES

- [1] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E. Wireless Sensor Networks: a Survey. *Computer Networks*, 38(4):393-422, 2002.
- [2] Considine, J., Li, F., Kollios, G., Byers, J. Approximate Aggregation Techniques for Sensor Databases. *ICDE*, 2004.
- [3] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J. M., Hong, W. Model-Driven Data Acquisition in Sensor Networks. *VLDB*, 2004.
- [4] Elson, J., Roemer, K. Wireless Sensor Networks: A New Regime for Time Synchronization. *Computer Communication Review*, 33(1):149-154, 2003.
- [5] Ganeriwal, S., Kumar, R., Srivastava, M. Timing-Sync Protocol for Sensor Networks. *ACM SenSys*, 2003.
- [6] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P. Collection Tree Protocol. *ACM SenSys*, 2009.
- [7] Jurdak, R., Baldi, P., Lopes, C. Adaptive Low Power Listening for Wireless Sensor Networks. *IEEE TMC*, 6(8):988-1004, 2007.
- [8] Kim, S., Fonseca, R. Reliable Transfer on Wireless Sensor Networks. *IEEE SECON*, 2004.
- [9] Levis, P., Gay, D. *TinyOS Programming*. Cambridge University Press, 2009.
- [10] Li, Q., Rus, D., Global Synchronization in Sensor Networks. *IEEE Transactions on Computers*, 55(2):214-226, 2006.
- [11] Madden, S., Franklin, J. Fjording the Stream: An Architecture for Queries over Streaming Sensor Data. *ICDE*, 2002.
- [12] Madden, S., Franklin, M., Hellerstein, J. The Design of an Acquisitional Query Processor for Sensor Networks. *SIGMOD*, 2003.
- [13] Madden, S., Franklin, M., Hellerstein, J., Hong, W. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. *OSDI*, 2002.
- [14] Madden, S., Szewczyk, R., Franklin, M., Culler, D. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. *IEEE WMCSA*, 2002.
- [15] Manjhi, A., Nath, S., Gibbons, B. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. *SIGMOD*, 2005.
- [16] Mo, L., He, Y., Liu, Y., Zhao, J., Tang, S., Li, X., Dai, G. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. *ACM SenSys*, 2009.
- [17] Nath, S., Gibbons, P., Seshan, S., Anderson, Z. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *SenSys*, 2004.
- [18] Shnayder, V., Hempstead, M., Chen, B., Allen, G., Welsh, M. Simulating the Power Consumption of LargeScale Sensor Network Applications. *ACM SenSys*, 2004.
- [19] Silberstein, A., Braynard, R., Yang, J. Constraint Chaining: On Energy-Efficient Continuous Monitoring in Sensor Networks. *SIGMOD*, 2006.
- [20] Silberstein, A., Munagala, K., Yang, J. Energy-Efficient Monitoring of Extreme Values in Sensor Networks. *SIGMOD*, 2006.
- [21] Wu, Y., Li, X., Liu, Y., Lou, W. Energy-Efficient Wake-up Scheduling for Data Collection and Aggregation. *IEEE TPDS*, 21(2):275-287, 2009.
- [22] Yang, X., Lim, H., Özsu, M., Tan, K. In-Network Execution of Monitoring Queries in Sensor Networks. *SIGMOD*, 2007.