UNIVERSITY AT STONY BROOK

CEAS Technical Report 815

CENTRALIZED ENERGY EFFICIENT CONTROL OF CLUSTERS IN TWO-LAYER SENSOR NETWORKS

Z. Zhang, M. Ma, Y. Yang

September 16, 2004

Centralized Energy Efficient Control in Clusters of Two-Layer Sensor Networks

Zhenghao Zhang, Ming Ma and Yuanyuan Yang
Department of Electrical & Computer Engineering
State University of New York, Stony Brook, NY 11794, USA

Abstract—**In this paper we study heterogeneous sensor network with two layers. We deploy two types of nodes to the network: the basic sensor node and cluster head node. The powerful cluster head node organizes the much simpler sensor nodes around it into a cluster, and acts as cluster head. Such a two-layer network has better scalability, lower overall cost, and longer life. We focus on energy efficient control within clusters to prolong network life. The cluster head use polling to get data from sensors in its cluster, in stead of letting sensors send data randomly, such that less energies are wasted. We will first describe the network flow formalization for finding the load balanced data relaying paths. Then we will discuss the problem of finding a contention-free polling schedule that uses minimum time. We will show that this problem is NP-hard, and then give a fast on-line algorithm. We also conducted simulations on ns-2.**

Index Terms: Sensor networks, Cluster, Polling.

I. Introduction and Background

The use of wireless sensors networks will enable a wide variety of applications, including environmental monitoring, medical systems, etc. In a sensor network, a large number of sensors are deployed over an large area, with each sensor capable of collecting data and also sending data via wireless channels. To ensure correct data transfer, sensor have to be organized into a robust multi hop wireless network. This poses several challenges to network design. First, the network has to be scalable, since there can be thousands of sensors in the network. Second, to reduce the cost, sensors should be made as simple as possible, and as a result, sensors should only have some very simple functionalities. Third, the power supplies of sensors are limited and cannot be replaced, therefore the network has to be energy efficient.

In general, these challenges cannot be easily met. We can tackle the first challenge by introducing hierarchies: the network is partitioned into into clusters, and sensors need only communicate within clusters while the inter-cluster communication is done by the cluster head. [3] [11] gave protocols for cluster forming and cluster head selection. However, in these methods, sensors become more complicated since every sensor could be elected as cluster head and thus more transmitting and storage capabilities are needed for each sensor.

We note that for some applications, for example ground temperature monitoring, where the main job of sensors is to gather information and send it to outside observer, we can take this idea one step further. We can deploy two different kinds of sensor nodes to the network: the basic sensor node and the cluster head node, as shown in Fig.1. The basic sensor node is the majority of the sensor nodes, which is cheap and has relatively simple functionalities, such as data sampling and simple packet forwarding, and has a limited power supply and a small transmission power. The cluster head node is much powerful than basic nodes: it has much more computational capabilities and more power supplies and a much larger transmission power. A cluster head node organizes basic sensor nodes around it into a cluster, and acts as the cluster head. The basic sensor node collects the data and sends it to its cluster head. The cluster head node sends data to the outside observer or a nearby head node. The cluster head sends messages in a much larger power and its message can be heard by all sensors in the cluster. The transmission power of basic sensor much smaller, and to reach the cluster head, the packet has to be relayed by other sensors. The advantage of such a heterogeneous network is that majority of sensor nodes can be made very simple and very cheap, thus the overall cost of the network can be greatly reduced. In the rest of the paper we will refer to the basic sensor node as sensor, and the cluster head node as cluster head.

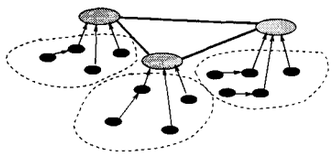

Fig. 1. A two-layer heterogeneous sensor network. The large nodes are the cluster heads. The small nodes are the basic sensor nodes.

We can regard this type of network has a two-layer network. The first layer is the individual cluster. Above the first layer, in the second layer, the cluster heads can organize themselves into another network, in which they can exchange information or send data to the outside observer. The second layer is essentially a static wireless ad hoc network, and many works in the literature can be applied to it. Question still remains as to how to design the first layer, or the in-cluster control, in an energy-efficient way, to prolong the life of the network.

Most commonly, sensors are assumed to send data packets randomly. From the point of view of saving energy, this has several disadvantages. First, energies could be wasted in collisions: several sensors may decide to send packets at the same time and all these packets have to be retransmitted. Second, energy can be wasted by overhearing: sensors may have to decode packets not destined for them. Third, in the effort to avoid collision, sensors may have to use control packets to coordinate with each other, which also consumes a significant amount of energy. Lastly and also most importantly, when sensors send packets in a random manner, a lot of energy will be wasted in idle listen-

The research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085 and CCR-0207999.

ing. This is because sensors have to constantly monitor the radio channel for possible data packets destined for them, and the power consumption for idle listening is at the same magnitude as sending and receiving [8].

To overcome these problems, [8] introduces MAC layer control that reduces collisions and overhearing, and also let sensors periodically wake up and sleep to reduce idle listening, for applications when the data generating rate is low. However, the waste of energy is still significant. For example, for idle listening, in their experiments sensors still have to be awake for nearly 24% of the time. We note that the energy is wasted because of the lack of coordination among sensors: sensors do not know when other sensor will do and have to "be prepared" for most of the time. This is inevitable in a distributed system. However, in a heterogeneous network, with the presence of the cluster head, sensors can be fully controlled by the cluster head and be fully coordinated, and all possible sources of energy wasting mentioned above can be eliminated. The cluster head can send out messages to tell sensors to send packets which will not cause collision. The cluster head can also tell sensors when to receive, thus no overhearing will occur. There will no control packets from sensors. After enough data has been collected, the cluster head can tell all sensors to enter the sleep mode until next wake up time. Other advantages of this centralized framework is that it needs less functionalities of the sensors. Such a centralized framework is unconventional, but is not impossible since with some slight overhead, the cluster head can gather enough information about all sensors in its cluster. Of course in this case the cluster head has to take up more responsibilities.

