Optimal Cluster Association in Two-Tiered Wireless Sensor Networks

WeiZhao Wang* Wen-Zhan Song† Xiang-Yang Li* Kousha Moaveni-Nejad*

Abstract— **In this paper, we study the two-tiered wireless sensor network (WSN) architecture and propose the optimal cluster association algorithm for it to maximize the overall network lifetime. A two-tiered WSN is formed by number of small sensor nodes (SNs), powerful application nodes (ANs), and base-stations (BSs, or gateways). SNs capture, encode, and transmit relevant information to ANs, which then send the combined information to BSs. Assuming the locations of the SNs, ANs, and BSs are fixed, we consider how to associate the SNs to ANs such that the network lifetime is maximized while every node meets its bandwidth requirement. When the SNs are homogeneous (e.g., same bandwidth requirement), we give optimal algorithms to maximize the lifetime of the WSNs; when the SNs are heterogeneous, we give a 2-approximation algorithm that produces a network whose lifetime is within $1/2$ of the optimum. We also present algorithms to dynamically update the cluster association when the network topology changes. Numerical results are given to demonstrate the efficiency and optimality of the proposed approaches. In simulation study, comparing network lifetime, our algorithm outperforms other heuristics almost twice.**

Keywords— **Network lifetime, wireless sensor networks, two-tiered, clustering.**

I. INTRODUCTION

The advances in Micro Electro-Mechanical Systems (MEMS) technology, digital circuit design and wireless communication have enabled the built of small, cheap and low power sensors, *e.g.*, the Mica2 and Mica2Dot motes [1]. With the introduction of these components, the new systems which are composed of thousands even millions of tiny computing devices interacting with the environment and communicating with each other becomes possible. In the meanwhile, several possible applications based on wireless sensor networks composed of thousands of such tiny device are expected to come into practice in the near future. These applications may include but are not restricted to: (1) *military applications* in the battle field, i.e., enemy surveillance, target tracking [2], [3] and countersniper systems [4]; (2) *environmental monitoring* in the countryside, i.e., microclimate monitoring on Great Duck Island, Maine [5], [6]; (3) *structural monitoring and emergency rescue*, i.e., structural health monitoring of the Golden Gate Bridge in San Francisco.

The low power of these tiny sensors raises a unique challenge for the large scale wireless sensor networks. Even these small sensors are able to act as a router to store and relay the data for other small sensors, it may take tens or even hundreds of hops for the information to reach the base station which is clearly not affordable in most circumstances. Another concern is that these small sensors that are one-hop away from the base station would have to route nearly every piece of information generated by the sensors in the network, which makes them run-out of power (called *die*) very quickly. Thus, the architecture of WSNs should be revisited. Due to its possible massive size, it is natural to build the large scale WSNs in a hierarchical model. In fact, several works have already addressed different issues regarding this hierarchical architecture, including minimizing the number of clusters [7], [8], minimizing the total energy consumption [9] and maximizing the lifetime [10], [11]. Following the work in [10], we consider how to maximize the lifetime of a wireless sensor network with thousands of tiny and simple sensor nodes (SNs) deployed in a region. In addition, there are several application nodes (ANs) who collect the information from small sensors, process the collected information to produce a local-view and send the information back to the base-station(BS) if necessary. Every application node can communicate with the base station either by some direct radio communication or routing through the relay of other application nodes. The base station often can connect to the Internet and is in charge of building the global view of the WSN, controls the application nodes, sends the data to remote site and reports

*Illinois Institute of Technology, Chicago, IL, USA, Email: wangwei4@iit.edu,xli@cs.iit.edu, moavkoo@iit.edu. The research of Xiang-Yang Li was supported in part by NSF CCR-0311174.

*Washington State University, Vancouver, WA 98686, Email: songwz@wsu.edu. The research of Wen-Zhan Song is supported by NASA ESTO 05-AIST05-0082

emergency events if necessary.

Usually the small sensors sleep for most of the time, only periodically sample the data and send to application nodes. Thus, once deployed, they usually could last several months even years before their battery needed to be replaced, if it is feasible. Although the application node has much more power compared with the small sensor nodes, they also consume much more energy due to the following reasons: (1) the ANs need to wake up or stay idle for much longer time, which is usually proportional to the number of small sensors in the cluster; (2) the ANs usually have transmission of bitstreams over much greater distances; (3) the ANs need to do more extensive local computations. If not carefully designed, the application nodes usually can only survive a shorter period of time than the small sensors can. On the other hand, if one or two small sensors die, the entire WSN will generally function properly while one or two application nodes running out of power could result in the complete lost of the coverage of certain area. Thus, here we focus on how to maximize the lifetime of application nodes instead of small sensors. In a WSN, usually a global view should be maintained with certain quality in order to work properly. Remember that the global view is built from the local-views sent by those application nodes, thus, the lifetime of the WSN heavily depends on the lifetime of certain ANs. Therefore, we define the lifetime of a wireless sensor network as a certain function of the lifetime of application nodes. In [11], the authors study the effect of the position of the base station on the lifetime of the WSN. In their model, they assumed that there are already some clusters and each cluster is composed of one application node and several small sensors. Each application node in a cluster can receive the information from all small sensors in the same cluster and send it to the base station either by direct connection or intermediate routing. However, how to form the cluster and what effect the cluster formation has on the lifetime of the WSNs has not been answered. In this paper, we mainly focus on how to form the clusters efficiently and study the effect of cluster formation on the lifetime of the WSN.

We categorize the WSN into four different cases: whether the SNs are homogeneous or heterogeneous, whether the ANs are homogeneous or heterogeneous. Here SNs are said to be homogeneous if the average produced data rates of SNs are same; otherwise, they are said to be heterogeneous. The ANs are said to be homogeneous if (1) the energy consumption function of every AN (based on total amount of data produced by the SNs in its cluster) is the same, and (2) the initial power of ANs are same; otherwise, these ANs are said to be heterogeneous. The main contributions of this paper are as follows. First, we present a series of max-flow-based methods to form the clusters that optimally maximize the lifetime of the WSN when the small sensors are homogenous. We separately study the cases when the ANs are homogenous or heterogeneous. Since these methods are centralized, we then present a new approach (called smoothing) with a low time complexity that is suitable for dynamic updating. When the small sensors are heterogeneous we present an algorithm to form clusters whose lifetime is no less than $\frac{1}{2}$ of the optimum. We then conduct extensive simulations to study the practical performances of our method compared with the performances of some simple heuristics. Our theoretical results are corroborated by our simulation studies.

The remainder of the paper is organized as follows. In Section II, we present some preliminaries and previous works. In Section III, we study the scenario in which the small sensors are the same. We then study the generalized case when the small sensors could be different in Section IV. We also discuss some other issues in Section V. Extensive simulations have been conducted to study the practical performances of our proposed solutions. We conclude our paper in Section VII with some possible future directions.

II. Preliminaries

A. Two-Tiered Wireless Sensor Networks

A two-tiered wireless sensor network (WSN) consists of a set of small sensor nodes (SN), denoted as $S_M = \{s_1, s_2, \cdots, s_m\}$, a set of application nodes (AN), denoted as $V_N = \{v_1, v_2, \cdots, v_n\}$, and at least one base station (BS). The ANs and SNs form *clusters*, and in each cluster there are many SNs and one AN. For simplicity, we assume that the application node v_i is in cluster $\mathcal{C}_i$ and the set of small sensors in cluster $\mathcal{C}_i$ is $S_i \subseteq S_M$. A small sensor, once triggered by the internal timer or some external signals, starts to capture and encode the environmental phenomena (such as temperature, moisture, motion measure, etc) and broadcast the data directly to all ANs within its transmission range and to certain ANs via the relay

of some other neighboring sensors. Here, if AN v_i can receive the data from the small sensor s_j, then we call v_i is a neighbor of s_j. Here sensor s_j may have to reach AN v_i via relay of other sensors. For notational simplicity, we use $N(v_i)$ to denote the neighboring small sensors of AN v_i. Remember that although several ANs can receive the data packets from the small sensor s_j, only the AN in the same cluster as s_j processes the information. Here, we assume that once formed, the cluster formation does not change over the time. We also let r_i be the data-rate of the small sensor s_i generates and $r(S) = \sum_{s_i \in S} r_i$ be the total data-rates produced by a set of small sensors S. Usually, the data-rate $r_i(t)$ is a function over the time t instead of a constant. However, if we average the rate over a period of time T, e.g., one day or one week, most often it is a constant. Thus, we can define the rate r_i as the the average rate over a period of time, i.e., $r_i = \frac{\int_{T_0}^{T+T_0} r_i(t)}{T}$. When receiving the raw data from SNs from its cluster, an AN might create an application specific local-view for the whole cluster by exploring some correlations among the data sent by different SNs. In the meanwhile, some data fusion can be conducted by ANs to alleviate the redundance in the raw data sent by SNs. After an AN creates a local-view of the data, it then forwards the information to a BS that generates a comprehensive global-view for the entire WSN. Notice that here an AN can communicate directly with a BS, or optionally, ANs can be involved in inter-AN relaying if such activities are needed and applicable.

