

Energy-Optimal and Energy-Balanced Sorting in a Single-Hop Wireless Sensor Network *

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

Abstract

A large number of sensors networked together form self-organizing pervasive systems that provide the basis for implementation of several applications involving distributed, collaborative computations. Energy dissipation is a critical issue for these networks, as their life-time is limited by the battery power of the sensors. In this paper, we focus on design of an energy-balanced, energy-optimal algorithm for sorting in a single-hop sensor network. Energy optimality implies that the overall energy dissipation in the system is minimized. Energy-balancedness ensures that all the sensors spend asymptotically equivalent amount of energy in the system. Uniform energy dissipation is desirable as it enables the network to remain fully functional for the maximum time. We demonstrate that given a single-hop, single-channel network of n randomly distributed sensors, sorting can be performed in $O(n \log n)$ time and energy, with no sensor being awake for more than $O(\log n)$ time steps. In a p -channel network, where $p \leq n^{1-\epsilon}$ for $0 < \epsilon \leq 1$, sorting can be performed in $O(n/p \log n)$ time and $O(n \log n)$ energy with no node being awake for more than $O(\log n)$ time steps.

1. Introduction

Sensor networks are large-scale, self-organizing, pervasive computing systems, where autonomous nodes (sensors) collaborate among themselves to achieve a larger objective. Such networks are being used in a large variety of scenarios ranging from military applications to ecological studies [5]. Sensor network applications often require deployment of sensors in remote areas where replacement of sensors is infeasible. Once a network is set up, the goal is to

keep it functional for the longest possible duration. Limited battery power makes energy a critical constraint for these networks.

Comparison problems (sorting, merging, searching) form an interesting group for algorithmic analysis, owing to their theoretical significance, and the large set of applications that require comparison kernels. Sensors collect a large amount of data samples from the environment and several samples may be erroneous. Input data samples collected at sensors are often sorted to filter out the extreme values. Pixel sorting is required in many image compression and coding algorithms [10]. Several applications require rank based ordering of sensors based on some metric. For example, sensors may be sorted in order of maximum remaining power [14] or proximity from a monitored target. A sensor network can also be considered as a distributed data management system [1]. Sorting is an important kernel in data mining and data management applications.

In this paper, we present an energy-optimal algorithm for the sorting problem in a single-hop network of n randomly distributed sensor nodes. The algorithm is also *energy-balanced*, which implies that the energy dissipation in all the sensors is uniform, and no node dies early due to excessive usage. 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. 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 2) has been utilized for our analysis, which assumes energy dissipation and time taken for local computation, transmission or reception of a unit of data to be unity.

In wireless radio networks, sensors dissipate significant amount of energy in listening to transmissions not intended

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

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 up can result in sensors missing events when switched off. Thus, several researchers [5] [11] have proposed use of an additional low power paging or signaling channel to wake up and synchronize transmissions between sensors.

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

Simulation results demonstrate that algorithm *BSORT* (n) discussed in Section 5.3 has the best energy performance. It is energy-balanced and energy-optimal.

The rest of the paper is organized as follows. We discuss our network and energy model assumptions in Section 2. Related work is presented in Section 3. The property *energy-balancedness* is defined in Section 4. Our algorithms for sorting are described in Section 5. We present our simulation results in Section 6, and conclude in Section 7.

2. 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. Sensors can detect collisions in the network.
- Along with a communication channel, sensors are equipped with wake up mechanism. The wake up mechanism can be implemented using either a paging channel or internal hardware timers.
- The energy dissipation at a sensor node 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 [12]. 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 inactive state energy dissipation is zero.
- All sensors are homogeneous and globally time synchronized. Time synchronization has been investigated in [4]. 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 [8].

Wake Up Mechanism: In our model, we assumed 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) a specific sensor is woken up, (b) all sensors are woken up, (c) all sensors with index smaller than a chosen value are paged, and lastly, (d) all sensors with index larger than the transmitted value are woken up.