We will give methods for the cluster head to organize the sensors in a robust and energy efficient way. The goal is to maximize the battery lives of sensors in the cluster. We use *polling* to collect data from the sensors, i.e., the cluster head sends a message to the sensor it wants to hear from, and only the sensor received this message will start sending. The main problems that need to be solved are the follows. First, there should be a relaying path by which the packet from a sensor can be relayed to the cluster head. The relaying paths for sensors are good if the the transmission loads of sensors are balanced, i.e., there is no sensor relaying too many packets while other sensors relaying too few. We will describe the solution to this problem, the network flow formalization which has appeared in [4] [10], and also give our implementation methods for it. Second, we need to find a polling schedule. A good polling schedule should not cause any collision and should finish as soon as possible. We will show that finding such a schedule is NP-hard, and give a fast greedy algorithm for it. Note that the difference of the greedy algorithm from polling protocols like 802.11 PCF and Bluetooth is that the latter are for single hop networks while the former is for multi hop networks. We will then address some implementation issues and give some possible improvements. We will also use ns-2 simulations to show the performance of our method.

II. Cluster Operations

We target at applications where the data generation rate is low. Under the control of the cluster head, sensors will sleep for most part of the time, and wake up only for a short while to send data.

In a duty cycle, the sensors works in the following way. First, sensors wake up, entering the active listening mode, at the time specified by the cluster head before they went to asleep last time. They will wait for the message broadcast by the cluster head, indicating the beginning of a duty cycle. After the cluster head broadcast this message, each sensor sends a short packet back, acknowledging the cluster head, also informing the cluster head of the number of packets it has to send. Based on this, the cluster head will find a polling schedule, and controls the sensors in a time slotted manner, where time slot is the length of time for one data packet transmission. At the beginning of a time slot, the cluster head will broadcast a polling message, indicating which sensors can start to send their packets, also which sensors should receive packets. All other sensors will be enter the idle listening mode for one time slot. Before the next time slot starts, all sensors will listen to the radio channel, waiting for the next polling message. After hearing the polling message, sensors that are polled will send their packets, and sensors that had received a packet in the previous time slot will relay the packet to the next hop neighbor. And so on, until all packets have been received. After this, the cluster head broadcasts a message, set all sensors to the sleep mode, and also telling them the next wake-up time. We can see that sensors works in a way like a pipelined system, and the polling message is like the clock. To reduce the length of a time slot, we do not use link level acknowledgment. If a packet is lost, the cluster head will poll the sensor again.

III. Optimal Routing

For this and the next section, we will assume the network has been partitioned into clusters and focus only in the operations in an individual cluster. The first problem needs to be solved is to find paths for each sensor, by which its packet is relayed to the cluster head. To do this, the cluster head will need to know with which sensors a sensor can reliably communicate with, or the *connectivity patterns* of the cluster. We will describe methods to find the connectivity patterns in later sections, for now, we assume the cluster head has this knowledge.

An obvious solution to the problem is to build a tree, in which the cluster head is the tree root, and each sensor relays all the packets coming from its dependents to its parent. However, the tree is not the optimal structure, since examples can be found in which some sensor relays too many packets while others relay very few. A result of this is that the sensor with too many relaying burden dies soon, which may disconnect the network. In an optimal structure, the load should be balanced.

Define the *transmission load*, or simply *load* of a sensor in a duty cycle as the number of packets it has to send out, including its own packets and packets from other sensors it has to relay. As an example, consider the cluster shown in Fig.2, in which an edge connects two sensors if they can communicate with each other. Suppose each sensor has one packet to send to the cluster head t. Due to the connection pattern, S_1 has to relay for S_3, S_2 has to relay for S_4, and S_5 has to relay for S_6 to S_8. S_5 can send packet to either S_1 or S_2. If we use a tree, S_5 can only send to one of them, and as a result, either S_1 or S_2 will have load 6, while the other's is only 2. If we split the traffic out of

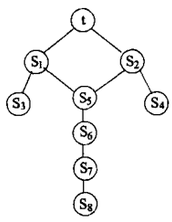

Fig. 2. The connection pattern of a cluster, in which tree is not the optimal for relaying data packets.

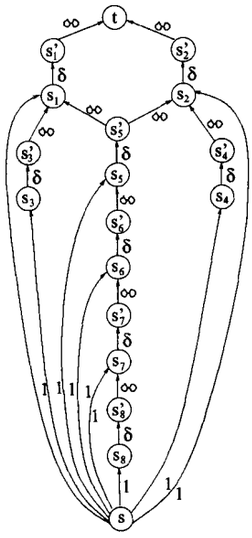

Fig. 3. Network flow formalization for Fig.2.

S_5 into two, with 2 packets going to S_1 and two packets going to S_3, the load of S_1 and S_2 will both be 4. In fact 4 is also the maximum load among all sensors.

A. Maximum Flow Formalization

The problem of finding optimal packet relaying paths has been studied independently by [4] [10]. It can be formalized as a min-max problem: find a routing strategy such that the maximum load of sensors is minimum.