B. Energy Model and Notations

In this paper, we assume that SNs wake up periodically to collect, process and transmit the data to the ANs. This assumption is natural and used often in practice. For instance, the small sensors are configured to transmit once every 5 mins in the Great Duck Island project [6]. After the formation of the clusters, the ANs are able to decide the time slot in which they need to wake up to receive the data sent by SNs from its own cluster. It is critical to reduce the wake up time due to the high power consumption for idle listening, *i.e.*, sometimes it is as large as the receiving power and about $1/2$ of the transmit power. Recently minimizing the wake up time has been addressed at different layers [12], [13], [14]. In this paper, our focus is on how to form clusters properly so that the network lifetime is maximized. How to schedule the wake-up time for SNs has been addressed extensively [15], [16], [17] and our formation could be coupled with any of these schemes. It is reasonable to expect that the live time of an AN *decreases* when the number of small sensors in its cluster *increases*. Thus, for an AN, its energy consumption mainly are composed of three parts: (1) the energy consumed to receive the information from the small sensors of its own cluster, (2) the overhearing cost incurred by those sensors not belonging its cluster and can reach this AN; (3) the energy consumed to process the information and the energy consumed to send the information to the base station; and (4) the energy consumed when the ANs are idle listening. We implicitly assume that the power consumption of (2) - (4) is a fixed value. Energy consumption for ANs in different applications and scenario may be different. Thus, we do not rely on any special assumptions about energy consumption. Given an AN v_i, let $S_i \in S_M$ be the set of small sensors in its logical cluster. The power consumption of the AN v_i is a general function $p_i(r(S_i), N(v_i))$, where $r(S_i)$ is the total data-rate of the small sensors in S_i. Since $N(v_i)$ does not depend on the cluster formation and can be taken as a constant for a given application node v_i, we can simplify the power consumption function as $p_i(r(S_i))$. This model takes into account all other power consumption that is a fixed value. The only assumption in this paper is that function $p_i(x)$ should satisfy that $p_i(x) > p_i(x')$ when $x > x'$, i.e., the more information the small sensors in the cluster generate, the more energy the application node consumes. Notice that, the above monotone increasing property is only assumed to be true for each AN. For two different ANs v_i and v_j, it is possible that $p_i(x) < p_j(x')$ when $x > x'$.

C. Lifetime of a Two-Tiered WSN

In this paper, we assume that $\mathcal{P}_i$ is the initial battery power level of the application node v_i and $p_i(r(S_i))$ is its *average* energy consumption rate when the set of small sensors S_i is in the cluster $\mathcal{C}_i$. The lifetime of an individual AN v_i is define as $l_i = \frac{\mathcal{P}_i}{p_i(r(S_i))}$. According to the *criticality* of a mission, several different definitions of sensor network lifetime has been defined in the literature.

- CRITICAL APPLICATION NODE LIFETIME (CANLT): The mission fails when any AN runs out of energy, i.e., the lifetime L_N is $L_N = \min_{i=1}^{N}\{l_i\}$. The first AN that run out of energy are denoted as the critical

AN. If mission fails when α percentage ANs are run out of energy, then it is called α-Critical Application Node Lifetime (α-CANLT).

• FULL COVERAGE LIFETIME (FCLT): A small sensor is called a *covered* sensor if it has at least one alive AN neighbor. The total sensing area of all *covered* sensors is called the covered area of the WSN here. The mission fails when the covered area of the WSN is smaller than the originally covered area. If mission fails when the ratio of the covered area over the originally covered area is smaller than percentage β, then it is called β-Full Coverage Lifetime (FCLT).

In the literature, the definition of network lifetime implicitly assumes that small sensors have longer lifetime than application nodes. The reasons include, but are not limited to, several practical considerations: (1) In many applications, small sensors are densely deployed to provide better tolerance. Hence, the small sensors could schedule themselves to several independent groups, and when one group is on duty, other groups can sleep. Thus, the lifetime of small sensors in a group can be prolonged as a functional unit. (2) Application nodes need collect data from small sensors and send to base station which might be far away, which often costs more energy.

In this paper, we adopt the above definitions when conducting theoretical analysis and simulations. Notice that, there are also several other different definitions of lifetime of a wireless sensor network, e.g., see [11]. Our analysis can be extended to those scenarios similarly.

D. Previous Works

Numerous literatures have discussed efficient cluster formation for wireless ad hoc and sensor networks. Although almost all works assumed that there are some nodes acting as *clusterheads* who are in charge of gathering the information from other nodes and sending back to some base stations, the criteria of forming the clusters vary from case to case. One fundemental difference between the cluster formation problem studied in this paper and the traditional cluster formation problems is that *every* node could be a clusterhead in the traditional methods, while only the AN can be the clusterhead for the problems studied here.

In the Linked Cluster Algorithm (LCA) [7], a node becomes the clusterhead if it has the highest identity among all nodes within one hop of itself or among all nodes within one hop of one of its neighbors. This algorithm was improved by the LCA2 algorithm [18], which generates a smaller number of clusters. The LCA2 algorithm elects the node, with the lowest ID among all nodes which are not within 1-hop of any chosen clusterheads, as a new clusterhead. The algorithm proposed in [19], chooses the node with highest degree among its $1-$hop neighbors as a clusterhead. In [20], the authors propose a distributed algorithm that is similar to the LCA2 algorithm. The Distributed Clustering Algorithm (DCA) uses weights associated with nodes to elect clusterheads [21]. It elects the node that has the highest weight among its 1-hop neighbors as the clusterhead. The DCA algorithm is suitable for networks in which nodes are static or moving at a very low speed. The Max-Min $d-$cluster Algorithm proposed in [22] generates $d-$hop clusters with a run-time of $O(d)$ rounds. All above approaches are aiming to minimize the number of clusters such that any node in any cluster is at most d hops away from the clusterhead.

In [23], the authors proposed a clustering algorithm that aims at maximizing the lifetime of the network by determining optimal cluster size and optimal assignment of nodes to clusterheads. They assumed that the clusterhead consumes power to send the data to the nodes in its cluster via broadcast. Thus, the power consumption of the cluterhead depends on its transmission range, not on the number of nodes in this cluster. This is fundamentally different from our model in which the ANs consume power to process and relay the collected information that is closely related to the number of small sensors in the cluster.

Results reported in [10], [11] are closest to this paper in spirit. In [10], Pan *et al.* studied the problem of maximizing lifetime of a two-tiered WSN with focus on the top-tier. By assuming the *prior known* fixed cluster formation, the authors mainly studied how to place the base-station in the network such that the lifetime of the WSN is maximized. The ANs are assumed to be homogenous in [10] and generalized to be heterogenous in [11]. The authors also discussed how to relay the packets via ANs to some fixed based stations. In this paper, we will focus on the lower-tier of the two-tiered WSN: how to form the cluster (associate small sensors to application nodes) so the network lifetime is maximized.

III. Homogeneous Small Sensors

In this section, we study the case when the small sensors are homogeneous, *i.e.*, all small sensors have the same data rate, say r. Thus $r(S) = r \cdot |S|$, where $|S|$ is the number of small sensors in the set S. We specifically discuss how to maximize the lifetime under the critical application node lifetime (CANLT) definition when the application nodes are homogenous in subsection III-A and heterogenous in subsection III-B.

A. Homogeneous Application Nodes

In this subsection, we discuss how to maximize the lifetime of the WSN when all application nodes are homogeneous, i.e., their initial on-board energy are the same, say $\mathcal{P}$ and the energy consumption functions are the same, say $p(x)$. Remember that $L_N = \min_{i=1}^{n}\{l_i\} = \min_{i=1}^{n} \frac{\mathcal{P}}{p(r\cdot|S_i|))}$ and $p(x)$ is increasing. Thus maximizing the lifetime L_N of the WSN is equivalent to minimizing the maximum cluster size. For simplicity, we denote $x_{i,j} = 1$ if the sensor s_j belongs to cluster $\mathcal{C}_i$, and $x_{i,j} = 0$ otherwise. Let $N(v_i)$ be the set of sensors who are v_i's neighbors. We formalize the problem of maximizing L_N as the following Integer Programming (IP).

$$\min \max_{v_i \in V_N} \sum_{s_j \in S_M} x_{i,j} \tag{1}$$

Subject to constraints

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{2}$$
$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \tag{3}$$
$$\sum_{v_i} x_{i,j} = 1, \forall s_j \tag{4}$$

Obviously, a feasible solution of the above IP problem is a *feasible* cluster formation. For simplicity, the set of small sensors in the cluster $\mathcal{C}_i$ is denoted as S_i in this paper, when no confusion is caused. Next we present two different approaches to solve the above IP exactly.

A.1 Efficient Centralized Approach

First, we use a Max-Flow approach to solve the IP (1). By adopting the traditional techniques and relaxing the constraints $x_{i,j} \in \{0,1\}$ to $0 \le x_{i,j}$, we transform the IP (1) to a linear programming as follows:

$$\min T \tag{5}$$

subject to constraints

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{6}$$
$$x_{i,j} \ge 0, \forall v_i, \forall s_j; \tag{7}$$
$$\sum_{s_j \in S_M} x_{i,j} \le T, \forall v_i; \tag{8}$$
$$\sum_{v_i} x_{i,j} = 1, \forall s_j \tag{9}$$

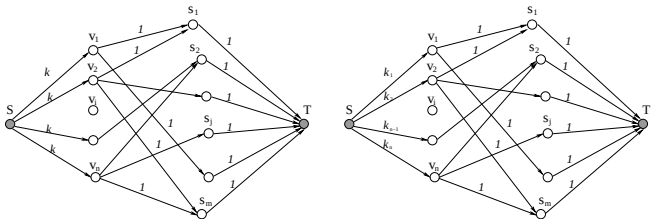
Given the linear programming, we first construct a flow network as shown in Figure 1 with s as the source and t as the sink. There is a directional link $\overrightarrow{sv_i}$, $1 \le i \le n$ with capacity k, a directional link between $\overrightarrow{v_i s_j}$ with capacity 1 if $s_j \in N(v_i)$ and a directional link $\overrightarrow{s_j t}$ with capacity 1. If $T \le k$, then a solution $\mathbf{x}$ of the above linear programming implies a feasible flow for the corresponding flow network defined in Figure 1: $x_{i,j}$ is the flow from v_i to s_j. On the other hand, when $k \le T$, a feasible flow f with total flow m for flow network defined in Figure 1 implies a feasible solution for the linear programming also: $x_{i,j}$ is the flow $f(v_i, s_j)$. In the following Lemma 1 we show the relation between a feasible solution to the feasible system (6) and the maximum network flow in the network shown in Figure (1).