3. Related Work

Several research groups have focused upon algorithm design for collaborative computation in sensors networks. The ESU model assumes a broadcast communication channel, and has several assumptions similar to the BCM [9] model. However, the BCM model has been primarily utilized for time analysis, whereas energy awareness is the emphasis of the ESU model. The paging channel defined in the ESU model aids in designing efficient power state schedules for the sensor nodes.

Our analysis of the algorithm *INDEX_l* (n) in Section 5.4 is similar to the prefix summing algorithm discussed in [7].

Significant amount of energy is dissipated in broadcasts over the communication channel. A balanced algorithm for sorting has been discussed in [2] and [9], which minimizes the total number of broadcasts (and time) in the system. The algorithm is energy-optimal if the receiving power of the nodes is negligible.

In this paper we assume a network model, where transmission power is equal to the reception power and present an energy-balanced, energy-optimal algorithm for the problem of sorting. Communication versus computation energy tradeoffs for sensor networks have been investigated in [12].

In our model, we assume that the network is time-synchronized and initialized. An energy efficient initialization algorithm has been discussed in [8]. Time synchronization in wireless networks is investigated in [4].

The selection problem has been investigated for a single-hop wireless sensor network in [13]. The energy balanced selection algorithm is an important kernel of the sorting algorithm discussed in this paper.

4. 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.

5. Sorting

The problem of sorting n numbers is one of the most widely analyzed problem owing to its theoretical importance and use in a wide range of applications. The sorting problem is defined as the following. We are given n sensors with one data element each. Each sensor has a unique id s , where $1 \leq s \leq n$. We want to redistribute the data such that at the end of the algorithm, the data element with rank i is stored in the sensor with id $s = i$. Here $1 \leq i \leq n$.

In addition to a unique id, each sensor is also assigned an index j , where $1 \leq j \leq n$. 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. Initially, we assume a single-hop network with a single communication channel, and extend our analysis to a p -channel network in Section 5.4.

5.1. Time-Optimal Implementation

A time-optimal algorithm for sorting has also been discussed in [9] and is described as follows. At a given time step t , where $0 \leq t \leq n - 1$, the sensor with id $s = t + 1$ 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. The data elements can now be routed to the appropriate destinations. At time step t , where $0 \leq t \leq n - 1$, sensor $t + 1$ listens and the sensor containing the element of rank $t + 1$ transmits.

MERGE (n_1, n_2)

1. All sensors with index j and $set = 1$ assign variable $p = j$.
 2. All sensors with $set = 0$ assign $p = 0$. Initially $i = 1$.
 3. Let s_i denote sensor with $set = 0$ and index i .
 4. If ($i > 2$) s_i receives p from s_{i-1} .
 5. All sensors except s_i switch themselves off.
 6. s_i wakes all sensors with index $j \geq p$ and $set = 1$.
 7. s_i broadcasts its data value v .
 8. The sensor with index n_2 and $set = 1$ compares received v to its data v_{n_2} .
 9. If ($v_{n_2} \leq v$) it transmits $p_{n_2} + 1$, where p_{n_2} denotes the value of p stored at this sensor.
 10. All awake sensors compare the received data v from s_i with their data.
 11. All sensors with smaller data switch themselves off.
 12. The awake sensors with $set = 1$ increment p by one.
 13. The awake sensors with $set = 1$ find p_{min} , the smallest p among themselves.
 14. $p_{min} - 1$ is transmitted to s_i , s_i sets p to the received value.
 15. If ($p \geq (n_1 + n_2)$) jump to Step 18.
 16. s_i wakes up and transmits $p + 1$ to sensor with index $i + 1$ and $set = 0$, and switches off.
 17. $i + +$, Goto Step 4.
 18. All sensors reindex themselves to p .
 19. End.
-

Figure 1. Pseudocode for MERGE (n_1, n_2)

Note that the above algorithm assumes that all data elements are distinct and have unique destinations. 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 algorithm can ensure that all elements have a unique rank. Each time a sensor which has not transmitted its result yet, hears a value equal to its own, it increments the rank counter. This ensures that the rank of all data elements is distinct.