This problem can be solved in polynomial time by formalizing into a network flow problem. At first we consider the case when each sensor has exactly one packet to send in a duty cycle. Suppose there are n sensors in the cluster, and first suppose in this duty cycle each sensor has exactly one packet to send. Let the head node be the sink t. Each sensor, say S_1, corresponds to two nodes s_1 and s_1' in the digraph of the network flow problem, where s_1 is the "input" node and s_1' is the "output" node. There is an arc from s_1 to s_1' with capacity δ, where δ is a parameter and actually controls the load of sensor S_1. If sensor S_2 is in the transmission range of S_1, there is an arc from s_1' to input node s_2 with infinite capacity. Similarly, if cluster head can be reached by S_1, there is an arc from s_1' to t with infinite capacity. There is an additional node, the source s. There is an arc from s to s_i with unit capacity for all $1 \leq i \leq n$. For example, the network flow formalization for Fig.2 is shown in Fig.3.

To find the optimal relaying paths, we can start with a small δ, then run the Ford- Fulkerson algorithm. If there is a flow of value n, where n is the number of sensors, we are done. Otherwise, increment δ by one and run the Ford-Fulkerson algorithm again. Note that the current flow must be a valid flow after δ is incremented, therefore when re-running the Ford-Fulkerson algorithm, we do not have to start from a zero flow. Also note that after incrementing δ, if from every sensor there is a route to the cluster head, the flow must also be increased by one. Therefore the algorithm will terminate in $O(n^3)$ time, and when terminated, it will either return a set of relaying paths that are optimal, or reporting that the some sensors are isolated.

It is easy to extend the method to the case when the number of packets from sensors are different. Suppose in a duty cycle, the number of packets from sensor S_i is p_i. We can let the capacity of arc s to s_i be p_i, and use the same algorithm to find the optimal relaying paths. The Ford-Fulkerson algorithm runs in $O(Vn^2)$ time in this case, where $V = \sum_{i=1}^{n} p_i$.

However, computing the optimal routing strategy in every duty cycle may not be practical since running the maximum flow algorithms can be time consuming. Besides, we note that we actually do not have to balance the load per duty cycle: all that is required is that the load is balanced over a long time period. Thus, we can find an optimal routing strategy based on the *average traffic intensity*, which is the average number of packets generated by sensors observed in a long time period.

B. Practical Issues

B.1 Source Routing

To the best of our knowledge, all previous works focus on the theoretical aspects of the problem, that is, how to find the optimal relaying paths. Practically, after find the optimal relaying paths, we should find a way to make traffic move along it. This problem was realized in [10] but not solved. One solution to this is to use *source routing*: Each sensor remembers the relaying path from it to the cluster head, and adds it to the header of the packet it sends. Sensor that receives this packet will then relay it to the sensor that appears next in the relaying path. This will ensure that traffic will flow according to the optimal relaying paths. However, this will also add length to the data packets and will thus waste energy. Equivalently, we can let each sensor remember a one-hop routing table for all its dependents, where a dependent of sensor S is a sensor whose relaying path involves S. For example, if the relaying paths of S_6 and S_7 in Fig.2 are $S_6 - S_5 - S_1 - t$ and $S_7 - S_6 - S_5 - S_2 - t$, respectively, we can let S_5 remember to forward S_6's packet to S_1 and forward S_7's packet to S_2. Since only one hop information is needed for each dependent, this will not need too much of storage space. Also note that a relaying path is simply an augmenting path of the Ford-Fulkerson algorithm, therefore the paths for source routing can be found when running the algorithm.

B.2 Multiple Paths Rotation

Note that when traffic intensities of sensors are different, a sensor may have several relaying paths. For example, if on average 3 packets are generated by sensor S_1 in a duty cycle, the maximum flow algorithm may find two paths, path one carrying

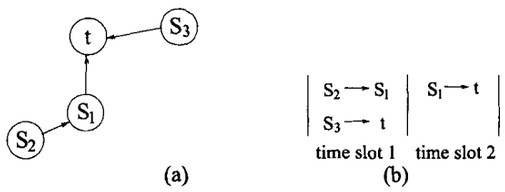

Fig. 4. (a). A simple cluster. S_1, S_2 and S_3 have 0, 1, and 1 packet to send, respectively. $S_2 \rightarrow S_1$ and $S_3 \rightarrow t$ do not cause collision. (b). The polling schedule.

2 units of flow and path 2 carrying 1 unit. To simplify controlling, in a duty cycle, we would like sensors send packets only on one path. Thus, to balance the load, we can let sensors send packets on these paths alternatively, in proportions to the units of flows the paths carry. In the example, we can let S_1 send packets along path 1 for two duty cycles and along path 1 for one duty cycle.

IV. Polling in Multi Hop Sensor Networks

As said earlier, the cluster head uses polling to get data from sensors. The major difference between single hop polling and multi hop polling is that in multi hop clusters, the cluster head can poll more than one sensors at the same time if their packet transmissions do not cause collision. As a simple example, consider the cluster shown in Fig.4(a). Suppose S_1, S_2 and S_3 have 0, 1, and 1 packet to send, respectively. Because the packet from S_2 has to use two hops to reach t, if we only poll one sensor at a time, i.e., poll a sensor only after the packet from the sensor polled previously has been received, 3 time slots are needed. However, if transmission $S_2 \rightarrow S_1$ and $S_3 \rightarrow t$ do not cause collision, we can poll S_2 and S_3 together at the first time slot. At the second time slot, S_1 relays the packet from S_2 to t. Thus the polling finishes in only 2 time slots, as shown in Fig.4(b).

We will show in this section that the problem of finding an optimal collision free schedule which is the schedule by which the polling finishes in the shortest time is NP-hard, and then give a fast greedy algorithm for it. We will first assume the relaying paths of sensors are fixed. At the end of this section, we will explain that it is also hard to jointly solve routing and scheduling problem.

