

Optimal Energy-Balanced Algorithm for Selection in a Single Hop Sensor Network

Mitali Singh Viktor K. Prasanna
Department of Computer Science
University of Southern California
Los Angeles, CA, 90089.
Email: {mitalisi, prasanna}@usc.edu

Abstract—Sensor networks are being used to implement a large set of applications that require distributed, collaborative computations. Selection is an important kernel in several sensor applications, particularly those involving speech and image data processing. In this paper we present an energy and time optimal algorithm for the selection problem in a single hop wireless network. The algorithm also has the property that it is *energy-balanced*. This implies that all sensors dissipate asymptotically equal amount of energy. Uniform energy dissipation is desirable as it enables the network to remain fully functional for the maximum time. We demonstrate that in a single hop, single channel network of n randomly distributed sensors, selection can be performed in $O(n)$ time and $O(n)$ energy, with no sensor being awake for more than $O(1)$ time steps. We extend our results for a p -channel network, where $p \leq n^{1-\epsilon}$ and $0 < \epsilon \leq 1$. We show that selection can be performed in time $O(n/p)$ and energy $O(n)$ with no sensor being awake for more than $O(1)$ time steps.

I. INTRODUCTION

Wireless sensor networks are being used in several applications such as automatic target recognition, traffic monitoring, and hazard detection, among others. These networks can be considered as large scale dynamically configurable, distributed systems, where autonomous sensors collaborate among themselves to achieve a larger objective. Sensor networks are often deployed to monitor remote areas, where replacement of sensors is difficult. This makes energy a precious resource for these networks as the functionality of the network is limited by the battery power of the sensors.

Selection is an important algorithm used in several sensor applications, and sensor network management algorithms. The selection problem is defined as follows, given n sensors with one data element each, we want to find the sensor with the i^{th} smallest (largest) data element. Note that the problem of finding the median, or finding the sensors with i smallest elements are instances of the selection problem. Once the sensor with the data element of rank i is determined, it broadcasts this value. All sensors with larger data elements switch off and the remaining are the sensors with i smallest elements.

Several sensor applications, especially those involving speech and image data processing, perform rank selection. Median filters and rank based filters find extensive use in several nonlinear signal processing applications as well as data

acquisition and fault tolerant systems. Multiple signal aggregation, maximum likelihood estimation, online calibration, pixel sorting are some other problems that use rank selection.

Sensor networks are often heterogeneous. Low power sensors (trip wires) monitor the field continuously to detect an event. High power sensors are put to sleep till some event is detected that requires more processing. The trip wires sense the environment and deduce that an event of “interest” has occurred when an input of a chosen rank exceeds a pre-specified threshold. Online calibration schemes utilize rank based analysis of data to reject signals with extreme values. An interesting application is diagnostic X-Ray, where sensors placed behind the exposed area are used to calibrate the dose for the best contrast [1]. This method is used to replace the conventional methodology that uses an open loop, calibration method and is preferred as it decreases the number of exposures. The maximum likelihood estimation of inputs having a Laplacean distribution is given by the median input value [2]. Online calibration and maximum likelihood estimation are important kernel operations in several beamforming algorithms [3]. Reconstruction of a missing input value using a combination of the input value of k nearest sensors has been exploited in [4], which discusses a neural network based software sensor system for coagulation control.

Selection is also used in several network management protocols. Consider a scenario, where sensors detect an event, and the application requires only the i nearest sensors to monitor it at any time. The i nearest sensors can be found based on a measured data sample. If the event is a fire in the forest, the sensors nearer to the fire measure higher temperature. For target detection problems, when the acoustic or seismic source is in the near-field, the curvature of the received signal depends on its distance from the receiver [3]. Alternatively, dynamic resource allocation in the network involves identification of a subset of sensors that have remaining power above a threshold. Rank based ordering is also used in implementation of several scheduling, routing [5] and synchronization algorithms in the network.

In this paper, we present a time and energy optimal algorithm for the selection problem in a single hop network of n randomly distributed sensor sensors. The algorithm is also *energy-balanced*, which implies that the energy dissipation in all the sensors is uniform, and no sensor dies early due to

excessive use. This is desired to ensure that the functionality of the network is not affected due to early die out of some critical sensors (e.g., portion of the area left unmonitored).

We assume a single-hop, time-synchronized, wireless sensor network. This implies that all sensors can hear transmission from any sensor in the network. Only one sensor can transmit at any given time on a chosen communication channel to ensure collision free transmission. A *uniform cost* model (see Section II) has been utilized for our analysis. This model assumes that the transmission range of the sensor radio is fixed. Energy dissipation for local computation, transmission or reception of a unit of data is unity.

In wireless radio networks, sensors dissipate significant amount of energy in listening to transmissions not intended for them. Thus, in addition to reducing the amount of computation and communication in the network, the algorithm should also define a schedule for switching off sensors when not participating in any computation or communication. The wake up of sleeping (switched off) sensors can be self-initiated based on an internal timer or on detection of an event. Self-initiated wake ups can result in sensors missing events when switched off. Thus, several researchers [6] [7] have proposed use of an additional low power paging or signaling channel to wake up and synchronize transmission between sensors.

We demonstrate that given n randomly distributed sensors in a single-hop, single-channel network, selection can be performed in $O(n)$ time and energy with no sensor awake for more than $O(1)$ time steps. In a p -channel sensor network, where $p \leq n^{1-\epsilon}$ and $0 < \epsilon \leq 1$, selection can be performed in $O(n/p)$ time, with $O(n)$ energy, and no sensor awake for more than $O(1)$ time steps.

The rest of the paper is organized as follows. We discuss our network and energy model assumptions in Section II. Related work is compared in Section III. The property *energy-balanced* is defined in Section IV. Our algorithms for selection are described in Section V. Simulation results are presented in Section VI, and we conclude in Section VII.

II. NETWORK AND ENERGY MODEL

In this section we define the **Energy-aware, Single-hop, Uniform-cost (ESU)** model for wireless sensor networks. The assumptions for the model are discussed below.

- The ESU model is defined for a single-hop sensor network. Sensors communicate over one or more broadcast communication channels. At any given time, only one sensor can transmit over a single broadcast channel. Two or more sensors transmitting concurrently over a single channel result in a collision. In a unit time step, a sensor can tune to at most one communication channel.
- Along with a communication channel, sensors are equipped with a wake up mechanism. The wake up mechanism can be implemented using either a paging channel or internal hardware timers. The power dissipation for the wake up mechanism is considered to be negligible.
- The energy dissipation at a sensor is defined to be the sum of the transmission energy, reception energy, and

the computation energy. The sensors are assumed to have a fixed transmission range. Thus, the energy dissipation for transmission, reception, and local computation of one unit of data is assumed to be constant. The value of the constants varies depending on the radio electronics, the type of processor, and other hardware-dependent factors [8]. To keep our analysis independent of the implementation technology, we normalize the constants to unity. Thus, we define a *uniform cost* model for analyzing algorithms, where energy and time cost for transmission, reception, and computation of a unit of data is unity.