Lemma 1: If we fix $T = k$, then there is a feasible solution to the FS (6) if and only if the maximum flow of the network illustrated by Figure 1 is m, where m is the cardinality of S_M.

Proof: The proof that if there is a feasible solution to the FS (6) then the maximum flow of the network illustrated by Figure 1 is m is trivial and omitted here. We only prove that if the maximum flow of the network illustrated by Figure 1 is m then there is a feasible solution to the FS (6). Without loss of generality, let $f_{i,j}$ denote the flow on link $v_i s_j$ and obviously, $f_{i,j} = 0$ for all v_i and $s_j \notin N(v_i)$ because the link $\overrightarrow{v_i s_j}$ does not exist. Notice that for every node v_i, all the inflow comes from link $\overrightarrow{sv_i}$, thus $\sum_{s_j \in S_M} f_{i,j} \le k$ for each node v_i. If the cardinality of the flow is m, then the flow on each link $\overrightarrow{s_j t}$ is 1 which implies that $\sum_{v_i} x_{i,j} = 1$. This proves that $f_{i,j}$ is a feasible solution to FS (6). ■

Usually, for a maximum flow problem, the flow on each directional link could be any real number. Fortunately, the solution generated by the flow-network, illustrated by Figure 1, has a well-known property.

Lemma 2: If the capacity function takes on only integral values, then the maximum flow f has the



(a) homogeneous SNs (b) heterogenous SNs

Fig. 1. A flow network for two-tiered WSN.

property that $|f|$ is integer-valued. Moreover, for all vertices u and v, the flow on edge uv is an integer.

Lemma 2 immediately shows that the flow on each link $\overrightarrow{v_i s_j}$ is either 0 or 1. Remember that the flow on link $\overrightarrow{v_i s_j}$ corresponds to $x_{i,j}$, which implies that $x_{i,j} \in \{0, 1\}$ for every v_i and $s_j \in N(v_i)$. Thus, a feasible solution to FS (6) also satisfies the constraints (2). Let $\mathbf{x^{min}}$ be the solution to IP (1) and $T^{\min} = \min \max_{v_i \in V_N} \sum_{s_j \in S_M} \mathbf{x}_{i,j}^{min}$.

If we know the exact value of $T^{\min}$, then we can find $\mathbf{x}^{\min}$ by solving the maximum flow problem in network (1). Remember that $T^{\min}$ is a non-negative integer and is at most m, thus by performing a binary search on $T^{\min}$ we can find the exact value.

We can find the solution to IP (1) by solving $\log m$ max-flow problems for different values of T. Thus, the time complexity for Max-Flow approach is $m \cdot \log m \cdot (n+m)^3$, which is very expensive and impractical. Notice that the cluster formation problem with minimum cluster size becomes the Maximum Cardinality Matching problem in a bipartite graph [24]. In [24], Hopcroft and Karp presents the best known algorithm that achieves the time complexity $\sqrt{m} \cdot nm$. This reduces the time complexity from $O((n+m)^3)$ to $O(nm \cdot (n+m)^{1/2} \log(n+m))$ for a fix value T. Therefore, we can solve the IP (1) in time $O(n \cdot m^{3/2} \log^2(m))$.

A.2 Efficient Distributed Algorithm by Smoothing

Although the previous approach computes a clustering quickly in centralized manner, it may be too expensive to collect the necessary information. In this subsection, we propose a different approach that can be implemented efficiently in a distributed manner. The basic idea of this approach is to construct a virtual directed graph on ANs and iteratively move the sensors from those clusters who have the largest number of small sensors to smaller clusters. In the virtual directed graph, there is an edge $\overrightarrow{v_i v_k}$ from AN v_i to v_k if there is a sensor s_j that can be moved from the cluster of v_i to the cluster of v_k. The weight of the edge is the number of such small sensors that can be moved from the cluster of v_i to the cluster of v_k. Following algorithm presents the method constructing a virtual graph based on a feasible solution $\mathbf{x}$ to FS (2).

Algorithm 1 Constructing the virtual graph

Input: A set of ANs V_N, a set of small sensors S_M and a feasible solution $\mathbf{x}$, *e.g.*, assigning s_j randomly to a v_i where $s_j \in N(v_i)$.

Output: A directed virtual graph $VG(\mathbf{x})$.

```
Set V_N as the vertices for virtual graph VG.
for every pair of v_i and s_j such that x_{i,j} = 1 do
    for every v_k such that s_j ∈ N(v_k) do
        if there is no directed edge v_i v_k from v_i to v_k then
            Add a directed edge v_i v_k from v_i to v_k.
            Set the weight of the edge to c(v_i v_k) = 1.
        else
            Update the weight as c(v_i v_k) = c(v_i v_k) + 1.
```

In the directed virtual graph $VG(\mathbf{x})$, if there is a path from v_i to v_j, then we say v_i *reaches* v_j. All vertices that v_i can reach forms a set $\mathcal{R}_i(\mathbf{x})$, called the *clique* centered at the AN v_i. Given a solution $\mathbf{x}$ of FS (2) and its corresponding virtual graph $VG(\mathbf{x})$, we have the following property about cliques (its proof is omitted due to space limit).

Lemma 3: Given a feasible assignment $\mathbf{x}$ of small sensors to ANs and its corresponding virtual graph $VG(\mathbf{x})$, for any AN v_i and its clique $\mathcal{R}_i(\mathbf{x})$ in $VG(\mathbf{x})$, if $\mathbf{x}'$ is also a feasible assignment of SNs to ANs, we have $\sum_{v_j \in \mathcal{R}_i(\mathbf{x})} |S_i(\mathbf{x})| \leq \sum_{\mathbf{v_j} \in \mathcal{R}_\mathbf{i}(\mathbf{x}')} |\mathbf{S_i}(\mathbf{x}')|$

The Algorithm relies on the relation between $\omega_i(\mathbf{x})$, $\omega_j(\mathbf{x})$ and T^{min} where v_i is the AN with the largest weight and v_j is the AN with the smallest weight in $\mathcal{R}_i(\mathbf{x})$.

Lemma 4: Let v_i be the AN with the largest weight and v_j be the AN with the smallest weight in $\mathcal{R}_i(\mathbf{x})$ under any feasible assignment $\mathbf{x}$, then $|S_j(\mathbf{x})| \leq T^{min} \leq |S_i(\mathbf{x})|$.

Proof: Remember that v_i has the maximum weight among all ANs, thus $T^{min} \leq |S_i(\mathbf{x})|$ trivially holds. Therefore, we only need to prove $|S_j(\mathbf{x})| \leq \mathbf{T}^{\min}$. We prove this by contradiction. For the sake of contradiction, we assume all ANs reachable by v_i has a weight greater than T. From the assumption

that $|S_k(\mathbf{x})| > \mathbf{T}^{\min}$ for every $v_k \in \mathcal{R}_i(\mathbf{x})$, we have $S(\mathcal{R}_i(\mathbf{x})) = \sum_{\mathbf{v_k} \in \mathcal{R}_\mathbf{i}(\mathbf{x})} |\mathbf{S_k}(\mathbf{x})| > \mathbf{T}^{\min} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x})|$, where $|\mathcal{R}_i(\mathbf{x})|$ denotes the number of ANs reachable by v_i.

Let $\mathbf{x^{min}}$ be the solution to IP (1). From Lemma 3, we obtain that $\sum_{v_k \in \mathcal{R}_i(\mathbf{x})} |S_k(\mathbf{x^{min}})| \geq \sum_{\mathbf{v_k} \in \mathcal{R}_\mathbf{i}(\mathbf{x})} |\mathbf{S_k}(\mathbf{x})| > \mathbf{T}^{\min} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x})|$. Remember that $T^{\min} \cdot |\mathcal{R}_i(\mathbf{x})| \geq \sum_{\mathbf{v_k} \in \mathcal{R}_\mathbf{i}(\mathbf{x})} |\mathbf{S_k}(\mathbf{x^{min}})|$. Thus $T^{\min} \cdot |\mathcal{R}_i(\mathbf{x})| > \mathbf{T}^{\min} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x})|$, which is a contradiction. This finishes our proof. ■

Given a virtual graph constructed by Algorithm 1 based on a feasible assignment of SNs to ANs, our approach to find a better solution is to iteratively apply a process called SMOOTH to reduce the maximum weight of the application nodes if possible. Here, the weight of an application node v_i under a feasible assignment $\mathbf{x}$ is the number of small sensors assigned to the cluster $\mathcal{C}_i$, denoted as $\omega_i(\mathbf{x})$.

Algorithm 2 Smooth Algorithm

Input: A feasible assignment $\mathbf{x}$.
Output: A solution to IP (1).

1: Construct virtual graph $VG(\mathbf{x})$ based on the feasible assignment $\mathbf{x}$ using Algorithm 1.
2: **repeat**
3: Find the AN with the largest weight, say v_i. If there are more than one such ANs, choose one randomly.
4: Find the AN with the smallest weight in $\mathcal{R}_i(\mathbf{x})$, say v_j. If there are more than one such ANs, choose one randomly.
5: Apply procedure SMOOTH$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$.
6: **until** $\omega_i(\mathbf{x}) \leq \omega_\mathbf{j}(\mathbf{x}) + \mathbf{1}$

Theorem 5: Algorithm 2 terminates after at most m iterations, with an solution to IP (1).

Proof: From Lemma 4, we have $\omega_j(\mathbf{x}) \leq \mathbf{T}^{\min} \leq \omega_\mathbf{i}(\mathbf{x})$. If Algorithm 2 does not stop at this iteration, we have $\omega_i(\mathbf{x}) > \omega_\mathbf{j}(\mathbf{x}) + \mathbf{1}$, which implies that $T^{min} > \omega_j(\mathbf{x}) + \mathbf{1}$. For a feasible solution $\mathbf{x}$, we define $\delta_i(\mathbf{x}) = |\mathcal{S}_\mathbf{i}(\mathbf{x})| - |\mathcal{S}_\mathbf{i}(\mathbf{x}^{\min})|$ if $\omega_i(\mathbf{x}) > \mathbf{T^{min}}$ and 0 otherwise. Let $\Delta(\mathbf{x}) = \sum_{\mathbf{v_i} \in \mathbf{V_N}} \delta_\mathbf{i}(\mathbf{x})$, it is not difficult to observe that $\Delta(\mathbf{x})$ will be decreased by 1 for each iteration. Thus, Algorithm 2 terminates after at most m iterations.