A. Interference Patterns

We can see that to find a collision free schedule, it is crucial to know the *interference patterns* in the cluster. That is, the cluster head has to know whether a group of transmissions are contention free, or *compatible*. A model that is widely used in performance analysis is the geometrical model, or the protocol model in [15], which assumes that two sensors can communicate with each other if the distance between them is less than a constant r, and two transmissions, say $S_1' \rightarrow S_1$, $S_2' \rightarrow S_2$ are compatible if and only if the distance between S_1 and S_2' is less than $(1+\Delta)r$, as well as between S_2 and S_1'. This model is convenient for performance analysis, but we cannot use it in determining polling schedules for the following reasons.

First, in reality, the covering area of a sensor, which is the area in which other sensors can correctly receive message from it, may well not be a disc. A number of factors, for example reflecting objects and multi path fading, can make the covering area very oddly shaped and might not even be convex, especially in urban areas [1]. The second reason is even more profound. The geometrical model suggests that if a group of transmissions are compatible if and only if they are pairwise compatible. However, in reality, a necessary condition for a group of transmissions being compatible is that the accumulated interferences at any receiver are not too large. For example, as in Fig.5, suppose three transmissions, $S_1' \rightarrow S_1$, $S_2' \rightarrow S_2$ and $S_3' \rightarrow S_3$ are pairwise compatible. Let $P(S_i', S_j)$ be the the signal power from S_i' present at S_j. The three transmissions being pairwise compatible implies that any single $P(S_i', S_j)$ is not large for $i \neq j$. However, when the three transmissions occur at the same time, if $P(S_1', S_2) + P(S_3', S_2)$ is too large, transmission $S_2' \rightarrow S_2$ will still fail.

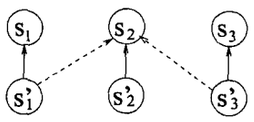

Fig. 5. Pairwise compatibility dose not guarantee compatibility.

To make network robust to any environments, we do not make any assumption about the covering areas and interference patterns and treat them as arbitrary. The cluster head gets the connectivity patterns and interference patterns by testing the involved transmissions and sensors, and we will describe the method in detail later. Note that testing the interference patterns for all possible groups of transmissions needs exponential time. Thus, we will assume that the cluster head only knows the compatibility of all no more than M transmissions, where M is a small positive integer, like 2 or 3.

B. The Optimal Polling Schedule is Hard to Find

We will first consider the problem of how to find an optimal polling schedule. We will assume that the cluster head has known the number of packets each sensor plan to send. We will first assume the sensors do not store a received packet in its buffer: a received packet is forwarded to the next hop immediately in the next time slot.

We say a sensor is of level i along its path to the head there are i hops. If all sensors are in level 1, the scheduling reduces to single hop polling and is trivial. We will call the problem when some sensors have level more than 1 the MULTI HOP POLLING PROBLEM and refer to it as MHP problem.

To see MHP problem is NP-hard, first we consider a special structure called "two level star with relaying only first level", which is referred to later as TSRF. A TSRF is a tree, with the root being the cluster head, and each branch is consisted of 2 sensors and one of these two sensors is connected to the head. Fig.6(a) shows a TSRF with 5 branches. More specifically, let t be the root, sensors denoted by $S_1, S_2, S_3, \ldots$, are connected to t, or are in the first level, sensors denoted by $S_1', S_2', S_3', \ldots$ are in the second level, with S_1' is only connected to S_1, S_2' is only connected to S_2, etc. Each sensor in the second level has exactly one packet to send, and sensors in the first level has no packet to send. The relaying path for the packet generated in sensor S_1' is $S_1' - S_1 - t$. Some transmissions, say, $S_1' \rightarrow S_1$ and $S_3 \rightarrow t$

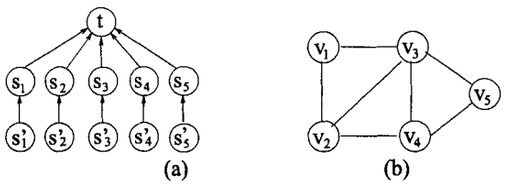

Fig. 6. (a) A TSRF with 5 branches. (b). An arbitrary graph with 5 vertices.

can occur at the same time if they do not cause collision.

The problem is: given a TSRF and interference pattern, does there exist a schedule by which all sensors can send their packets to the head by time k? We will call it TSRF POLLING PROBLEM and will sometimes refer to it as TSRFP.

Lemma 1: The TSRF POLLING PROBLEM is NP-complete.

Proof. It is clear that this problem belongs to NP. To see its NP-completeness, we reduce the Hamiltonian Path problem to it.

Given any instance of the undirected Hamiltonian Path problem, we construct an instance of TSRFP problem as follows. Denote the graph of the Hamiltonian Path problem as G, and denote the vertices in G as $v_1, v_2, v_3, \ldots$. Each node in the graph corresponds to a branch of the TSRF. For example, the TSRF for the graph shown in Fig.6(b) is Fig.6(a), where v_1 corresponds to branch $S'_1 - S_1 - t$, v_2 corresponds to branch $S'_2 - S_2 - t$, etc. If two vertices are adjacent, for example v_1 and v_2, transmissions $S_1 \rightarrow t$ and $S'_2 \rightarrow S_2$ are compatible, transmissions $S_2 \rightarrow t$ and $S'_1 \rightarrow S_1$ are also compatible. Otherwise, these two pairs of transmissions are not compatible. k is set to be $n+1$, where n is the number of vertices in G.

Note that in a schedule that needs only $n+1$ time slots, the transmissions must be back to back: all sensors in the second level must start to send in consecutive time slots with no "pause" in between. Sensor S'_i and S'_j can start in consecutive time slots if and only if $S_i \rightarrow t$ and $S'_j \rightarrow S_j$ do not cause collision. In this case, by our construction, there is an edge from v_i to v_j in G. Therefore, this schedule determines a Hamiltonian path in G. ■

Since any algorithm that solves the MULTI HOP POLLING PROBLEM can solve the TSRFP, thus we have

Theorem 1: The MULTI HOP POLLING PROBLEM is NP-hard.