- A sensor has two power states active or switched off. A sensor can transmit, receive, or compute only in the active state. In the active state energy dissipation per unit time is unity. In the switched off state energy dissipation is zero.
- All sensors are homogeneous and globally time synchronized. Time synchronization has been investigated in [9]. The GPS system (if available) can also be used for time synchronization.
- The network has been initialized and each sensor has a unique id. Energy efficient initialization algorithms have been discussed in [10].

Wake Up Mechanism: Our model assumes existence of a wake up mechanism for the sensors. This can be implemented either using hardware timers or a paging channel. In this paper, we consider presence of a low power paging channel, but our algorithms can be easily modified to run with internal timers only. The power dissipation in the paging channel is negligible. It has very low bandwidth and is active all the time. At any time a sensor can send wake up signals to all other sensors in the network. We assume that only the following four types of wake up signals are transmitted over the channel. (a) Wake up a specific sensor, (b) Wake up all sensors, (c) Wake up all sensors with index smaller than the broadcasted index, and lastly, (d) Wake up all sensors with index larger than the transmitted index.

III. RELATED WORK

In this section we discuss other related research focused upon algorithm design for wireless sensor networks. Sensors in a wireless single hop sensor network use broadcasts for communication. The BCM model defined in [11] has been extensively used for algorithm design for sensor networks. The ESU model proposed in this paper has several similarities with the BCM model, owing to the broadcast nature of the transmissions. The BCM model has been primarily utilized for time analysis (minimizing broadcasts), whereas energy awareness is the emphasis of the ESU model. The wake up mechanism discussed in the ESU model aids in designing efficient power state schedules for the sensors. The wake up mechanism has not been defined in the BCM model.

Our analysis of the algorithm $INDEX_l(n)$ discussed in Section V-D is similar to the prefix summing algorithm discussed in [12]. Sorting and ranking in a broadcast communication

model has been investigated in [11] for reducing the total number of broadcasts in the system. The algorithms proposed are energy optimal if the receiving power of sensors is negligible and only broadcast energy is accounted for. The algorithm presented in this paper is energy optimal even if the receiving power of sensors is significant.

We assume that the network is time-synchronized and initialized. An energy efficient initialization algorithm has been discussed in [10]. Time synchronization in wireless networks has been investigated in [9].

IV. ENERGY BALANCED ALGORITHMS

A distributed algorithm is said to be *energy-balanced* if energy dissipation in all the sensors is uniform. Thus, if the overall energy dissipation in a sensor network of n sensors is given by $E(n)$, an energy-balanced algorithm has the property that no sensor in the network dissipates more than $O(E(n)/n)$ energy.

Consider the following scenario. A single-hop network of n sensors is deployed to monitor a field. At any time sensor i , where $i \leq n$, finds its remaining battery power to be below a threshold value. It wants to find the sensor with maximum remaining power that can take over its tasks. We assume $i = n$ and consider the following two algorithms for finding the id of the sensor with the maximum remaining power.

(a) Sensor n maintains variables m and x initialized to 0. At the end of the algorithm, variable m stores the maximum power, and x stores the id of the sensor with maximum power. All sensors transmit their power level to sensor n one by one. Each time sensor n receives a value, it compares the received value with m . If the received value is larger, it updates the variable x with the identity of the sender, and assigns m the new value. At the end of $n - 1$ transmissions sensor n stores the id of the sensor with the maximum remaining power in x .

(b) Each sensor maintains a variable x and m initialized to its own id and remaining power. At time t , where $0 \leq t \leq n - 2$, sensor $t + 1$ transmits the value of x and m to sensor $t + 2$. Sensor $t + 2$ compares the value of received m to its own. If received m is larger, it updates its variable m and x with the value received. At the end of $n - 1$ iterations sensor n receives the id of the sensor with maximum power.

Both the algorithms described above involve a sequence of $n - 1$ transmissions, receptions, and comparisons. They have time and energy complexity $O(n)$. However, algorithm (b) is energy-balanced, whereas algorithm (a) is not energy-balanced. In algorithm (b) all sensors dissipate energy for at most one transmission, reception and comparison, which is $O(1)$. In algorithm (a) all the sensors except sensor n dissipate $O(1)$ energy, whereas sensor n dissipates $O(n)$ energy. Note that this can be very harmful to this sensor as it was already low in power.

The property that an algorithm is *energy-balanced* ensures that all sensors deplete power in a uniform manner, and no sensor is overused. One of the goals in deployment of sensor networks is that the network remains functional for the maximum time. If energy dissipation is not uniform in the

network, a network will collapse even if the total amount of power in the sensors is large but some critical sensors have exhausted their battery.

V. SELECTION

We consider the selection problem in a single hop network of n randomly distributed sensors, where each sensor contains a single data element. Our goal is to find the i^{th} smallest data element among all the sensors.

We assume that all sensors have a unique id s and a unique index j , where $1 \leq s, j \leq n$. Initially at start of the algorithm we assign $j = s$. In the remaining part of the algorithm, s remains fixed, but the index j keeps changing as per the context. Assigning a new index to a sensor is termed as reindexing. For our initial analysis we assume a single-hop network with a single communication channel, and extend our results to a p -channel network in Section V-D.

A. Time Optimal Implementation

A time optimal algorithm for sorting has been discussed in [11]. The algorithm can be used to solve the selection problem and is described as follows. At time step t , where $0 \leq t \leq n - 1$, sensor $s = t + 1$ transmits and all the other sensors listen. Each sensor maintains a counter initialized to one. Each time a sensor hears an element smaller than its value, it increments the counter by one. After, n transmissions, the counter value at each sensor denotes its rank. At the end of n steps, the sensor with rank i transmits the value to all the other sensors.

Note that the above algorithm works on the assumption that all data elements are distinct. Consider the scenario where two sensors contain elements of the same rank. They will transmit at the same time resulting in a collision over the channel. However, a small modification to the above algorithm can ensure that all elements have a unique rank. Each time a sensor which has not transmitted its result as yet, hears a value equal to its own, it increments the rank counter. This ensures that the rank of all data elements are distinct.

Analysis of the Algorithm: The above runs in time $O(n)$. The algorithm is time optimal as each sensor must transmit data at least once. Each sensor transmits once and receives $O(n - 1)$ messages. The energy complexity of the algorithm is $O(n^2)$ and each sensor is awake for $O(n)$ time steps. The algorithm is energy balanced, but is not energy-optimal. As discussed in Section V-B, the algorithm can be implemented using energy $O(n)$.