The algorithm runs in time $O(n)$. Note this is the lower bound as each sensor must transmit its value at least once. Thus, the algorithm is time-optimal. There are n sensors and each sensor is awake for $O(n)$ timesteps. This implies that the energy complexity of the algorithm is $O(n^2)$. The algorithm is energy-balanced, but is not energy-optimal. The lower bound for energy dissipation for sorting is $O(n \log n)$ as discussed in Section 5.2.

5.2. Energy-Optimal Implementation

An energy-optimal implementation for sorting in a single hop sensor network can be achieved by adapting the merge-

MSORT (n)

1. If ($n \leq 2$) sort the elements using two broadcasts.
 2. If ($n > 2$) execute the following steps.
 3. All sensors initialize variables $set = 0$ and $done = 0$.
 4. All sensors with index $j > \lceil n/2 \rceil$ reindex themselves to $j - \lceil n/2 \rceil$, and switch off.
 5. Recursively call **MSORT** ($\lceil n/2 \rceil$) on awake sensors.
 6. All awake sensors assign $set = 0$ and $done = 1$.
 7. Wake up all sensors.
 8. All sensors with $done = 1$ switch themselves off.
 9. Recursively call **MSORT** ($n - \lceil n/2 \rceil$) on awake sensors.
 10. All awake sensors assign $set = 1$.
 11. Wake up all sensors.
 12. **MERGE** ($\lceil n/2 \rceil, n - \lceil n/2 \rceil$).
 13. End.
-

Figure 2. Pseudocode for MSORT (n)

sort algorithm discussed in [3]. The implementation of the algorithm in a single hop sensor network is described in Figure 2.

1. **MERGE** (n_1, n_2): We first discuss implementation of procedure **MERGE** (n_1, n_2) as illustrated in Figure 1. This algorithm is used to merge two sorted sets of sensors S_1 and S_2 of size n_1 and n_2 respectively, into a single sorted set of size $n_1 + n_2$. Sensors in set S_1 are indexed from $1 \rightarrow n_1$ and have bit $set = 0$. Similarly, sensors of set S_2 are indexed from $1 \rightarrow n_2$ but have bit $set = 1$. At the end of the algorithm the two sets are merged into a single sorted set. The sensor with index j , where $1 \leq j \leq (n_1 + n_2)$ stores the data element of rank j in the merged set.
2. **MSORT** (n): Next we describe **MSORT** (n), which is an adaptation of the algorithm mergesort for a single hop sensor network. The algorithm partitions the sensors into two sets based on their index. The two sets of size $\lceil n/2 \rceil$ and $n - \lceil n/2 \rceil$ are sorted by making a recursive call to the procedure. The sorted sets are then merged together into a single sorted set by invoking **MERGE** ($\lceil n/2 \rceil, n - \lceil n/2 \rceil$).

Analysis of the Algorithm MSORT (n): Let $T(n)$ and $E(n)$ denote the time and energy complexity of the algorithm. If $n = 2$, each sensor transmits, receives and compares once. Thus, $T(2) = 3$ and $E(2) = 6$. Let us consider the scenario where $n > 2$. The time and energy complexity of the algorithm **MERGE** (n_1, n_2) is $O(n_1 + n_2)$. In the worst case a sensor in set S_1 compares its value with all the sensors in S_2 and remains awake for $O(n_1)$ time steps. For example, when the data elements of sensors in S_1 are larger than the