B.1 Reality Check

Note that in the proof we constructed a TSRF with interference patterns resembling the connection patterns of an arbitrary graph. A natural question is whether the interference pattern in a TSRF can indeed be arbitrary.

We verify this with the interference model, which is the physical model in [15], as follows. We assume that in the TSRF, two transmissions, $S'_i \rightarrow S_i$ and $S_j \rightarrow t$, do not interfere with each other if and only the following two inequalities hold:

$$\begin{cases} P(S'_i,t)\gamma & < P(S_j,t) \\ P(S_j,S_i)\gamma & < P(S'_i,S_i) \end{cases}$$

where γ is a constant. Note that as illustrated in [1], whether a packet can be successfully received or not is not completely determined by signal power, so this model is still a simplified version of reality. However, we still use it because it makes senses and also because of the lack of other good models.

In wireless environments, $P(S_j,S_i)$ can be arbitrary. Thus, we can assume $P(S_j,t)$ be larger than any $P(S'_i,t)\gamma$ for all i and j. We can let $P(S'_i,S_i) = \eta$ for all i. Then if there is an edge between v_i and v_j, we can let $P(S_j,S_i)\gamma < \eta$. Otherwise, we can let $P(S_j,S_i)\gamma > \eta$. This will produce the desired interferences.

C. The Scheduling Problem Is Still NP-hard when Packets Can Be Delayed

Note that so far we assume that once a sensor received a packet it will forward it to the next sensor immediately. In other words, packets will not be delayed in intermediate sensors along the path. This makes the controlling simple, and reduces the size of memories in sensors. Moreover, as we will show soon, there is also no benefit to allow packets be delayed: the scheduling problem is still NP-hard even if we allow packets be temporarily stored in intermediate sensors.

Theorem 2: The MULTI HOP POLLING PROBLEM is NP-hard even when packets can be delayed.

Proof. Again consider the TSRF. Suppose there is an algorithm that answers the question: given a TSRF and interference pattern, does there exist a schedule by which all sensors can send their packets to the head by time k, when allowing the delaying of the packets? Again we can let $k = n+1$, where n is the number of branches in the TSRF.

The claim is that if there is a schedule that all sensors' packets have been received by the cluster head by time $n+1$, the same schedule must also be a legitimate schedule for the case when packets are not delayed. To see this, note that in this schedule, in every time slot after the first, there must be a sensor in the first level relying packets to the cluster head. Suppose at the first slot, S'_i sends a packet to S_i. Then in the second time slot, S_i must send this packet to the cluster head. Suppose in the second time slot S'_j is sending its packet to S_j. Then in the third time slot, S_j must be sending S'_j's packet to the cluster head. The same argument can be carried on and as a result, no packet is delayed in this schedule. Therefore, the same algorithm can be used to answer the TSRFP problem with no packet delay. ■

D. The Scheduling Problem Is Still NP-hard when Each Sensor Has Exactly One Packet to Send

In some applications, sensors sample data periodically and generate exactly one packet per duty cycle. Therefore, we have to consider a special case of the MULTI HOP POLLING PROBLEM, in which each sensor has exactly one packet to send. We will call it EXACT ONE PACKET MULTI HOP POLLING PROBLEM, and abbreviate it as X1MHP. This problem, unfortunately, is still NP-hard.

Theorem 3: The EXACT ONE PACKET MULTI HOP POLLING PROBLEM is NP-hard.

Proof. We prove this by reducing the TSFRP to it. Given any instance of the TSFRP, we construct an instance of the X1MHP as follows.

For each branch in TSFRP instance, say $S'_1 - S_1 - t$, we add another auxiliary branch, as shown in Fig.7. The relaying path

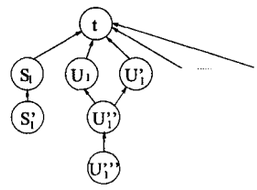

Fig. 7. The auxiliary branch for branch $S_1' - S_1 - t$.

for U_1''' is $U_1''' - U_1'' - U_1' - t$. The relaying path for U_1'' is $U_1'' - U_1 - t$. Transmission $U_1'' \rightarrow U_1'$ is compatible with $S_1 \rightarrow t$. Other transmissions in this auxiliary branch is not compatible with any other transmissions.

For any optimal solution to the X1MHP problem, suppose sensor U_1''' start to send its packet at time slot α. At time slot $\alpha+1$, U_1'' will relay this packet to U_1'. If sensor S_1 is not sending its packet to t at time slot $\alpha+1$, we can "move" it to this time slot, since $U_1'' \rightarrow U_1'$ is compatible with $S_1 \rightarrow t$, and the resulting schedule should still be optimal. After pairing them up, we move them to the beginning of the schedule, that is, we let U_1''' start to send at time slot 1, and let S_1 start to send at time slot 2. The resulting schedule should still be optimal since $U_1''' \rightarrow U_1''$ and $U_1' \rightarrow t$ are not compatible with any other transmissions, and therefore are not occurring at the same time with any other transmissions in the optimal schedule. For the same reason, we can also move U_1'', U_1' and U_1 to the front, or to let them start to send at time slot 4, 6, 7, respectively. Then we pair up S_2 with its auxiliary branch and move it to the beginning of the schedule and so on.

As a result, the optimal schedule will consist of two parts, one in the beginning for sensor S_1, S_2, ... and sensors in their auxiliary branches, followed by a schedule only for sensors S_1', S_2', ... The second part must be an optimal schedule for the TSFRP instance. ■

Using similar arguments as in Theorem 2, it can be shown that

Theorem 4: The EXACT ONE PACKET MULTI HOP POLLING PROBLEM is NP-hard even when packets can be delayed.