B. Energy and Time Optimal Implementation

A worst case linear time selection algorithm for a uniprocessor system has been discussed in [13]. This algorithm can be adapted for a single-hop sensor network as illustrated in Figure 2.

We discuss the sub procedure *INDEX* (n), followed by the description and analysis of the algorithm *SELECT* (n, i). The goal of the algorithm is to find the data element of rank i , in a single-hop network of n sensors.

INDEX_l(n)

1. Sensor with index $j = 1$ initializes $b = l$, wakes up sensor with index $j = 2$, broadcasts b , and switches off.
 2. For ($t = 0$; $t < n - 2$; $t++$).
Sensor $j = t + 2$ stores the received value b_{recv} in b .
if($l > 0$) { $b++$, $l = b$.}
Sensor j wakes up sensor $j + 1$, broadcasts b , and switches off.
 3. End.
-

Fig. 1. Pseudocode for *INDEX_l(n)*

- 1) **INDEX_l(n)**: Consider n sensors in a single hop network. Each sensor maintains a copy of variable $0 \leq l \leq 1$. Let k represent the total number of sensors with $l > 0$. The goal of this algorithm is to determine k , and reindex the sensors with $l > 0$ uniquely from $1 \rightarrow k$. The pseudocode for the algorithm is discussed in Figure 1, and it is used to reindex sensors in Step 8 of the algorithm *SELECT(n, i)*. A token variable b is passed among the sensors. On receiving the b , each sensor does the following. It checks the stored variable l and increments b if $l > 0$. Also it assigns $l = b$. At the end of the algorithm the value of b at the last sensor denotes the number of sensors with $l > 0$ and this is broadcasted to all the sensors. Moreover, the value of l at each sensor, where $l > 0$ denotes its new index. These sensors can reindex themselves by assigning $j = l$. The algorithm requires variable b to be circulated among the sensors. Simple analysis shows that there are $O(n)$ transmissions, comparisons and receptions. The time and energy complexity of this algorithm is $O(n)$. Each sensor is awake for at most $O(1)$ time steps. Note that the sub procedure *INDEX_h(n)* is similar. It operates on variable h instead of l .

- 2) **SELECT(n, i)**: The algorithm proceeds as follows. The sensors divide themselves into groups of five and find the median of each group. The algorithm is then recursively called on the sensors with the median values to find the median-of-medians. This median of medians is stored in variable x , and is used to partition the sensors into two groups. One group, with data elements larger than x , and the other set contains the remaining data elements. The sensors with data elements $v \leq x$ assign $l = 1$ and the remaining assign $h = 1$. The sensors are reindexed using sub procedures *INDEX_l(n)* and *INDEX_h(n)*. Sensor n checks the value of variables l and h . If $l > 0$, it broadcasts l else $n - h$. This represents the number of data elements smaller than x . If $i \leq x$, the element of the i^{th} rank lies among sensors with $l > 0$, else it lies in sensors with $h > 0$. The appropriate set is chosen and the procedure is recursively called after

SELECT(n, i)

1. The n sensors divide into $\lceil n/5 \rceil$ groups. A sensor with index j belongs to group $(j - 1)/5 + 1$.
 2. The median of the data elements in each of the $\lceil n/5 \rceil$ groups is found by sorting the elements of each group (of which there are five at most) and taking its middle element. (If the group has an even number of elements take the larger of the two medians.)
 3. Each sensor maintains a copy of its index. Reindex the sensors containing the median elements to the index of their group. Remaining sensors switch off.
 4. Use *SELECT*($\lceil n/5 \rceil$, $\lceil n/10 \rceil$) recursively to find the median-of-medians x .
 5. Wake up all sensors, restore old index, and broadcast x .
 6. Each sensor compares its data element v with x .
 7. If $v \leq x$, it sets $l = 1$ else $h = 1$.
 8. Reindex sensors using *INDEX_l(n)* and *INDEX_h(n)*.
 9. Sensor n broadcasts l if $l > 0$, and $n - h$ if $h > 0$. This value is stored as k in the sensors and represents the number of elements smaller than or equal to x .
 10. Each sensor compares the value of i with k .
 11. if($i \leq k$) {
Sensors with bit $l = 0$ switch off.
Remaining sensors reindex to l .
Recursively call *SELECT(k, i)*.
} else {
Sensors with bit $h = 0$ switch off.
Remaining sensors reindex to h .
Recursively call *SELECT(n - k, i - k)*.
}
 12. End.
-

Fig. 2. Pseudocode for *SELECT(n, i)*

updating the parameters. The rejected sensors switch off.

Analysis of the Algorithm: We analyze the execution time and energy dissipation for the various steps of the algorithm. Step 1 is trivial. Each sensor with index j considers itself to be a member of the group $(j - 1)/5 + 1$. This requires a single time step with each sensor being awake for one time step. Similarly Steps 3, 5, 6, 7, 9, 10 require time $O(1)$ and energy $O(n)$, with each sensor being awake for $O(1)$ time steps. Step 8 invokes calls to sub procedures *INDEX_l(n)* and *INDEX_h(n)*, which require time and energy $O(n)$ (as discussed earlier). Step 2 involves sorting all the elements in their respective groups. This can be accomplished in time n with no sensor being awake for more than five time steps. At time t , where $0 \leq t \leq n - 1$ sensor with index $j = t + 1$ transmits and sensors with index j_1 such that $(j_1 - 1)/5 + 1 = (j - 1)/5 + 1$ (same group) listen. Each listening sensor compares the transmitted value to its own value to compute its rank in the group. Total time for this step is n and a sensor is awake for at most 5 time steps.

Step 4 is a recursive step. Let the time and energy complexity of this algorithm be $T(n)$ and $E(n)$ respectively. Step 4 can be executed in time $T(n/5)$ using energy $E(n/5)$. Similarly, Step 11 has time and energy complexity $T(\max(k, n - k))$ and $E(\max(k, n - k))$ respectively.

Note that in Step 11 *SELECT* is recursively called on at most $7n/10 + 6$ elements. This is because at least half the medians found in Step 2 are greater than or equal to x . Thus at least half of the $\lceil n/5 \rceil$ groups contribute three elements greater than x except the last group which is of smaller size and the group containing x . Thus no of elements greater than x is at least $3n/10 - 6$. Similarly number of elements smaller than x is at least $3n/10 - 6$. The time and energy complexity is given by (refer [13] for details).

$$\begin{aligned} T(n) &\leq T(n/5) + T(7n/10 + 6) + O(n) && \text{if } n > 80 \\ &\leq \Theta(1) && \text{if } n \leq 80 \\ E(n) &\leq E(n/5) + E(7n/10 + 6) + O(n) && \text{if } n > 80 \\ &\leq \Theta(1) && \text{if } n \leq 80 \end{aligned}$$

By substitution it can be shown that both time and energy complexity for this algorithm is $O(n)$. Rank estimation requires at least $n - 1$ comparisons, which implies that the lower bound on energy dissipation is $O(n)$. Thus, the algorithm *SELECT* (n, i) is energy optimal.