$INDEX_l(n)$

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

Figure 3. Pseudocode for $INDEX_l(n)$

data elements contained by elements in S_2 . This implies a sensor may be awake for $O(n)$ time steps in each recursive call to procedure $MSORT(n)$. Steps 5 and 9 are recursive calls on problem size at most $\lceil n/2 \rceil$. Thus we conclude,

$$T(n) = 2T(n/2) + O(n) = O(n \log n)$$

$$E(n) = 2E(n/2) + O(n) = O(n \log n)$$

The work complexity analysis of algorithms can be used to obtain bounds on energy complexity of an algorithm. Let $W(n)$ denote the work complexity of an algorithm, which represents the total number of operations performed by the algorithm. Since our model assumes a uniform cost model, each operation costs one unit of energy, and $E(n) = O(W(n))$. The lower bound on the work complexity for any sorting algorithm is $O(n \log n)$ as these many comparisons are required. This implies that $O(n \log n)$ is also the lower bound on the energy complexity for sorting. Thus, $MSORT(n)$ is energy optimal.

Note that the algorithm is not time-optimal (see Section 5.1) and is not energy-balanced. A sensor may be awake for $O(n)$ time steps in Step 12. There are $O(\log n)$ recursive calls to the algorithm, which implies that a sensor may be awake for $O(n \log n)$ time steps.

5.3. Energy-Balanced Energy-Optimal Implementation

The algorithm $MSORT(n)$ is not energy-balanced as the number of times a sensor is awake depends on the data value stored by it. In an energy-balanced algorithm, the number of wake ups at a sensor should be data oblivious. The algorithm $QuickSort(n)$ discussed in [3] has this property. However, this algorithm has time and energy complexity $O(n \log n)$ on average and $O(n^2)$ in the worst case. The worst case behavior can be reduced to $O(n \log n)$ by using the *median* as the partitioning element. In this section, we discuss an energy-optimal and energy-balanced implementation of the sorting algorithm described by procedure

$SHIFT_l(n, k)$

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

Figure 4. Pseudocode for $SHIFT_l(n, k)$

$BSORT(n)$ illustrated in Figure 6. We first describe the various subprocedures of the algorithm.

1. $INDEX_l(n)$: Consider a set of n sensors indexed from $1 \rightarrow n$. Let us assume k denotes the number of sensors with variable $l > 0$. The procedure $INDEX_l(n)$ is used to reindex the sensors with variable $l > 0$ from $1 \rightarrow k$. At the end of the procedure, the new index is stored in updated value of variable l . Details of the algorithm are presented in Figure 3. The time and energy complexity of this algorithm is $O(n)$ and each node is awake for at most $O(1)$ time steps. $INDEX_h(n)$ is analogous but operates on variable h in place of l .
2. $SHIFT_l(n, k)$: The procedure $SHIFT_l(n, k)$ moves data from sensors with variable $l > 0$ and index $j \leq n - k$ to sensors with $l = 0$ and $j \geq n - k + 1$. The pseudocode for the algorithm is described in Figure 4. Thus, if we 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 *rightmost* position. Steps 1 and 2 are used to identify and index the sensors that need to exchange data. At step 3 sensor n broadcasts the number of data exchanges required, and this value is stored in variable m . In step 4 the data is exchanged between the sensors.

The total time and energy for the algorithm is $O(n)$. Each sensor is awake for $O(1)$ time steps. The algorithm $SHIFT_h(n, k)$ is similar, but operates on variable h instead of l .
3. $BSELECT(n, i, d)$: The algorithm $BSELECT(n, i, d)$

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_j be id of sensor with index j .
 - At time $t = j$ sensor with id s_j wakes up.
 - It wakes and transmits to sensor with id $s_j - z/2$
 - Sensor with id s_j switches off and sensor with id $s_j - z/2$ reindexes to t .)
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. Rest of the 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 the sensors, restore old indices, 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). *INDEX* _{h} (n).
10. Sensor n broadcasts l if $l > 0$, and $n - h$ if $h > 0$.
 - This value is stored in all sensors in k . Note k represents 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.

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

is used to find the element of the i^{th} rank in a set of n sensors in an energy-optimal and energy-balanced manner. It is a recursive algorithm and d represents the depth of recursion at any instance. By choosing $i = \lceil n/2 \rceil$, this algorithm can be used to find the median element in a set of n sensors. Detailed analysis of this algorithm is presented in [13]. The pseudocode for the algorithm is illustrated in Figure 5, and discussed briefly below.