D.1 Reality Check

We can first let the received power for transmissions in the auxiliary branches be very weak. Consider the first auxiliary branch. To make a transmission $U_1 \rightarrow t$ and $U_1' \rightarrow t$ incompatible with all other transmissions, we need only raise $P(S,t)$ for all sensor S not in the first level. To make $U_1''' \rightarrow U_1''$, $U_1'' \rightarrow U_1$ no compatible with any other transmissions, we can raise $P(S,U_1'')$ and $P(S,U_1)$ for all other sensors. To make $U_1'' \rightarrow U_1'$ compatible with only with $S_1 \rightarrow t$, we can let $P(S,U_1')$ be relatively large for all sensors other than S_1.

E. The On-line Polling Algorithm

In this section we focus on finding an algorithm that gives sub-optimal contention-free polling schedules. Another requirement of this algorithm is that is should be able to run *on-line*, since it has to be able to deal with packet loss. Note that in wireless communications, success packet delivery is not guaranteed. Since the cluster head knows which time slot a sensor start to send its packet and the hop count from the sensor to it, it knows exactly which time slot it should expect to receive that packet. Therefore, the cluster head can find out when a packet is lost and once this happens, it can simply poll the sensor again. Therefore the scheduling algorithm should be on-line, since new polling requests may come (actually the old ones come again) when the polling is going on.

The NP-hardness of the problem and the on-line requirement left us no other choices but to adopt a very simple algorithm. We will refer each packet as a *polling request*, or simply *request*. At the beginning, each request is active. When a request has been added to the schedule, it becomes idle. On the time slot when the packet should have been received by the cluster head, if it is not received, the request will become active again. Otherwise, it will be deleted. The algorithm will only consider active polling requests. Since the cluster head knows only all possible combinations of no more than M transmissions, at any time, the algorithm will only allow no more than M concurrent transmissions.

Before the first time slot, the schedule is empty. The algorithm scans through the requests according to an arbitrarily predetermined order, and adds a request to the schedule if it dose not cause contention with the existing ones (if started at the first time slot). After maximum number of requests have been added to the schedule, or after M request have been added, the algorithm halts. The cluster head will tell sensors appeared in the schedule to send their packets in the first time slot. In the following time slots the algorithm will work in exactly the same way, until all packets have been received.

To see whether a packet can be sent at time slot $\alpha+1$, suppose the hop count is c, we need to check whether the c transmissions involved will collide with the transmissions in the existing schedule from time slot $\alpha+1$ to $\alpha+c$. Suppose there are m requests in the schedule. For each time slot, there are totally $\sum_{i=1}^{m} \binom{m}{i} = 2^m - 1$ possible groups of transmissions that need to be checked. Note that $m \leq M-1$. Thus it would require $O(c2^M)$ time. Suppose there are totally N requests and the maximum hop count is C. The work in each time slot can be finished in $O(NC2^M)$ time. Note that although there is an exponential term, 2^M, in the bound, M is fixed and is really small, thus the algorithm is still a linear time algorithm to the input size.

F. Joint Routing and Scheduling is also Hard

So far, in this section, we have considered the scheduling problem when the relaying paths of sensors have been given. However, we may want to find the optimal solution while not fixing the routing paths. This will give us more flexibilities and better solutions. However, without much difficulty, we will soon show that this problem is also NP-hard, and that is why we choose to break the bigger problem into two sub problems and solve them one by one.

Note that the life of a sensor is determined by its transmission load and the polling time of the cluster. We can assume the life of S_i is reversely proportional to the *power consumption rate* $\alpha R_i + \beta T$, where R_i is the transmission load of the S_i and T is the polling time, and α and β are constants. The problem is:

TABLE 1
POLLING ALGORITHM

Input : The polling requests. **Output:** Polling schedule.

while requests are not all deleted
  if a packet is received
  Delete the request for this packet
  end if
  if a packet that is expected to arrive is not received
  Set the request for this packet to active.
  end if
  while less M requests in the schedule at this time slot
  Add an active request to the schedule to
  current time slot if it does not cause collision.
  Mark the request as idle.
  end while
  wait until next time slot.
end while

Given the connection pattern and the interference pattern of a cluster, find relaying paths and a polling schedule such that the maximum power consumption rate is minimum. We will call it the JOINT MULTI HOP ROUTING AND POLLING PROBLEM, or the JMHRP problem.

JMHRP is clearly also NP-hard since it must be no easier than the TSFRP problem: for any TSFRP instance, we can construct an instance of the JMHRP problem, with same interference patterns, while removing all transmission links except $S_i' - S_i$ and $S_i - t$ for all i. That is, only allowing data transmissions within the branches. (Note that in the TSFRP problem, there could be links between sensors in different branches.) Clearly, if there is an efficient algorithm for the JMHRP problem, the same algorithm can also be used to give answer to the TSFRP problem.

V. IMPLEMENTATION ISSUES AND IMPROVEMENTS

A. Cluster Forming

Initially, the sensor network should be partitioned into clusters. The cluster head should know which sensors are in its cluster and sensors should also know which clusters they belong to. This is *Cluster Forming*, and is a research topic of its own [5] [3] [11].

In this paper we assume that the clusters have been formed and focus on in-cluster controls. However, one possible way of cluster forming is to let cluster heads compute the Voronoi diagrams and let sensors in the same Voronoi cell belong to the same cluster. This requires sensors know their locations, which is a legitimate assumption since for many applications, information will be meaningless without knowing the locations of where they come from.

After cluster has formed, the cluster head should know which sensors belong to its cluster. We can first let the cluster head discover sensors that can directly communicate with it, then let these sensors find sensors that are two-hop away, then use the new sensors to find sensor three-hop away, until no new sensors are discovered. Each sensor can remember the first sensor that discovered it as the parent of it, who will be in charge of forwarding its packets to the cluster head. Note this is only to set up a temporary data relaying path.