The above algorithm is time and energy optimal, but is not energy balanced. Consider a scenario where the data elements are sorted. This implies that sensor with index j stores element with rank j as illustrated in Figure 3. We want to find the element with rank n . Clearly, this algorithm will require sensor n to be awake for the entire time. Sensor n is awake for $O(n)$ time steps, whereas sensor 1 is awake for fewer time steps. Thus, the number of time steps a sensor must stay awake depends on the data element stored. In the following subsection, we discuss an energy-balanced implementation of the above algorithm.

C. Optimal Energy-balanced Implementation

The main observation made in the previous algorithm was that the number of time steps that a sensor was awake depended on the rank of its data element. However, to ensure energy balancing the algorithm should be oblivious to the input. This can be achieved by shuffling data between various recursive calls of the algorithm as described in algorithm *BSELECT* (n, i, d) (see Figure 5). The goal of the algorithm is to find the data element of rank i among n sensors in an energy-optimal, energy-balanced manner. The algorithm has an additional parameter d , which denotes the depth of recursion for any instance of the algorithm. Initially $d = 0$.

We first discuss the sub procedure *SHIFT_l* (n, k), followed by description and analysis of algorithm *BSELECT* (n, i, d).

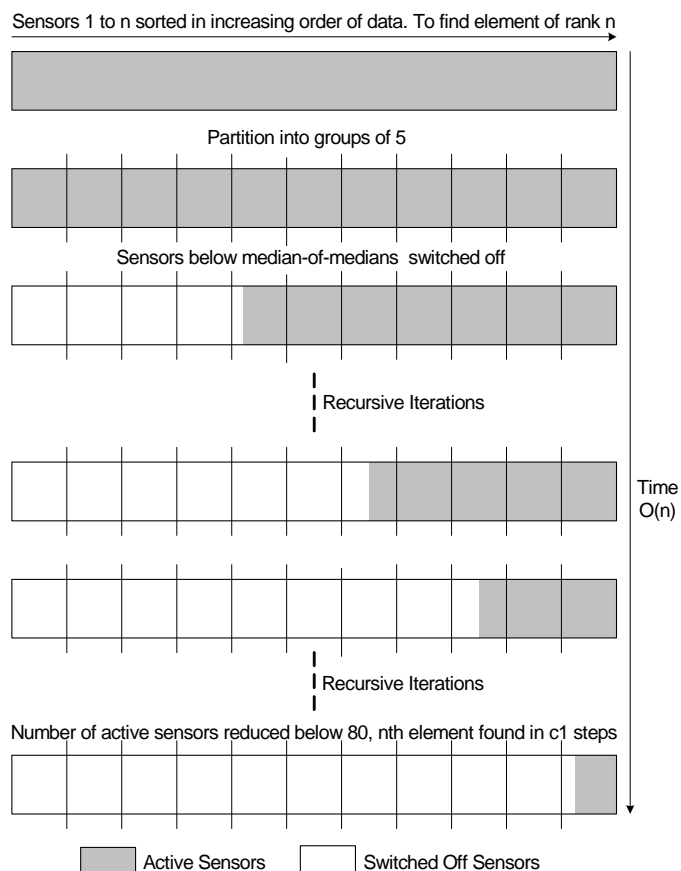


Fig. 3. Power states of sensors in recursive calls to *SELECT*

- 1) ***SHIFT_l*** (n, k): The procedure *SHIFT_l* (n, k) moves data from k sensors with variable $l > 0$ to the sensors having index $n - k + 1 \leq j \leq n$. The pseudocode for the algorithm is illustrated in Figure 4. Consider the sensors to be arranged in increasing order of index from left to right, the k data elements of sensors with $l > 0$ are shifted into a contiguous block of size k at the right most position (see Figure 6).

The algorithm is as follows. All sensors store variable $l \geq 0$. The variables l_n and h_n are passed among all the sensors in the order of their index. Each sensor checks if it has data to be shuffled. If the sensor has $l > 0$ and index $j < (n - k)$ it increments variable l_n by one. Similarly, if $l = 0$ and $j \geq (n - k)$, it increments h_n by one. At the end of this step, at time t each sensor with $l > 0$ and $l_n = t$, transmits to sensor with $l = 0$ and $h_n = t$ and then the reverse. Thus, the data is shuffled in a collision free manner.

Simple analysis shows that the time and energy complexity of this algorithm is $O(n)$. Each sensor is awake for $O(1)$ time steps. *SHIFT_h* (n, k) works similarly but operates on variable h instead of variable l .

- 2) ***BSELECT*** (n, i, d): The algorithm *BSELECT* (n, i, d) is an energy-balanced adaptation of the *SELECT* (n, i)

SHIFT_l(n, k)

1. Each sensor initializes $l_n = l$ and $h_n = h$.
 2. For ($t = 0$; $t < n - 1$; $t++$)
Sensor $t + 1$ wakes up sensor $t + 2$, broadcasts l_n, h_n .
Sensor $t + 1$ switches off.
Sensor $t + 2$ checks value of l and index j .
If ($l > 0$) and ($j < (n - k)$) l_n++ .
If ($l == 0$) and ($j \geq (n - k)$) h_n++ .
 3. Sensor n broadcasts l_n , the number of sensors with $l > 0$, which is stored in all sensors as m .
 4. At $0 \leq t \leq m - 1$, sensor with ($l_n = t + 1$) and ($l > 0$) exchanges data with sensor containing ($h_n = t + 1$) and ($l = 0$).
 5. All sensors switch off.
 6. End.
-

Fig. 4. Pseudocode for *SHIFT_l(n, k)*

algorithm discussed earlier. Analysis of the *SELECT*(n, i) algorithm shows that for $n \leq 80$, the total time to run is $\Theta(1)$. Any sensor is awake for at most $\Theta(1) \leq c_1 = O(1)$ time steps. Thus, we focus upon the scenario when $n > 80$. As discussed in Section V-B, each recursive call to procedure *SELECT*, reduces the data size from n to $7n/10 + 6$ or less. For $n > 80$, $(7n/10 + 6) < 8n/10$. Consequently, after $k = 2/\log(10/8) \leq 7$ recursive calls the data size is reduced from $n \rightarrow n/4$ or less. Thus, only $n/4, n/16, n/32 \dots$ sensors stay awake after every seven recursive calls. This property is exploited to redistribute data in the sensors to ensure that the algorithm is energy balanced.