Remember when Algorithm 2 terminates, we have $\omega_i(\mathbf{x}) \leq \omega_\mathbf{j}(\mathbf{x}) + \mathbf{1}$. Combining with the relation $\omega_j(\mathbf{x}) \leq \mathbf{T^{min}} \leq \omega_\mathbf{i}(\mathbf{x})$, we have $\omega_j(\mathbf{x}) \leq \mathbf{T^{min}} \leq \omega_\mathbf{i}(\mathbf{x}) \leq \omega_\mathbf{j}(\mathbf{x}) + \mathbf{1}$. This implies that $T^{min} = \omega_i(\mathbf{x}) - \mathbf{1}$ or $T^{min} = \omega_i(\mathbf{x})$. First, we consider the case when $T^{min} = \omega_i(\mathbf{x}) - \mathbf{1}$. In this case, we have $\omega_j(\mathbf{x}) \geq \mathbf{T}^{\min}$ for every $v_j \in \mathcal{R}_i(\mathbf{x})$ which implies $\sum_{v_j \in \mathcal{R}_i(\mathbf{x})} |S_i(\mathbf{x})| \geq \mathbf{T}^{\min} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x}) - \mathbf{1}| + \mathbf{T}^{\min} + \mathbf{1}$. For the solution $\mathbf{x}^{min}$ of IP (1), every AN's weight is not greater than T^{min}. Thus, $\sum_{v_j \in \mathcal{R}_i(\mathbf{x})} |S_i(\mathbf{x^{min}})| \leq \mathbf{T}^{\min} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x})|$. From Lemma 3, we have $\sum_{v_j \in \mathcal{R}_i(\mathbf{x})} |S_i(\mathbf{x^{min}})| \geq \sum_{\mathbf{v_j} \in \mathcal{R}_\mathbf{i}(\mathbf{x})} |\mathbf{S_i}(\mathbf{x})|$. This implies that $T^{min} \cdot |\mathcal{R}_i(\mathbf{x})| \geq \mathbf{T^{min}} \cdot |\mathcal{R}_\mathbf{i}(\mathbf{x}) - \mathbf{1}| + \mathbf{T^{min}} + \mathbf{1}$, which is a contradiction. Thus, $T^{min} = \omega_i(\mathbf{x})$. Remember that $\mathbf{x}$ is a solution to the FS (2). Therefore, $\mathbf{x}$ is a solution to IP (1). This finishes our proof. ■

Algorithm 3 SMOOTH$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$

Input: A feasible assignment $\mathbf{x}$ and its corresponding directed virtual graph $VG(\mathbf{x})$, a pair of nodes v_i and v_j.

1: Let $v_{i_0} v_{i_1} \cdots v_{i_k}$ be the path connecting v_i and v_j with the minimum number of hop. Here, $v_{i_0} = v_i$ and $v_{i_k} = v_j$.
2: **for** $t = 0$ to $k - 1$ **do**
3: Assume that $x_{t,l} = 1$ for some SN s_ℓ with $s_\ell \in N(v_t)$ and $s_\ell \in N(v_{t+1})$. Set $x_{t,l} = 0$ and $x_{t+1,l} = 1$, i.e., move s_ℓ from cluster $\mathcal{C}_t$ to cluster $\mathcal{C}_{t+1}$.
4: **for** every v_a such that $s_\ell \in N(v_a)$ **do**
5: Update $c(\overrightarrow{v_{i_t} v_a}) = c(\overrightarrow{v_{i_t} v_a}) - 1$. Remove directed link $\overrightarrow{v_{i_t} v_a}$ if $c(\overrightarrow{v_{i_t} v_a}) = 0$.
6: Update $c(\overrightarrow{v_{i_{t+1}} v_a}) = c(\overrightarrow{v_{i_{t+1}} v_a}) + 1$. Add a directed link $\overrightarrow{v_{i_{t+1}} v_a}$ if $c(\overrightarrow{v_{i_{t+1}} v_a}) = 1$.
7: Set $\omega_j(\mathbf{x}) = \omega_\mathbf{j}(\mathbf{x}) + \mathbf{1}$ and $\omega_i(\mathbf{x}) = \omega_\mathbf{i}(\mathbf{x}) - \mathbf{1}$.

Now we analyze the time complexity of Algorithm 2. In procedure SMOOTH$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$, there are at most n nodes on the path between v_i, v_j and up to n iterations in the "FOR" loop between line 4-7. Thus, the time complexity of SMOOTH$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$ is $O(n^2)$. From Theorem 5, it takes at most $O(m \cdot n^2)$ for Algorithm 2 to terminate. Constructing the virtual graph based on a feasible solution $\mathbf{x}$ could take time $O(m \cdot n^2)$. Thus, the total time complexity of smoothing algorithm is also $O(m \cdot n^2)$. If $n = o(\sqrt{m})$, then Algorithm 2 outperforms the best known max-flow algorithm by $\log^2 m$; when n is a constant the time complexity becomes $O(m)$ which is optimal.

A.3 Efficient Distributed Implementation

So far we have illustrated the basic idea of the Smoothing algorithm, which clearly can be implement in a distributed manner. In the remainder of the section, we will describe how this method can be implemented efficiently. Given an AN v_i, we say v_i is adjacent to AN v_j if there is a small sensor s_k in the cluster $\mathcal{C}_i \bigcap N(v_j)$. If v_i and v_j are not adjacent, then we define the distance between v_i and v_j as the smallest number of hops between them if we consider the adjacent graph of the ANs. For an AN v_i that is adjacent v_j, let ℓ be the largest non-negative integer such that $\frac{p_j(r\cdot\sum_{s_k\in S_M} x_{j,k}+\ell\cdot r)}{\mathcal{P}_j} < \omega_i(\mathbf{x})$. We define the *difference* of v_i and v_j as $\mathrm{dif}_{i,j}(\mathbf{x}) = \ell$. Based on the notation of difference, we have following localized algorithm.

Algorithm 4 Distributed Smoothing algorithm for AN v_i

Input: An initial assignment $\mathbf{x}$, $\gamma_{i,j}$ for every adjacent AN v_j that is the number of sensors that are in $N(v_i)\bigcap \mathcal{C}_j$.

1: When v_i receive an UPDATE-LEAVE or UPDATE-JOIN message from an adjacent AN v_j, it updates $\gamma_{i,j}$ if necessary.
2: Let v_j be one of v_i's adjacent AN with the maximum difference. Here, we break the tie arbitrarily.
3: **if** $\mathrm{dif}_{i,j}(\mathbf{x}) \geq \mathbf{1}$ **then**
4: Send a REQUEST message to AN v_j.
5: When v_j receives all REQUEST messages the ANs that adjacent to it, it sends out an ACK message to the AN that has the maximum weight and REJECT messages to all other ANs.
6: **if** v_i receives an ACK message from v_j **then**
7: Choose one SN, say s_k, in $\mathcal{C}_i \bigcap N(v_j)$. Set $x_{i,k} = 0$ and send SUCC message with the ID k to v_j.
8: Upate $\gamma_{i,j} = \gamma_{i,j} - 1$ and send the UPDATE-LEAVE message with ID k to all adjacent ANs.
9: When v_j receives the SUCC message from v_i with ID k, it first sets $x_{j,k} = 0$ and $\gamma(j,i) = \gamma(j,i) + 1$. After that it also sends UPDATE-JOIN message with ID k to all adjacent ANs.
Remark: Afterward, we also say that the small sensor s_k is *migrating* from cluster $\mathcal{C}_i$ to $\mathcal{C}_j$.

Regarding the distributed Algorithm 4, we have the following theorem.

Theorem 6: Algorithm 4 converges in at most $m\cdot n$ rounds and total message complexity is $O(n^2 \cdot m)$ if the ANs are homogeneous.

Proof: Given an assignment $\mathbf{x}$, we denote $\kappa_i(\mathbf{x})$ as the number of small sensors in ith largest cluster. Let $\Gamma_i(\mathbf{x}) = \sum_{\mathbf{j=1}}^{\mathbf{i}} \kappa_{\mathbf{j}}(\mathbf{x})$, and $\mathbf{x}^k$ be the assignment of sensors in round k. Considering $\Gamma^k = \sum_{i=1}^{n} \Gamma_i(\mathbf{x}^k)$. If there is a small sensor joining $\mathcal{C}_{i^k}$ and leaving $\mathcal{C}_{j^k}$ in round k, then $|S_{i^k}| > |S_{j^k}| + 1$ and $i^k < i^k$. Notice that after the small sensor migrating from cluster $\mathcal{C}_{i^k}$ to $\mathcal{C}_{j^k}$, $\Gamma_\ell(\mathbf{x}^k)$ decreases by 1 if $j < \ell \leq i$ and does not change otherwise. Thus, Γ^k decreases by 1 for every small sensor migrating. It is not difficult to observe that if there is no small sensor migrating in round k, then Algorithm 4 terminates. Since $\Gamma^1 < n \cdot m$, Algorithm 4 terminates in at most $n \cdot m$ rounds.