B. Knowing The Connectivity

The cluster head should first find out the connectivity patterns in its cluster, i.e., We can let sensors to broadcast in turn, then poll each sensor to see which sensor it has heard from. a sensor can send packets to which sensors. This will need $O(n)$ time where n is the number of sensors, and is only needed at the initialization phase. After the connectivity pattern is known the cluster head can find the optimal routing path using the maximum flow algorithm described earlier.

C. Knowing The Interference Pattern

After the optimal routing path is found, to determine the polling schedule, the cluster head needs to know the interference pattern. As said earlier, we get this by testing each group of no more than M transmissions needed in the optimal routing path. We can poll each pair of sensors involved in the transmissions at the same time, then poll the sensors that are supposed to receive the packets in turn. This will finish in $O(n^M)$ time.

D. Acknowledgments Collecting

As said earlier, after cluster head has received all the packets, sensors enters the sleep mode. Ideally, later, sensors should wake up at exactly the same time. However, due to possible drift of clocks in sensors, this might not be the case. Therefore, to make sure that all sensors have waken up before data transmission, the cluster head should broadcast a inquiry message. All sensors received this message should then reply an acknowledgment. Only after receiving acknowledgments from all sensors will the cluster head start polling. In addition, along with this acknowledgment message is that sensor can inform the cluster head of the number of packets it plan to send. Note that when collecting the acknowledgments, the length of a time slot can be made much shorter, since the acknowledgment packet is much shorter than data packets.

One possible way doing this is to let cluster head poll the sensors, as in the case when each sensor has exactly one packet to send. However, there are more efficient ways. Note that since the network is multi hop, some sensors will have to relay the ack packet for other sensors, and while relaying, sensors can add in their own ack to this packet. Thus, only one ack packet is needed for sensors along a path to the cluster head, and as a result, only the sensor at the beginning of the path needs to be polled.

Using similar arguments as in Section IV-F, we can see that problem of finishing the acknowledgments collecting process in minimum time is also NP-hard. To solve it, we can break it into two subproblems: first, find a set of paths that covers all sensors; then, find schedule to finish polling sensors appears at the beginning of the paths in minimum time. The second problem can be solved by the algorithm for multi hop polling described before.

To solve the first subproblem, we use the relaying paths for the data packets as "candidate" paths, and choose among them a set of paths that covers all sensors with minimum total number of hop counts. This problem can be solved by regarding the sensors as elements in a whole set, and regarding the paths as

subsets. Each subset has a cost equal to it hop count. Then find a group of subsets that covers the whole set with minimum total cost. This problem is exactly the Weighted Set Cover Problem, and is also NP-hard, but a fast greedy algorithm can be used to give sub optimal solutions. This greedy algorithm iteratively chooses a subset that has the minimum covering cost, where covering cost of a subset is defined as the ratio of the cost of it over the number of uncovered elements it contains.

E. Removing Inter-Cluster Interference

In a sensor network, there are many clusters, each controlled by its own cluster head. With the centralized polling scheme, sensors in the same cluster do not interfere with each other. However, at the boundaries of the clusters, sensors belonging to different clusters may still do, if their cluster heads decide to poll them at the same time. In this section we will describe methods to deal with this situation.

The simplest way is to allow data transmission in only one cluster at a time. A token can be rotated among the cluster heads, and only the cluster head that captures the token can start data transmission in its cluster. It works for the case when the number of clusters are not large, and the data transmission within a cluster can be finished in relatively short time periods as compared to a duty cycle.

Another more efficient way is to let sensors in nearby clusters operate in different radio channels, such that they can send packets at the same time without causing collision. The problem then needs to be solved is: how many radio channels are needed, and how to assign then to the clusters?

Regarding a radio channel as a color, this problem is equivalent to giving adjacent clusters different colors. This is the famous 4-color problem for planar graphs and we know that 4 colors, or 4 radio channels will be sufficient. To find the coloring, a cluster head can be elected to run a centralized algorithm for the coloring problem.

However, algorithms for coloring planar graphs with 4 colors or even 5 colors may be too complicated. On the other hand, there is a simple centralized recursive algorithm that colors a planar graph with at most 6 colors, using the property that in a planar graph, there must be a vertex with degree no more than 5. From a practical point of view, this algorithm is more attractive since although using 6 colors or 6 channels will probably need more bandwidth than using 5 or 4 colors, but is not particularly harder or more expensive for making sensors since each sensor will operate on exactly one channel. Another advantage of this algorithm is that it can be easily changed into a distributed algorithm, for circumstances where distributed control is preferred. For details of the algorithm the please refer to [12].

F. Dividing the cluster into sectors

Note that during the data transmission period, if a sensor will not be involved in transmissions occurred later, it can enter the sleep mode immediately to save energy. A sensor can enter the sleep mode in this way if and only if all the packets generated by itself and all the packets it has to relay have been correctly received by the cluster head. Therefore, if we want to set a sensor to sleep mode sooner, we can poll it and its dependents first. Also note that by the time a sensor can enter the sleep mode, all its dependents can also enter the sleep mode. These observations lead us to considering dividing a cluster into a number of sectors, where a sector is consisted of a group of sensors and each sensor has a routing path to the cluster head. We can let each sector wake up and do data transmission in turn, such that sensors need to be awake for shorter time periods. Also note that once the sectors are determined, routing and polling mechanisms for clusters described in previous sections can be readily applied to sectors.

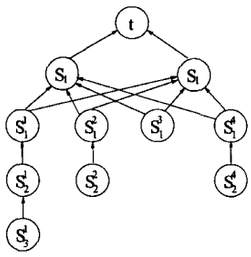

Fig. 8. The construction of the CPAR problem for set $\{3, 2, 1, 2\}$.