The algorithm *BSELECT*(n, i, d) is identical to algorithm *SELECT*(n, i) except for Steps 1 and 12 which perform the data shuffling. Step 12 ensures that at the end of a single recursive call, the reduced data is on the right side of the block. This is performed by invoking sub procedure *SHIFT_l(n, k)* (or *SHIFT_h(n, k)*). Consider the scenario when a recursive call on procedure reduces the the data set to m . These m sensors are distinguished using variable l (or h), which is set to one. The procedure shifts data from sensors with index $j \leq m$ to sensors with index $j \geq n - m$ as illustrated in Figure 6.

Step 1 performs the data shuffling after every 7 recursive calls to the procedure. After every 7 recursive calls the data set is reduced from n to $n/4$ elements which is stored in sensors with id $3n/4 \leq j \leq n$. The data is shifted to sensors with id $n/4 \leq j \leq n/2$. This ensures that each time sensors with smaller id are becoming active and with larger id are switched off (see Figure 7). Thus, no sensor is awake in more than $O(7) + O(c_1)$ time steps which is $O(1)$.

Analysis of the Algorithm BSELECT(n, i, d): The

BSELECT(n, i, d)

1. if ($d == 7$) and ($n > 1$) {
Assign $d = 0$.
Let z be the id of sensor with index n .
Let s_{t+1} be id of sensor with index $t + 1$.
At time $0 \leq t \leq n - 1$, sensor with id s_{t+1} wakes up. It wakes up sensor with id $s_{t+1} - z/2$ and exchanges data.
Sensor with id s_{t+1} switches off and sensor with id $s_{t+1} - z/2$ reindexes to $t + 1$.}
 2. The n sensors divide into $\lceil n/5 \rceil$ groups. A sensor with index j belongs to group $(j - 1)/5 + 1$.
 3. The median of the data elements in each of the $\lceil n/5 \rceil$ groups is found by sorting the elements of each group (of which there are five at most) and taking its middle element. (If the group has an even number of elements take the larger of the two medians.)
 4. Each sensor maintains a copy of its index. Reindex the sensors containing the median elements to the index of their group. Remaining sensors switch off.
 5. Use *BSELECT*($\lceil n/5 \rceil, \lceil n/10 \rceil, 0$) recursively to find the median-of-medians x .
 6. Wake up all sensors, restore old index, and broadcast x .
 7. Each sensor compares its data element v with x .
 8. If $v \leq x$, it sets $l = 1$ else $h = 1$.
 9. Reindex sensors using *INDEX_l(n)* and *INDEX_h(n)*.
 10. Sensor n broadcasts l if $l > 0$, and $n - h$ if $h > 0$. This value is stored as k in the sensors and represents the number of elements smaller than or equal to x .
 11. Each sensor compares the value of i with k .
 12. if($i \leq k$) {
SHIFT_l(n, k)
Sensors with index $1 \leq j \leq n - k$ switch off.
Remaining sensors reindex from j to $j - (n - k)$.
Recursively call *BSELECT*($k, i, d + 1$).}
else {
SHIFT_h(n, n - k)
Sensors with index $1 \leq j \leq k$ switch off.
Remaining sensors reindex from j to $j - k$.
Recursively call *BSELECT*($n - k, i - k, d + 1$).}
 13. End.
-

Fig. 5. Pseudocode for *BSELECT(n,i,d)*

analysis of Steps 2 to 11 is the same as that of Steps 1 to 10 in algorithm *SELECT*(n, i). Step 1 requires n data transmissions and requires $O(n)$ time to run using $O(n)$ energy. Step 12 is similar to Step 11 of algorithm *SELECT*(n, i) except for an extra call to procedure *SHIFT_l(n, k)* (or *SHIFT_h(n, k)*). This procedure can be executed in time $O(n)$, with energy $O(n)$ with no sensor awake for more than $O(1)$ time steps.

Note that the overall energy dissipation of the algorithm is increased due to the remapping required in the algorithm. However, the number of elements shifted

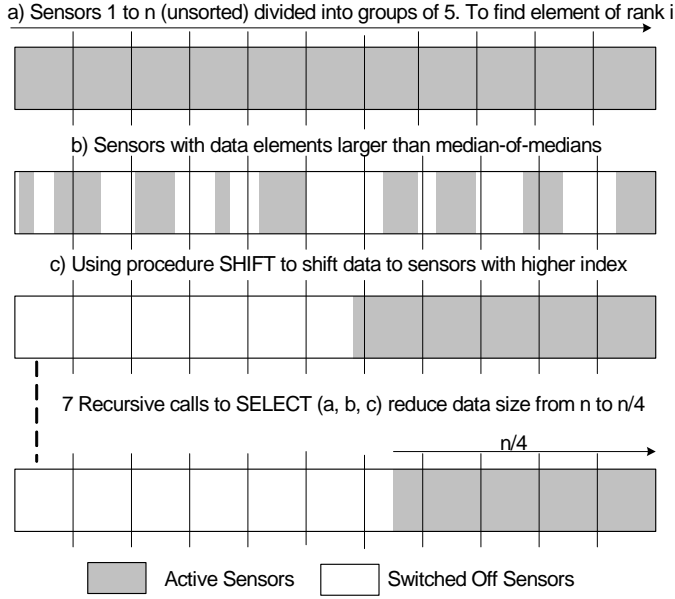


Fig. 6. Power states of sensors in a single call to *BSELECT*

in successive calls are $n/4 + n/16 \dots n/2^k$, which is smaller than $n/2$. Thus, the overhead is not large as observed in simulations (see Section VI).

Remark 5.1: The proposed algorithm for selection is not an in place algorithm. Thus, if the data order in sensors is important a copy must be maintained at each sensor.

Theorem 5.1: In a single channel, single hop sensor network of n sensors, selection can be performed in time $O(n)$, using energy $O(n)$ and no sensor awake for more than $O(1)$ time steps.

D. Selection in a p -channel sensor network

In this section, we extend our analysis for the selection algorithm in a single-hop network with a single channel to a network with p communication channels. Having more than one communication channel permits a larger number of sensors to communicate concurrently, and thereby increasing the amount of parallelism in the execution of the algorithm.

In order to communicate over a chosen channel, both the transmitting and the receiving sensor must agree on a common communication channel. One strategy is that an extra signaling channel is used to tell sensors, the channel they must communicate upon. However, this results in additional energy dissipation. On the contrary, our channel assignment is static and is computed by a sensor using only its index value and no additional communication.