In every round, every AN sends only one REQUEST message and receives at most one REJECT message. Thus, there is at most $O(n)$ REQUEST and REJECT messages. It is also not difficult to observe that every AN sends at most one ACK messages. Thus, there are at most $O(n^2 \cdot m)$ REQUEST, ACK and REJECT messages in total. On the other hand, there is exact one UPDATE-LEAVE and UPDATE-JOIN message for every small sensor migrating. Thus, there are at most $O(n\cdot m)$ UPDATE-LEAVE and UPDATE-JOIN messages. Therefore, the overall message complexity is $O(n^2 \cdot m)$. ■

Notice that the message complexity analysis is very pessimistic. In simulations, it is much smaller than the worst case analysis. Observe that when Algorithm 4 terminates, it not necessarily gives an optimal solution. However, Algorithm 4 gives the best solution among all localized algorithms in which every AN can only know the information of its adjacent ANs. Furthermore, if we define the diameter of the network as the largest distance of the ANs, we have the following theorem (its proof is omitted due to space limit).

Theorem 7: When Algorithm 4 terminates, it gives an assignment with maximum cluster size at most $T \leq T^{\min} + D$ where D is the diameter of the network.

A.4 Dynamic updating

In wireless sensor networks, some old sensors may run out of battery and new sensors may be deployed from time to time. Furthermore, in certain applications, some sensors are able to move. Thus, it is necessary to consider how to dynamically update the cluster when the WSN changes. Here, we focus on the case when single small sensor leaves or joins the WSN. If there are more multiple sensors joining and/or leaving, it could be easily be reduced to the single sensor case. When an AN node runs out of power, the small sensors in its cluster has to be reassigned, *i.e.*, it is treated as the case that a group of small sensors joins the networks. We discuss how to dynamically update the cluster formation after the formation of the cluster according to the localized Algorithm 4 by cases.

Small sensor leaving. In this case, we assume the sensor s_k originally belongs to cluster $\mathcal{C}_j$. The weight of v_j is decreased by 1. Assume v_i is the AN with the largest weight that is adjacent v_j. If $\omega_j(\mathbf{x}) \geq \omega_{\mathbf{i}}(\mathbf{x})$ then v_j sends an UPDATE-LEAVE message with ID k to every adjacent v_a with ID k. Otherwise, we first choose any arbitrary small sensor $s_\ell \in \mathcal{C}_i \bigcap N(v_j)$. Remove s_ℓ from $\mathcal{C}_i$ and assign it to $\mathcal{C}_j$. v_j sends an UPDATE-LEAVE message with ID k and an UPDATE-JOIN message with ID ℓ to every adjacent AN; v_j sends an UPDATE-LEAVE message with ID ℓ to every adjacent AN.

Small sensor joining. In this case, we assume the sensor s_k joins the wireless sensor network. Find the AN v_i with the minimum weight such that $s_k \in N(v_i)$ and assign s_k to cluster $\mathcal{C}_i$. v_i sends out an UPDATE-JOIN with ID k to every adjacent AN.

For any individual small sensor leaving or joining, the clustering terminates in at most D rounds and the overall message complexity is at most $O(n \cdot D)$ where D is the diameter of the network. However, in simulations, we found that the message complexity and convergence rounds are both $O(1)$ most of the time.

B. Heterogeneous Application Nodes

In subsection III-A, we discuss how to form the clusters when both the small sensors and application nodes are homogeneous. However, in practice, such node homogeneity cannot always be guaranteed. For example, the initial onboard energy of ANs built by different vendors may not be proportional to the bitrate at which they generate, or the application nodes could be redeployed (e.g., new ANs join the system long after old ANs have been activated). Furthermore, two different application nodes may consume different energy to receive, process and send the information to the base station even given the same set of small sensors. Thus, it is more practical to assume the application nodes are heterogenous. In this paper, we consider the heterogeneity in two ways: the initial on board energy $\mathcal{P}$ and energy consumption function $p(x)$ where x is the sum of the rate of the small sensors in the cluster.

In this subsection, we redefine *weight* of a AN v_i for assignment $\mathbf{x}$ as $\omega_i(\mathbf{x}) = \frac{\mathbf{p_i}(\mathbf{r}\cdot\sum_{\mathbf{s_j}\in\mathbf{S_M}}\mathbf{x_{i,j}})}{\mathcal{P}_\mathbf{i}}$, where $\mathcal{P}_i$ is the initial onboard energy and $p_i(x)$ is energy consumption function. Here, the lifetime of the network is defined as

$$L = \max \min_{v_i \in V_N} \frac{\mathcal{P}_i}{p_i(r \cdot \sum_{s_j \in S_M} x_{i,j})} = \min \max_{v_i \in V_N} \omega_i(x).$$

Thus maximizing the lifetime is equivalent to minimizing the maximum weight over all ANs. Similar to the approach for the homogenous application node case, we formalize the problem as an Integer Programming as follows.

$$\min \max_{v_i \in V_N} \frac{p_i(r \cdot \sum_{s_j \in S_M} x_{i,j})}{\mathcal{P}_i} \tag{10}$$

Subject to constraint

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{11}$$

$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \tag{12}$$

$$\text{and } \sum_{v_i} x_{i,j} = 1, \forall s_j \tag{13}$$

B.1 Max-Flow Approach

Similar to the homogenous case, by adopting the traditional techniques, we transform the IP (1) into a new IP as follows:

$$\min T \tag{14}$$

Subject to constraint

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{15}$$

$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \tag{16}$$

$$\sum_{v_i} x_{i,j} = 1, \forall s_j; \tag{17}$$

$$\sum_{s_j \in S_M} x_{i,j} \leq k_i, \forall v_i \tag{18}$$

Here $k_i = \frac{p_i^{-1}(T \cdot \mathcal{P}_i)}{r}$. Then $\sum_{s_j \in S_M} x_{i,j} \leq k_i$ is equivalent to $\frac{p_i(r \cdot \sum_{s_j \in S_M} x_{i,j})}{\mathcal{P}_i} \leq T$ since $p_i()$ is assumed to be monotone non-decreasing.

Let $\mathbf{x}^{\min}$ be the solution to IP (14) and $T^{\min}$ be the minimum weight of the ANs. Unlike in the homogenous SN case, the value $T^{\min}$ could be any positive real number here. Thus, the simple binary search on T^{min} does not work. However, since there must exist an index i such that $\sum_{s_j \in S_M} x_{i,j} = k_i$, we can guess such index from $i = 1$ to n then perform a binary search on $\sum_{s_j \in S_M} x_{i,j}$. Therefore, we only need to decide whether there is a solution for the following Feasible System for a given T.

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \quad (19)$$

$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \quad (20)$$

$$\sum_{v_i} x_{i,j} = 1, \forall s_j; \quad (21)$$

$$\sum_{s_j \in S_M} x_{i,j} \leq k_i, \forall v_i \quad (22)$$

Here, we construct a network with s as the source and t as the sink as shown in Figure 1. There is a directional link $\overrightarrow{sv_i}$ $(1 \leq i \leq n)$ with capacity k_i, a directional link between $\overrightarrow{v_i s_j}$ with capacity 1 if $s_j \in N(v_i)$ and a directional link $\overrightarrow{s_j t}$ with capacity 1. Similar to Lemma 1 for homogenous case, we have the following lemma. The proof is straightforward and is omitted here.

Lemma 8: There is a solution to FS (19) for a given T if and only if the maximum flow of the network (1) is m, where m is the cardinality of S_M.

By guessing the T for $\log m \cdot n$ time, we can find the solution to Integer Programming (10) by solving $\log m \cdot n$ bipartite max-flow problems. Thus, the time complexity for Max-Flow approach is $O(n^2 \cdot m^{3/2} \log^2(m))$ which is very expensive and impractical.

B.2 Smoothing Algorithm

In this subsection we continue to show that our smoothing Algorithm 2 also applies to the heterogenous case with only minor modification.

Lemma 9: Let v_i be the AN with the largest weight and v_j be the AN with the lowest weight that is reachable by v_i in a feasible assignment $\mathbf{x}$. Then $\omega_j(\mathbf{x}) \leq \mathbf{T^{min}} \leq \omega_{\mathbf{i}}(\mathbf{x})$.

Algorithm 5 Smoothing algorithm for heterogenous ANs

Input: An Integer Programming (10).

Output: The solution to Integer Programming (10).

1: Find a feasible solution $\mathbf{x}$, *e.g.*, randomly assign every SN to a neighboring AN.
2: Construct a virtual graph $VG(\mathbf{x})$ based on $\mathbf{x}$ by applying Algorithm 1.
3: **repeat**
4: Choose any one of AN with the largest weight randomly, say v_i.
5: Define $\omega_k^+(\mathbf{x}) = \frac{\mathbf{p_k}(\mathbf{r} \cdot \sum_{\mathbf{s_j} \in \mathbf{S_M}} \mathbf{x_{k,j}} + \mathbf{r})}{\mathcal{P}_\mathbf{k}}$.
6: Find the AN v_j with the smallest $\omega_j^+(\mathbf{x})$ in $\mathcal{R}_i(\mathbf{x})$. If there are more than one such ANs, choose one randomly.
7: Apply procedure SMOOTH_HETE$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$ if $\omega_i(\mathbf{x}) > \omega_\mathbf{j}^+(\mathbf{x})$
8: **until** $\omega_i(\mathbf{x}) \leq \omega_\mathbf{j}^+(\mathbf{x})$

Algorithm 6 Procedure SMOOTH_HETE$(v_i, v_j, VG(\mathbf{x}), \mathbf{x})$

Input: A feasible solution $\mathbf{x}$ of FS (11) and its corresponding directed virtual graph $VG(\mathbf{x})$, a pair of nodes v_i and v_j.