Managing sensors by sectors has yet another advantage. The number of sensors in a sector can be far less than the number of sensors in the entire cluster. This makes finding and storing the interference patterns much simpler, since far less number of groups of transmissions need to be tested and stored. For example, if we divide a cluster with 80 sensors into 8 sectors with each sector having 10 sensors, if $M = 3$, we need to test 1320 groups, which is far less then 85320 otherwise.

Question still remains as to how to optimally partition a cluster into sectors, such that the maximum power consumption rate of sensors is minimum. However, the partition problem will be NP-hard to find by this criterion since finding the optimal polling schedule is NP-hard. Another criterion that makes sense is: we say a partition is optimal if the maximum *pseudo power consumption rate* is minimum, where the pseudo power consumption rate of S_i is defined as $\alpha R_i + \beta' L_i$ where L_i is the number of sensors of the sector S_i belongs to, since the polling time will be roughly proportional to the number of sensors involved. Unfortunately, even when reduced to this, the optimal partitioning problem is still NP-hard.

Problem: Given a cluster and its connection patterns, does there exist a partition such that $\max\{\alpha R_i + \beta' L_i\} \leq \Theta$? We will call it CLUSTER PARTITION PROBLEM and will refer to it as CPAR problem.

Theorem 5: The CLUSTER PARTITION PROBLEM is NP-complete.

Proof. We can reduce it to the PARTITION PROBLEM, in which a set of positive integers $\{B_1, B_2, ...\}$ are given, and asks for partitioning the integers into two subsets and the sum of the numbers in two subsets should be equal.

Given any instance of the PARTITION PROBLEM, we construct an instance of the CPAR problem as follows. We draw two sensor nodes, S_1 and S_2, which is connected to the cluster head t. For the i_{th} number, written as B_i, we draw B_i sensors, denoted as S_1^i to S_δ^i, where $\delta = B_i$. These δ sensors are a branch, with S_j^i is connected only to S_{j-1}^i for $1 < j \leq \delta$. S_1^i is connected to both S_1 and S_2. Θ is set to be

$(n/2+1)(\alpha+\beta')$, where $n=\sum B_i$. For example, the construction for set $\{3,2,1,2\}$ is shown in Fig.8.

Note that the cluster can at most be divided into two sectors. Also note that S_1 and S_2 will collectively carry the load of n, thus if there is a partition satisfying Θ, the cluster must be divided into 2 sectors, since otherwise one of the pseudo power consumption rate of S_1 and S_2 will exceed Θ. The claim follows since aside S_1 or S_2, there must be exactly $n/2$ sensors in each sector, which is a solution to the PARTITION PROBLEM. ■

VI. Simulations

VII. Conclusions

In this paper we studied heterogeneous sensor network with two layers, where the network is partitioned into clusters, and a powerful cluster head controls all sensors in its cluster. We focused on energy efficient designed of clusters to prolong network life. We used polling to get data from sensors in stead of letting sensors send data randomly, such that less energies are wasted. We first described the network flow formalization for finding the load balanced data relaying paths. Then we will discuss the problem of finding a contention-free polling schedule that uses minimum time. We will showed that the problem of finding a contention-free polling schedule that uses minimum time is NP-hard, and then gave a fast on-line algorithm. We also conducted simulations on ns-2.

References

[1] D. Aguayo, et. al, "Link-level measurements from an 802.11b mesh network," *in Proceedings of ACM SIGCOMM 2004*, Portland, Aug. 2004.

[2] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," *in Proceedings of ACM/IEEE MOBICOM 2001*, Rome, Italy, June 2001.

[3] W. Rabiner, et. al, "Energy efficient communication protocols for wireless microsensor networks," *in Proceedings of the Hawaii International Conference on System Sciences*, Jan. 2000.

[4] J.H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *in Proceedings of IEEE/ACM INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[5] Alan Amis, et. al, "Max-min D-cluster formation in wireless ad hoc networks," *in Proceedings of IEEE/ACM INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[6] Y.-D. Lin and Y.-C. Hsu, "Multihop cellular: a new architecture for wireless communications," *in Proceedings of IEEE/ACM INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[7] S. Banerjee and S. KhullerA "Clustering scheme for hierarchical control in multi-hop wireless networks," *in Proceedings of IEEE/ACM INFOCOM 2001*, Anchorage, April 2001.

[8] W. Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," *in Proceedings of IEEE/ACM INFOCOM 2002*, New York, June 2002.

[9] V. Kawadia and P. Kumar, "Power control and clustering in ad hoc networks," *in Proceedings of IEEE/ACM INFOCOM 2003*, San Francisco, April 2003.

[10] A. Bogdanov, et. al , "Power-aware base station positioning for sensor networks," *in Proceedings of IEEE/ACM INFOCOM 2004*, Hong Kong, China, March 2004.

[11] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," *in Proceedings of IEEE/ACM INFOCOM 2004*, Hong Kong, China, March 2004.

[12] D. B. West, "Introduction to graph theory," *Prentice-Hall*, 1996.

[13] X. Hong, M. Gerla, H. Wang and L. Clare, "Load balanced, energy-Aware communications for Mars sensor networks," *In Proceedings of IEEE Aerospace 2002*, Bigsky, Mar. 2002.

[14] W. Hu, N. Bulusu and S. Jha, "A Communication paradigm for hybrid sensor/actuator networks," *To appear in Proceedings of the 15th Annual IEEE Conference on Personal, Indoor and Mobile Radio Communication (PIMRC 2004)*, Barcelona, Spain, Dec 2004.

[15] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388-404, March 2000.

[16] C. Florens, M. Franceschetti and R. J. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1110-1120, Aug. 2004.