We first discuss the implementation of the function $INDEX_l(n)$ on a p -channel sensor network. Note the analysis of procedure $SHIFT_l(n, k)$ is similar. An algorithm for a p -channel implementation of prefix summing discussed in [12] can be adapted for this algorithm and is discussed below. In order to keep the analysis simple, we assume n is

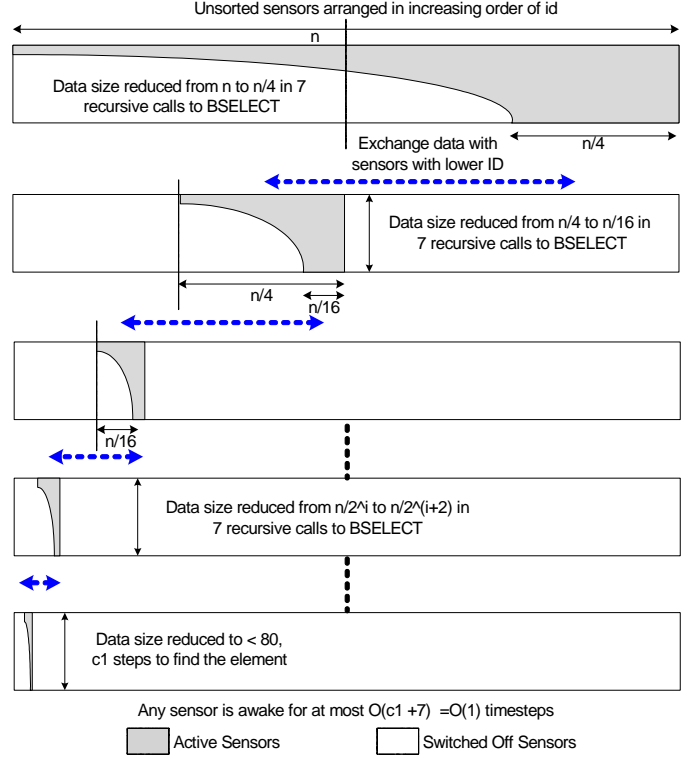


Fig. 7. Power states of sensors in recursive calls to *BSELECT*

divisible by p .

$INDEX_l(n)$

1. Divide sensors into p groups of size n/p each.
 2. Each group is assigned a distinct index g , where $1 \leq g \leq p$.
 3. All sensor reindex from j to $(j - 1) \bmod n/p + 1$.
 4. Assign a distinct channel to each group.
 5. Compute in parallel $INDEX_l(n/p)$ for each group.
 6. For $(t = 0, t < p + 1, t++)$
 - Sensor with index n/p in group $g = t + 1$ wakes up all sensors in group $g + 1$ and broadcasts l .
 - Each sensor in group $g + 1$ with $l > 0$ increments l with received value and switches off.
 7. End.
-

Fig. 8. Pseudocode for $INDEX_l(n)$

Lemma 5.1: The procedure $INDEX_l(n)$ can be implemented on a p -channel, single-hop sensor network to run in time $O(n/p)$, such that the energy dissipation is $O(n)$ and each sensor is awake for at most $O(1)$ time steps, where $p \leq n^{1-\epsilon}$ and $0 < \epsilon \leq 1$.

Proof: For simplicity, we first consider the case, where $p \leq \sqrt{n}$. The sensors are divided into p groups and a separate channel is assigned to each group. $INDEX_l(n)$ is called in parallel, on all groups of size n/p each. The time to execute

$INDEXG_l(n, p, c)$.

1. Repeat Steps 1 to 5 of $INDEXP_l(n)$.
 2. Each sensor with index $j = n/p$ assigns $l_o = l$, $j_o = j$ and all other sensors switch off.
 3. Awake sensor are assigned index of their group g .
 4. If ($c > 0$) recursively call $INDEXG_l(p, \sqrt{p}, (c-1))$.
 5. Each group with index g , is assigned channel g .
 6. Each awake sensor wakes up all sensors in its group.
 7. It broadcasts $l - l_o$ and reindexes to j_o .
 8. Each listening sensor increments l by the received value.
 9. End.
-

Fig. 9. Pseudocode for $INDEXG_l(n, p, c)$

this step is $O(n/p)$. Energy dissipated is $O(n/p)$ for each group with no sensor awake for more than $O(1)$ time steps. Thus, total energy is $O(n)$. In the next step, the value of l is transmitted sequentially (in order of group index) from sensor in a group to all sensors in the successive group. The sensors in the receiving group increment l by the received value. Time taken for this step is $O(p)$ since there are p groups and energy is $O(n)$. Thus total execution time is $O(n/p + p) = O(n/p)$ for $p \leq \sqrt{n}$ using $O(n)$ energy, with each sensor awake for $O(1)$ time steps.

Next we consider the scenario, where $p = n^{1-\epsilon}$ and $0 < \epsilon \leq 1$. Let $2^c = (1 - \epsilon)/(1 - 2\epsilon)$. The algorithm is described in Figure 9. Consider Step 4 of the algorithm. Each recursive call take time $p^{1/2^b}$, where $1 \leq b \leq c$. The remaining steps can be done in time $O(n/p)$. Thus, total running time for the algorithm is $O(n/p + p^{1/2} + p^{1/4} + \dots + p^{1/2^c}) = O(n/p + p^{1-1/2^c}) = O(n/p + p^{\epsilon/(1-\epsilon)}) = O(n/p + p^{1/(1-\epsilon)}/p) = O(n/p)$. \square

Theorem 5.2: Given a single-hop sensor network with p channels and n randomly distributed sensors, selection can be performed in time $O(n/p)$ with energy $O(n)$, and no sensor awake for more than $O(1)$ time steps. $p \leq n^{1-\epsilon}$, where $0 < \epsilon \leq 1$.

Proof: The proof is by induction on the problem size n . Let us consider the base scenario where $n = 2$ and $p = 1$, the rank of both elements can be found using two broadcasts. Let us assume that Theorem 5.2 holds for all problem sizes smaller than n . We analyze the algorithm for problem size n .

Step 1 involves $O(n)$ transmissions, which can be performed in time $O(n/p)$, using energy $O(n)$ in a p -channel network.

Steps 2, 4, 6, 7, 8, 10 and 11 are computed in time $O(1)$. Step 2 involves sorting the elements of each group and takes time $O(n)$. It can be simulated in time $O(n/p)$ in a p -channel network as follows. Let us consider that a sensor with index j belongs to group g , where $g = (j-1)/5 + 1$ and the channels are labeled from $1 \rightarrow p$. Assign all sensors in group g , channel $(g-1) \bmod p + 1$. We have divided the sensors into $\lceil n/5p \rceil$

blocks. Each block has p groups that have distinct channel assignment. Sensors in a block belonging to different groups can transmit concurrently. Thus, p sensors can transmit at the same time on distinct channels, and sorting can be performed in time $O(n/p)$ with energy $O(n)$ and no sensor being awake for more than $O(1)$ time steps.