1: Let $v_{i_0}(v_i)v_{i_1} \cdots v_{i_k}(v_j)$ be the path connecting v_i and v_j with the minimum number of hop. Here, $v_{i_0} = v_i$ and $v_{i_k} = v_j$.
2: **for** $t = 0$ to $k - 1$ **do**
3: Assume $x_{t,l} = 1$ and $s_\ell \in N(v_{t+1})$. Set $x_{t,l} = 0$ and $x_{t+1,l} = 1$.
4: **for** every v_a such that $s_\ell \in N(v_a)$ **do**
5: Update $c(v_{i_t}v_a) = c(v_{i_t}v_a) - 1$. Remove directed link $v_{i_t}(v_a)$ if $c(v_{i_t}v_a) = 0$.
6: **for** every v_b such that $s_\ell \in N(v_b)$ **do**
7: $c(v_{i_{t+1}}v_b) = c(v_{i_{t+1}}v_b) + 1$. Add a directed link $v_{i_{t+1}}v_b$ if $c(v_{i_{t+1}}v_b) = 1$.
8: Update $\omega_j(\mathbf{x}) = \omega_\mathbf{j}^+(\mathbf{x})$ and $\omega_i(\mathbf{x}) = \frac{\mathbf{p_i}(\mathbf{r} \cdot \sum_{\mathbf{s_j} \in \mathbf{S_M}} \mathbf{x_{i,j}} - \mathbf{r})}{\mathcal{P}_\mathbf{i}}$.

Theorem 10: Algorithm 5 outputs a solution of IP (14) and terminates after m iterations.

The proof of this theorem is omitted here due to space limit. Surprisingly, the time complexity of Algorithm 5 is also $O(m \cdot n^2)$, which is exactly the same as in the homogenous case. This reduces the time complexity by an order of $\sqrt{m}\log^2 m$ and more importantly, Algorithm 4 also works for the heterogenous case with only modification of the definition of difference. However, we only have the following conjecture for the convergence and message complexity of localized smoothing algorithm. It is an open and interesting problem to either prove or disprove the following conjecture.

Conjecture 1: Algorithm 4 terminates after at most $n \cdot m$ rounds and the total message complexity $O(n^2 \cdot m)$ when the ANs are heterogenous.

IV. Heterogeneous Small Sensors

Usually in WSNs, several different kinds of sensors cooperate together to fulfill some certain goals. Some sensors may generate data at a higher rate than others do, *e.g.*, the visual sensors have a bit-rate that is much higher than the bit-rate generated by a temperature sensor. Even in scenarios when all small sensors are of same type, sometimes sensors located at different locations may need to sample the data at a different time interval. Thus, it is more reasonable to assume that in a WSN different type of sensors produce different bit-rates.

By assuming that every small sensor has its own data rate r_i, we formalize the problem of maximizing the lifetime as an Integer Programming as follows:

$$\min \max_{v_i \in V_N} \frac{p_i(\sum_{s_j \in S_M} r_j \cdot x_{i,j})}{\mathcal{P}_i} \tag{23}$$

Subject to constraint

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{24}$$
$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \tag{25}$$
$$\sum_{v_i} x_{i,j} = 1, \forall s_j \tag{26}$$

Unlike the case for homogenous SNs in which we can find the solution that maximizes the lifetime exactly, Theorem 11 shows that it is NP-Hard to find the solution to IP (23).

Theorem 11: We can not find the solution of IP (23) in polynomial time if $P \neq NP$.

Proof: We consider the special case when application nodes are homogeneous. In this case, since $p_i(x) = p(x)$ is increasing, it is equivalent to minimizing the maximum $\sum_{s_j \in S_M} r_j \cdot x_{i,j}$ subject to constraints (24). If every AN v_i satisfies that $N(v_i) = S_M - v_i$, then the problem becomes the traditional job scheduling problem [25], [26], which is known to be NP-Hard. This finishes our proof. ■

Since solving IP (23) is NP-hard, we will present an algorithm approximating the optimal solution by borrowing some ideas from job scheduling [27], [28]. Again we transform IP (23) into Integer Programming (27) as follows.

$$\min T \tag{27}$$

Subject to constraints

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{28}$$
$$x_{i,j} \in \{0,1\}, \forall s_j, \forall v_i; \tag{29}$$
$$\sum_{v_i} x_{i,j} = 1, \forall s_j; \tag{30}$$
$$\sum_{s_j \in S_M} r_j \cdot x_{i,j} \leq k_i, \forall v_i \tag{31}$$

Here $k_i = p_i^{-1}(\mathcal{P}_i \cdot T)$. Let $\mathbf{x^{min}}$ be the solution to IP (27) and $T^{\min}$ be the $\min T$ under solution $\mathbf{x^{min}}$. It is easy to observe that $\mathbf{x_{ij}^{min}}$ satisfies the following constraint.

$$x_{i,j} = 0 \qquad \forall v_i, \forall s_j \quad r_j > k_i \tag{32}$$

If we relax the constraint $x_{i,j} \in \{0,1\}$ we obtain a Linear Programming as follows.

$$\min T \tag{33}$$

Subject to constraints

$$x_{i,j} = 0, \forall v_i, \forall s_j \notin N(v_i); \tag{34}$$
$$x_{i,j} \geq 0, \forall v_i, \forall s_j; \tag{35}$$
$$\sum_{v_i} x_{i,j} = 1, \forall s_j; \tag{36}$$
$$\sum_{s_j \in S_M} r_j \cdot x_{i,j} \leq k_i; \forall v_i \tag{37}$$

Let $x^\star$ be the solution to LP (33) plus constraint 32 and $T^\star$ be the value of $\min T$ under solution $x^\star$. Then $T^\star \leq T^{\min}$. By binary search on $T^\star$ we can find the

solution $x^\star$ to LP (33) plus constraint 32 in polynomial time. Furthermore, we can find a solution $x^\star$ that has some special properties. For a small sensor s_j, if there exists an AN v_i such that $0 < x_{i,j} < 1$, we call s_j is fractionally assigned to cluster $\mathcal{C}_i$. We construct a graph with vertex $V_N \bigcup S_M$ and add an edge $s_j v_i$ if and only if $0 < x_{i,j} < 1$. Obviously, it is a bipartite graph and it is generally known [28], [29] that we can transform the solution $x^\star$ to another solution x^* such that its corresponding bipartite graph is composed of forests with(or without) a line. Remember that every node in S_M connects to at least two nodes in A_N, thus there is a matching such that every node in S_M can connect to a distinct node in A_N. The final solution is to assign s_j to cluster with head v_i if one of the following two conditions holds:

- $x^*_{ij} = 1$
- s_j is connected with v_i in the matching.

In this section, to make sure that we can guarantee the performance of the above job-scheduling based approach, we add one more requirement for the power consumption function p_i. We assume that the marginal cost of $p_i(x)$ is not increasing, *i.e.*, for $x_1 \geq x_2$, $p_i(x_1 + \delta) - p_i(x_1) \leq p_i(x_2 + \delta) - p_i(x_2)$. This assumption is almost universally satisfied. If this assumption is not satisfied, we can construct examples to show that the above approach (based on job scheduling) cannot provide any theoretical performance guarantees, although its practical performance may still be good.

Theorem 12: Our job scheduling based method produces a cluster formation such that the lifetime of the WSN is at least $\frac{1}{2}$ of the maximum lifetime of the WSN.

Proof: For any AN v_i, there is at most one s_ℓ connecting to v_i in the matching and $r_\ell \leq k_i$. Thus, $\sum_{s_j \in S_M} r_j \cdot x_{i,j} = \sum_{s_j \in S_M \setminus r_\ell} r_j \cdot x_{i,j} + r_\ell \cdot x_{il} \leq 2k_i$. Therefore, $p_i(\sum_{s_j \in S_M} r_j \cdot x_{i,j}) \leq p_i(2k_i) \leq 2p_i(k_i) = 2\mathcal{P}_i \cdot T^\star \leq 2\mathcal{P}_i \cdot T^{\min}$. This finishes the proof. ■

V. Other Issues

In previous sections, we proposed several methods for clustering formation for two-tiered wireless sensor networks. Although we have addressed the heterogeneous application nodes and heterogeneous sensor nodes, there are still lots of interesting questions left for further research.

Multihop Network of SNs: In previous sections, we assumed that every SN can reach at least one AN directly. However, this assumption may not be true in practice because SN's transmission range is usually smaller than the AN's. Thus, the SNs formed a routing tree that could be reached by some ANs, which complicated the case. However, if we define $\mathcal{C}_i$ as all the SNs that belong to the tree that is rooted at v_i and $N(v_i)$ as all the SNs that can communicate or belong to some SNs in $\mathcal{C}_i$, our localized smoothing algorithm still has a pretty good performance in enhancing the life time although it does not achieve the optimal. Our conjecture is that it is impossible to find the optimal solution in this case unless $P = NP$ and it could be an interesting open question to find the some algorithms with good performance guarantee that integrates both the routing tree formation of SNs and the clustering of these SNs.

Multihop Network of ANs: Remember that in order to simply our analysis, we assumed that the application nodes send the packet directly to the base station. In the real world, it is possible that the application nodes form a multi-hop relay network to relay the packets to the base station. Different application nodes may be in different positions in this relay network according to the base station. This makes the problem even complicated, *i.e.*, all these ANs that can reach base station directly are expected to relay *all* the packets generated in the network. This makes the ANs closer to BS easier to be run out of power than far-away ANs. Notice that we did not pose any special restriction on the energy consumption function of an application node. In other words, we assumed a generic energy consumption function for ANs. If we can define $\mathcal{C}_i$ as all the SNs whose traffic need to be relayed via v_i and $N(v_i)$ as all the SNs that can communicate or belong to some SNs in $\mathcal{C}_i$, our localized smoothing algorithm still works when the relay network of the ANs are known in advance. However, some algorithms other than smoothing algorithms are needed to decide the ANs relay network formation when it is not known in advance and we list it as one of our possible future research directions.

Static vs. Dynamic Cluster: In previous sections, we discussed how to maximize the lifetime of the WSNs by forming some logical clusters. We required that the cluster formation is permanent, *i.e.*, the clusters do not change during the whole lifetime of WSN. Obviously, the lifetime of the WSN could be further improved if we allow a dynamic cluster formation,

i.e., the clusters will adapt to the remaining energy level of the ANs. Notice that the improvement of lifetime is not guaranteed since there is always overhead to dynamically update the cluster. For example, when the power consumption function satisfies a certain property, *e.g.*, it is linear, the lifetime is improved by at most a very small fraction. Thus, we only consider the static case in our simulation afterwards.