Step 9 invokes calls to $INDEX_l(n)$ and $INDEX_h(n)$. Using Lemma 5.1, we conclude that time taken for these steps is $O(n/p)$ and energy dissipation is $O(n)$, with each sensor awake for $O(1)$ time steps.

Step 12 calls procedures $SHIFT_l(n, k)$ and $SHIFT_l(n, k)$. Steps 1 to 3 of procedure $SHIFT_l(n, k)$ are similar to $INDEX_l(n)$. Using Lemma 5.1, we conclude that running time is $O(n/p)$ using energy $O(n)$, with each sensor awake for $O(1)$ time steps. Step 2 of procedure $SHIFT_l(n, k)$ requires m transmissions. A sensor with index t is assigned channel $(t-1) \bmod p + 1$. The transmission can be completed in time $m/p = O(n/p)$ using $O(n)$ energy.

Steps 5 and 12 make recursive calls to the procedure $BSELECT$ on problems of size $n/5$ and kn , where $k \leq 7n/10$. Let us consider the recursive call on problem of size $n/5$. If $p < n/5$, we know $T(n/5) = O(n/5p)$ by our induction hypothesis. If $p \geq n/5$, we can use $p_1 < n/5$ channels such that $T(n/5) = O(n/5p_1) = O(n^\epsilon) = O(n/5p)$ for some small positive value of ϵ . Similar analysis holds for recursive call on problem of size kn . This implies $T(n/p) = O(n/p)$.

From the above analysis, we conclude that the algorithm $BSELECT$ can be implemented on a p -channel sensor network in time $O(n/p)$ with energy $O(n)$, and each sensor awake for $O(1)$ time steps. \square

VI. SIMULATION RESULTS

In this section we present our simulation results for algorithms $SELECT(n, i)$ and $BSELECT(n, i, d)$ discussed in Section IV. Our main objective is to examine the energy balancedness of the two algorithms, and also to analyze the energy and time overheads for data shuffling in algorithm $BSELECT(n, i, d)$. In algorithm $BSELECT$, we performed energy balancing by data shuffling after a every $d = 7$ recursive calls. This is the maximum number of recursive steps required to reduce the data set from n to $n/4$ as discussed earlier. However, in simulations we observed that the data set was reduced to $n/4$ in fewer recursive calls, and remapping after every $d = 3$ recursive calls gave slightly improved results. $BSELECT-2$ in the figures below represents this scenario.

The analysis has been performed for problem sizes ranging from 10 to 10000. The data set was generated randomly, and the algorithms were invoked to find the element with rank 2 (w.l.o.g.). The simulations were performed for 500 iterations for each problem size. The results presented in this paper represent the averaged values with 95% confidence interval and 0.01% (or better) precision.

The simulator was designed based on the assumptions discussed in Section II. A counter was maintained at each sensor to count the energy units dissipated and a global counter was used to compute the total number of time steps taken by

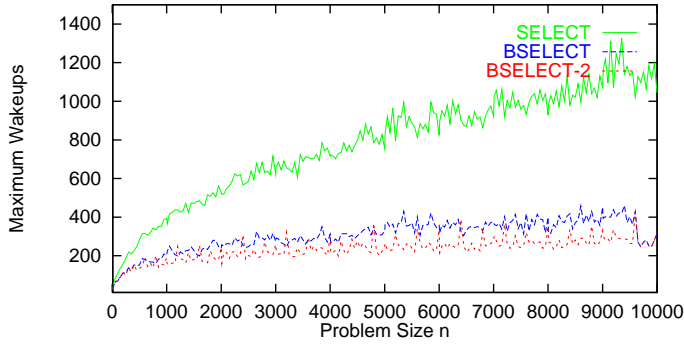


Fig. 10. Maximum number of time units a sensor is awake

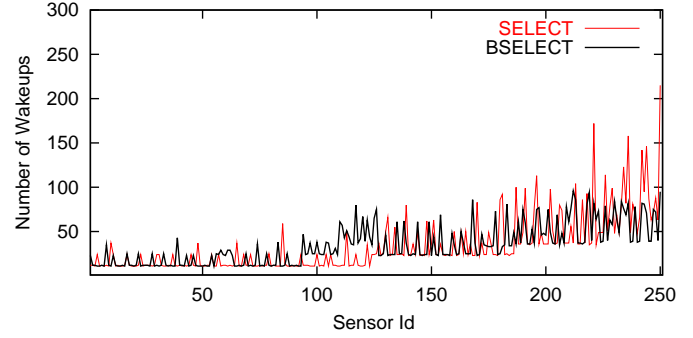


Fig. 11. Wake up distribution among sensors

the algorithm to run. Energy balancedness was compared by measuring the maximum number of wake ups for any sensor in the network.

Figure 10 illustrates the maximum number of timeslots a sensor is awake in the three algorithms. We observe that for algorithm *SELECT*, the maximum number of wake ups increases linearly with problem size and is $O(n)$. The maximum number of wake ups using algorithms *BSELECT* and *BSELECT-2* are linear for $n < 150$ but approach a constant value for larger values of n . The algorithm *BSELECT-2* reduces the number of wake ups marginally over algorithm *BSELECT*. Figure 11 shows the number of wake ups for all sensors in a network of size 250. The distribution is uniform for algorithm *BSELECT*, but shows large variations for algorithm *SELECT*.

Figure 12 and Figure 13 depict the time and energy dissipation of the algorithms assuming the uniform cost model discussed in Section II. We observe that the overall energy dissipation and time taken is least for algorithm *SELECT*, and highest for algorithm *BSELECT-2*. The increase in overall energy and time is a result of the data remapping overheads in algorithms *BSELECT* and *BSELECT-2*. The latter performs more remappings and thus has higher overheads. However, we observe that the overheads are very small as compared to the overall time and energy of the algorithm.

In our analysis, we have assumed a uniform cost model. In state-of-the-art sensors, the communication energy cost per byte is observed to be large than the computation cost per instruction by a factor of 1000 [8]. Energy balancing results in communication energy overheads. Figure 14 shows the energy costs for the three algorithms for a scenario, where a communication operation costs 1000 energy units as compared to a computation operation which consumes a single unit of energy. The overall energy of the system is increased for all three algorithms. However, again we observe that energy dissipation of algorithms *BSELECT* and *BSELECT-2* is larger than the energy dissipation of the algorithm *SELECT* by a very small fraction.

From the above simulations, we conclude that the algorithm *BSELECT* can be used to obtain an energy and time optimal, and energy balanced implementation of the selection problem.

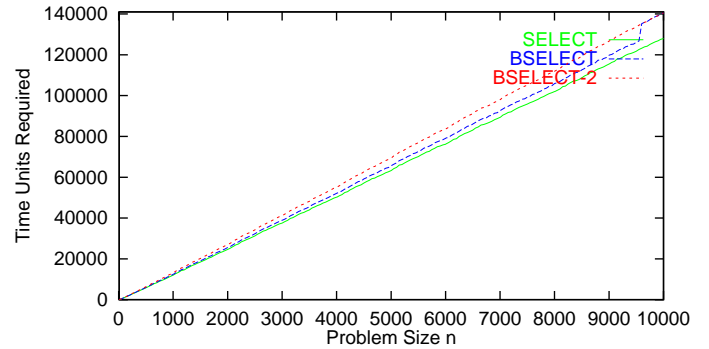


Fig. 12. Total number of time units in uniform cost model

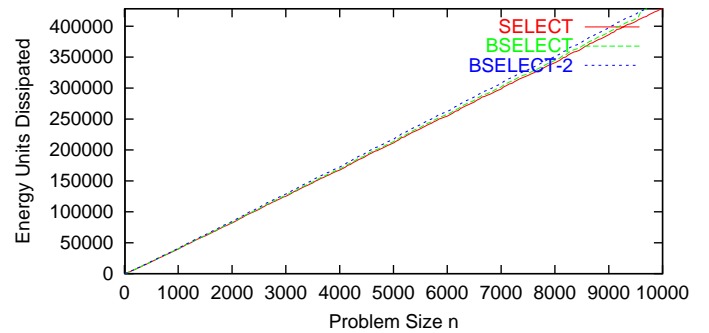


Fig. 13. Energy units dissipated in uniform cost model

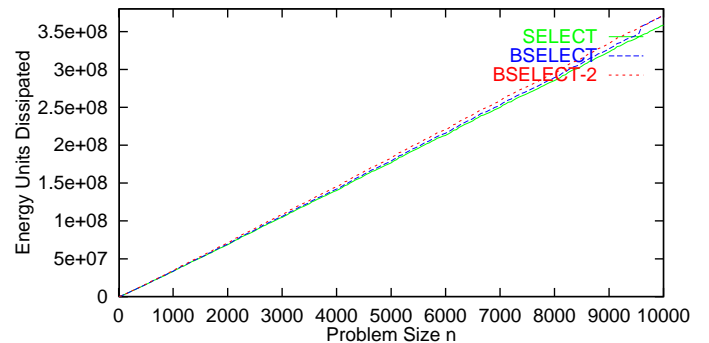


Fig. 14. Energy dissipated with communication cost 1000 units per operation

VII. CONCLUSION

In this paper, we have presented a time and energy optimal algorithm for the selection problem in a single-hop wireless sensor network. The results presented in this paper show larger improvements for networks with more than 80 sensors. Thus, they are applicable to either large sized single-hop networks or densely populate single-hop clusters in a larger multi-hop network. In [14], we discuss use a clustering approach for design of algorithms for multi-hop networks. The size of a single-hop network is limited by the transmission range of radios. Short transmission radios dissipate less power for a single transmission. However, in [15] it has been argued that increasing the transmission range of the sensor can improve robustness of the system.

Along with overall reduction in energy dissipation in the system, we have emphasized that the algorithms for sensor networks should also be energy-balanced to ensure uniform energy dissipation in the network. For a uniform cost model, energy balancing is similar to load-balancing as the aim is to insure that all sensors do asymptotically equivalent number of operations. However, a more rigorous analysis is required when the various operations have different energy costs. For example the transmission cost is higher than the reception cost.

We assumed a zero cost wake up paging channel for implementation and simulations of our algorithm. The paging channel can have significant overheads. The algorithm executes for $O(n)$ time steps and has n sensors. Thus energy dissipation can be as large as $O(n^2)$. Our algorithms can be easily adapted to function without an external paging channel, using internal timers only. However, such an approach is less robust as it requires more stringent time synchronization between the sensors. Moreover it introduces delay. For example, consider Step 6 in Figure 5. All sensors must time themselves to wake up at this step. The time taken for computation of Step 5 is data dependent and is $O(n/5)$. This implies that the time is less than $cn/5$ for some constant c , but the exact time depends on the data. If only timers are used to wake up sensors, the wake up must be scheduled assuming the maximum time which is $cn/5$. On the contrary using the paging channel the wake up can be scheduled immediately after completion of Step 5.

ACKNOWLEDGMENT

This work is supported by the DARPA Power Aware Computing and Communication Program under contract no. F33615-02-2-4005.

REFERENCES

- [1] I. E. Opris, "Analog rank extractors and sorting networks," Ph.D. dissertation, Dept. of Electrical Engineering, Stanford University, California, February 1996.
- [2] J. P. Norton, *An Introduction to Identification*. London Academic Press, 1998.
- [3] J. C. Chen, K. Yao, and R. E. Hudson, "Source localization and beamforming," *IEEE Signal Processing Magazine*, pp. 30–39, March 2002.
- [4] N. Valentin and T. Denoeux, "A neural network-based sensor for coagulation control in water treatment plant," *Intelligent Data Analysis*, vol. 5, pp. 23–39, 2001.
- [5] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *International Conference on Mobile Computing and Networking (MOBICOM)*, July 2001.
- [6] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2001.
- [7] C. S. Raghavendra and S. Singh, "Pamas-power aware multi-access protocol with signaling for ad hoc networks," *Computer Communications Review*, July 1998.
- [8] M. Singh and V. K. Prasanna, "System level energy trade-offs for collaborative computation in wireless networks," in *International Conference on Communications (ICC), Workshop on Integrated Management of Power Aware Communications, Computing and Networking*, May 2002.
- [9] J. Elson and D. Estrin, "Time synchronization in wireless sensor networks," in *International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless and Mobile Computing*, April 2001.
- [10] K. Nakano and S. Olariu, "Energy-efficient initialization protocols for radio networks with no collision detection," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, pp. 851–863, 2000.
- [11] K. Nakano, S. Olariu, and J. L. Schwing, "Broadcast-efficient protocols for mobile radio networks with few channels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, pp. 1276–1289, 1999.
- [12] K. Nakano, "Time and energy optimal list ranking algorithms on the k-channel broadcast communication model," in *International Computing and Combinatorics Conference (COCOON)*, August 2002.
- [13] T. H. Cormen, C. H. Leiserson, and R. L. Rivest, *Introduction To Algorithms*. The MIT Press, 1990.
- [14] M. Singh and V. K. Prasanna, "A hierarchical model for distributed, collaborative computation in wireless sensor networks," in *International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Advances in Parallel and Distributed Computational Models*, April 2003.
- [15] B. Krishnamachari, Y. Mourtada, and S. Wicker, "The energy-robustness tradeoff for routing in wireless sensor networks," Dept. of Electrical Engineering-Systems, University of Southern California, Tech. Rep. TR02-01, September 2002.