VI. Performance Studies

We conducted extensive simulations to study the performance of different algorithms and approaches introduced in this paper. As mentioned earlier in this paper the network is composed of application nodes and sensor nodes. Sensor nodes communicate with application nodes only and application nodes communicate with sensor nodes and other applications nodes. Here we mainly study the case with heterogeneous application nodes and homogeneous sensor nodes. Each sensor node has a transmission range and is able to communicate with application nodes within its transmission range and also each sensor has a sensing range and is able to monitor the area within its sensing range. We assume that the ANs have the same properties but are different in the initial power and the power consumption rate for sending a unit amount of data to the base-station. In addition, we assume that the transmission range and the sensing range of all sensor nodes are the same. Each application node consumes *one unit* of battery power to serve one sensor node for one day.

A. Simulation Environment

We randomly placed 2000 sensor nodes in a $800feet \times 800feet$ square region, the transmission range of each sensor node is set to $50feet$ and the sensing range is set to $10feet$. Notice that, the small sensors are typically randomly placed without any strategic locations. To guarantee that the area of a $a \times a$ square feet region is covered with high probability by n small sensors with sensing range s, we should have the following relation $n\pi(\frac{s}{a})^2 \simeq \ln n$. We found that the small sensors with these settings in our simulations will cover the majority part of the region. Then we put a different number of application nodes, from 150 to 300 (with incremental 25) and measured the network lifetime based on several different definitions of lifetimes. In addition, the initial battery power of each sensor node is a random value between 100 units and 200 units. Hereafter we call a small sensor node that has power remaining and has at least one alive application node in their transmission region as an *alive sensor*.

A.1 Other Heuristics

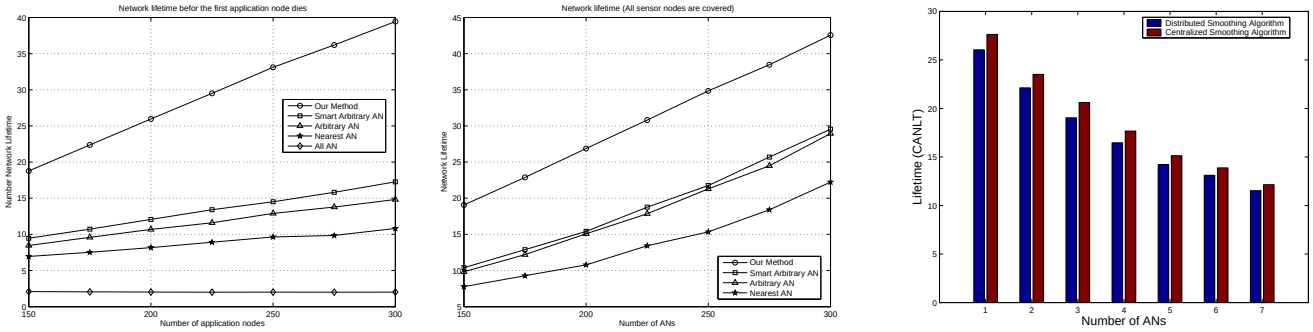
To study the performance of our smoothing Algorithm 2, we compare it with other heuristics listed below: (1) *[-Nearest]* Each sensor node is assigned to the nearest AN. (2) *[-Arbitrary]* Each sensor node is *randomly* assigned to one of the application nodes inside its transmission range. (3) *[-Smart-Arbitrary]* In this method, each sensor node is *randomly* assigned to one of the application nodes that is inside the sensor node's transmission range. The probability of a SN s_j assigned to a neighboring AN v_i is the ratio of the remaining power of v_i over the total remaining power of all neighboring ANs of this SN s_j. (4) *[-All]* Here, each sensor node is assigned to *all* the application nodes that are inside the sensor node's transmission range. This is clearly the worst method. Thus we will not compare with this method in most simulations.

B. Simulation Results

B.1 Lifetime

In this subsection, we compare the lifetime of four different methods under two different definitions of lifetimes: CANLT, FCLT. For α-CANLT and β-FCLT lifetime, when α and β are smaller, *i.e.*, $\alpha = 30\%$; our smoothing algorithm performances much worst than most the other methods. The reason is that our smoothing algorithm makes its best effort to maximize the lifetime when $\alpha = 100\%$ and $\beta = 100\%$. Thus, for α or β that is different from 100%, we can apply the following simple technique: send $1-\alpha$ or $1-\beta$ percentage of the ANs to sleep in a round-robin manner. This technique can apply to all five methods mentioned above for the sake of fairness. It is not difficult to observe that the lifetime of all four methods increase by a fix percentage ($\frac{1}{1-\alpha}$ or $\frac{1}{1-\beta}$). Thus, it suffices to use the CANLT and FCLT only. See Section II for definitions. Figure 2 (a), (b) show the lifetime of different assignment methods under lifetime definition CANLT, FCLT respectively. We generate 100 random WSNs and all results are the average over the performance of these 100 WSNs.

As can be seen, the network lifetime increases almost linearly with the number of application nodes



(a) CANLT (b) FCLT Localized and Centralized Algorithm

Fig. 2. Comparison of lifetime for different methods

available initially for all methods, except the simplest ALL approach that does not perform any logic cluster at all. A striking observation is that, as we expected, our smoothing based method outperforms all other tree methods under all four definitions of lifetimes regardless of the density of the application nodes. In all simulations, we found that our method generally outperforms the other methods by almost 100%. In other words, the network lifetime is almost *doubled* when our method is used to form the cluster.

We also compare the performance of the Centralized Smoothing Algorithm 2 (CSA) and Localized Smoothing Algorithm 4 (LSA). We fixed the number of the ANs to 50 and varies the number of SNs from 200 to 500. Figure 3 (c) shows difference of the lifetime (CANLT) between CSA and LSA, and it is not difficult to observe that the lifetime of LSA and CSA only differs about 5% to 8%. This corroborates our theoretical analysis and we will only compare the lifetime of CSA with other four methods afterwards.

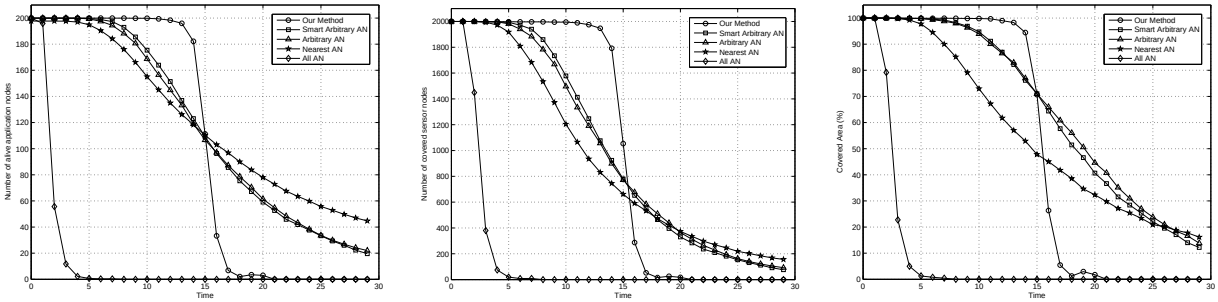
B.2 Load Balancing

As mentioned in Section III-B, for heterogeneous application nodes case, application nodes have different initial battery powers, and the objective of the Algorithm 2 is to assign less sensor nodes to application nodes that have lower remaining battery power and more sensor nodes to application nodes that have higher battery power. To see how good the load balancing of our algorithm is, we run simulation for the networks with 150 application nodes till all application nodes die. As can be seen in Figure 3, our algorithm achieves a very good load balancing meaning that all application nodes consume energy at a rate proportional to their initial battery power and then they all die together. The result for number of alive sensor nodes and also the percentage of coverage area are basically the same as shown in Figure 3 (b) and (c).

B.3 Area Coverage

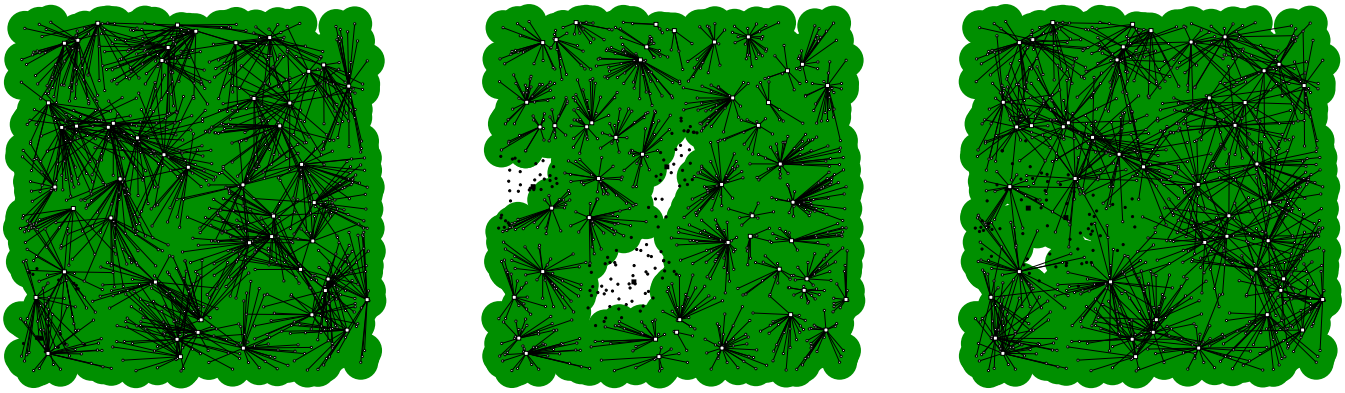
To further study the area covered by sensor nodes we build a two tiered sensor network with 50 application nodes, 1000 sensor nodes randomly placed in a $250feet \times 250feet$ region. The transmission range of all sensor nodes is set to $50feet$ and the sensing range is set to $10feet$. To serve 1000 small sensor nodes for one month, $30,000$ units of energy is needed. We set the initial battery power of each application node to a random value between 400 and 800 units. Each application node has on average 600 units of battery power. Figure 4 depicts the sensor node assignment and coverage of the network at various date by various methods.

In the method (called ALL) that do not perform logic cluster all application nodes die after 9 days. As can be seen in Figure 4, after 20 days, the "Nearest AN" and "Arbitrary AN" methods fail to keep all the application nodes alive and hence the area is not fully covered. Till the end of day 29, our method maintains to keep all the application nodes alive and hence the full coverage. Our algorithm guarantees a balanced power consumption of the application nodes and maintain full coverage. For the lifetime defined based on coverage area, our method improves the lifetime of the network by almost 20% for this networks (composed of 1000 sensors and 50 application nodes in a $250feet \times 250feet$ area). Our previous simulations (see Figure 2 (c) and (d)) reported a larger improvement if we had more ANs.

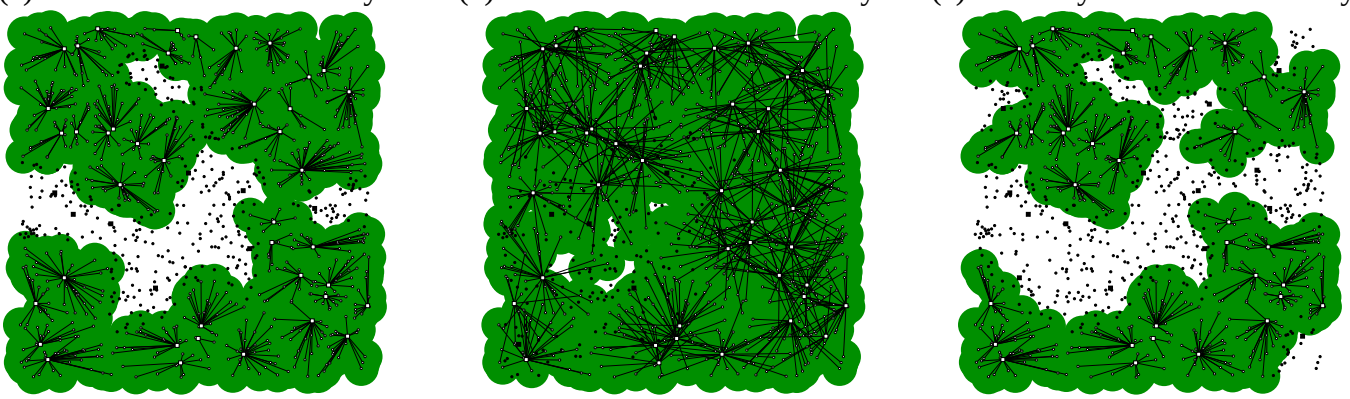


(a) Number of ANs alive (b) Number of SNs alive (c) Area percentage covered

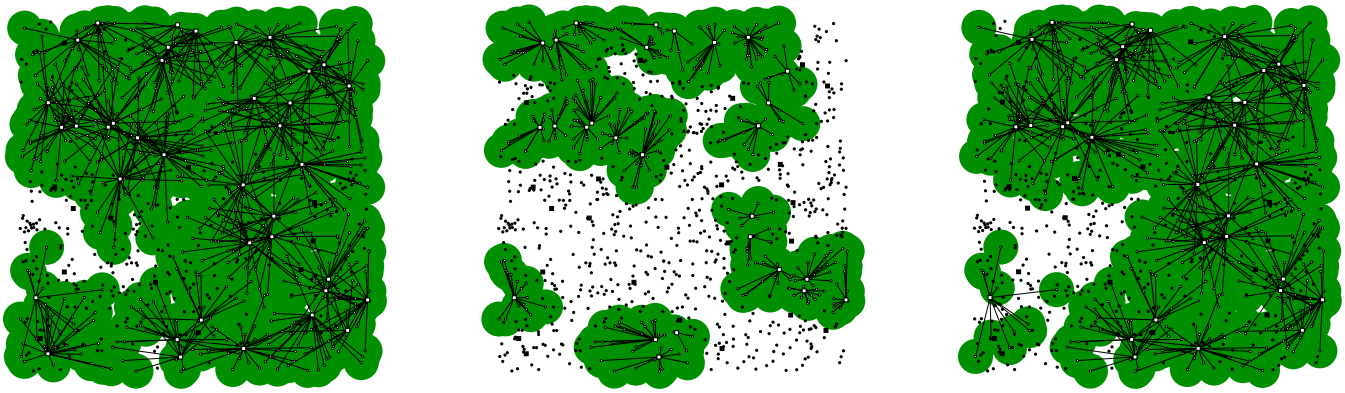
Fig. 3. Comparison for different methods



(a) Our Methods After 29 days (b) Nearest Method After 15 days (c) Arbitrary method after 15 days



(a) Nearest Methods After 20 days (b) Arbitrary Method After 20 days (c) Nearest method after 25 days



(d) Closest Methods After 25 days (e) Nearest Method After 29 days (f) Arbitrary method after 29 days

Fig. 4. Coverage in two-tiered networks.

VII. Conclusion

In this paper, we studied a generic two-tiered wireless sensor networks (WSN) composed of small sensor nodes (SNs), more powerful application nodes (ANs), and base-stations (BSs). We especially studied how to organize the WSN to form logic clusters to maximize the lifetime of the networks. By using CANLT as the definition of the lifetime for the WSN, we considered the scenarios when the application nodes are homogeneous or heterogeneous, the sensor nodes are homogeneous or heterogeneous respectively. When the sensors are homogeneous, we give optimal algorithms to maximize the lifetime of the networks; when the sensors are heterogeneous, we give a 2-approximation algorithm that produces a network whose lifetime is within $1/2$ of the optimum. We also showed that it is NP-hard to find the optimum cluster formation. Our theoretical results are corroborated by extensive simulation studies. Our simulations show that our algorithms actually perform much better not only theoretically but also for randomly generated WSNs.

There are some interesting important questions that have not been addressed in this paper. For example, maximizing the lifetime under other definitions deserves further study. It is also important to study how the schematically improve the lifetime by integrating the routing tree formation of small sensors, the routing tree formation of application nodes and the attachment of sensor routing trees to application nodes. Once again, we point out that our work is only an opening but not concluding work of this direction which deserve more research efforts in the future.

References

[1] http://www.xbow.com/Products/products.htm, "Mica2 and mica2dot," Tech. Rep.

[2] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications*, 2003.

[3] F. Constantin V. Crespi G. Cybenko J. Aslam, Z. Butler and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. 2003, pp. 150–161, ACM Press.

[4] A. Ledeczi M. Maroti, G. Simon and J. Sztipanovits, "Shooter localization in urban terrain," *Computer*, vol. 37, no. 8, pp. 60–61, 2004.

[5] J. Polastre R. Szewczyk A. Mainwaring, D. Culler and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. 2002, pp. 88–97, ACM Press.

[6] J. Polastre D. Culler R. Szewczyk, A. Mainwaring, "An analysis of a large scale habitat monitoring application," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. 2004, ACM Press.

[7] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1694 – 1701, Nov 1981.

[8] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *Proceedings of ITS*, 1994.

[9] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom)*, 2003.

[10] L. Cai Y. Shi J. Pan, Y. T. Hou and S. X. Shen, "Topology control for wireless sensor networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. 2003, pp. 286–299, ACM Press.

[11] L. Cai Y. Shi J. Pan, Y. T. Hou and S. X. Shen, "Optimal base-station locations in two-tiered wireless sensor networks," in *IEEE TRANSACTIONS ON MOBILE COMPUTING*, To appear.

[12] Wei Ye, John Heidemann, and Deborah Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, 2004.

[13] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, NY, USA, 2003, pp. 181–192, ACM Press.

[14] Tijs van Dam and Koen Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, NY, USA, 2003, pp. 171–180, ACM Press.

[15] J. Haartsen, "The bluetooth radio system," *IEEE Personal Communications*, vol. 7, pp. 28–36, 2000.

[16] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, Washington, DC, USA, 2000, p. 8020, IEEE Computer Society.

[17] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, 2001.

[18] J. Wieselthier A. Ephremides and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," in *Proceedings of IEEE*, 1987, vol. 75, pp. 56–73.

[19] A. K. Parekh, "Selecting routers in ad-hoc wireless networks," in *In Proceeding ITS*, 1994.

[20] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," in *IEEE Journal on Selected Areas in Communications*, Sep. 1997, vol. 15, pp. 1265–1275.

[21] S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99)*. 1999, p. 310, IEEE Computer Society.

[22] Thai H. P. Vuong Alan D. Amis, Ravi Prakash and Dung T, "Max-min d-cluster formation in wireless ad hoc networks," in *IEEE INFOCOM*, 2000.

[23] P. Monti C.F. Chiasserini, I. Chlamtac and A. Nucci, "Energy efficient design of wireless ad hoc networks," in *In Proceedings of European Wireless*, Feb. 2002.

[24] J.E. HOPCROFT and R.M KARP, "$n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, pp. 225–231, 1973.

[25] R. Graham, "Bounds for multiprocessing timing anomalies," in *SIAM Journal on Applied Mathematics*, 1969, vol. 17.

[26] D. S. Hochbaum and D. B. Shmoys, "Using dual approximation algorithms for scheduling problems theoretical and practical results," *J. ACM*, vol. 34, no. 1, pp. 144–162, 1987.

[27] E. Tardos J. K. Lenstra, D. B. Shmoys, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, no. 3, pp. 259–271, 1990.

[28] K. Jansen and L. Porkolab, "Improved approximation schemes for scheduling unrelated parallel machines," in *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*. 1999, pp. 408–417, ACM Press.

[29] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, no. 3, pp. 461–474, 1993.