A time and energy optimal algorithm for rank selection can be designed by adapting the *SELECT*(n, i) [3] algorithm for a single-hop sensor network. The implementation can be achieved using Steps 2 to 12 of the algorithm *BSELECT*(n, i, d) and has been discussed in detail in [13].

The sensors are divided into groups of five each and sorted. The median of the medians of the sorted group are found by making a recursive call to the procedure *SELECT*($\lceil n/5 \rceil, \lceil n/10 \rceil$) and stored in variable x . This is used to partition the set into two sets. One set contains all the elements smaller than or equal to x . The size of this set is determined. If it is smaller than i , then the element is contained in this set else in the other set. The algorithm is recursively called on the appropriate set and the sensors in the rejected set are switched off. As discussed in [13], this algorithm is time and energy optimal but is not energy balanced. The sensor containing the element of rank i , remains active till the algorithm terminates, while the sensors in the rejected set are switched off earlier depending on the rank of their data element. Consider the scenario where the sensors are sorted and the algorithm is called to find the sensor with rank n . Sensor n will be awake for time $O(n)$, whereas sensor with rank one switches off after the first iteration.

Algorithm *BSELECT*(n, i, d) obtains energy-balancedness by shuffling the data after a constant number of recursive calls to the procedure. It has an additional parameter d , which measures the depth of recursion, and initially $d = 0$. The main observation to be made in procedure *BSELECT*(n, i, d) is that each recursive call reduces data set from size n to $7n/10 + 6 < 8n/10$ or smaller (see [3] for details). This implies that at recursion depth k , the data set is reduced to $(8/10)^k \cdot n$. For $k = 7$, the data set is reduced to $n/4$. This property is exploited to obtain energy balancing by shuffling the data between the sensors (Steps 1 and 12).

By calling procedure *SHIFT*(n, k, l) in Step 12, we ensure that at end of each recursive step only the sensors with the highest id in the awake sensors contain the active data set. At recursive depth of $d = 7$, the

data is shifted to lower id sensors that were switched off in an earlier stage. This ensures that no sensor is awake for more than $O(1)$ timesteps. A detailed proof has been discussed in [13]. The time and energy taken by this algorithm is $O(n)$. The algorithm is time and energy optimal, and energy-balanced.

BSORT (n)

1. If $(n \leq 2)$ sort the elements using two broadcasts.
 2. If $(n > 2)$ the following steps are executed.
 3. Find median m , using *BSELECT* (n , $\lceil n/2 \rceil$), 0).
 4. Broadcast m to all sensors, each sensor compares m with its data element v .
 5. If $v \leq m$ a sensor sets $l = 1$ else $h = 1$.
 6. Sensors are reindexed using procedures *INDEX_l* (n) and *INDEX_h* (n).
 7. Sensor n broadcasts l if $l > 0$, and $n - h$ if $h > 0$.
This value is stored by all sensors in variable k .
 8. All sensors set *done* = 0. Sensors with $h > 0$ switch off.
 9. Recursively call *BSORT* ($n - k$) on awake sensors.
 10. All awake sensors set *done* = 1. Wake up all the sensors.
 11. Sensors with *done* = 1 switch off.
 12. Recursively call *BSORT* (k) on the awake sensors.
 13. End.
-

Figure 6. Pseudocode for *BSORT* (n)

4. ***BSORT* (n):** Lastly, we discuss the energy-balanced, energy-optimal implementation of the algorithm *BSORT* (n) as illustrated in Figure 6. The set of sensors is divided into two sets, using the median as the partitioning element. The two sets are then recursively sorted.

Analysis of the algorithm BSORT (n): Let us assume that the time and energy complexity of the algorithm are given by $T(n)$ and $E(n)$. If $n = 2$, each sensor transmits, receives and compares once. Thus, $T(2) = 3$ and $E(2) = 6$. Steps 9 and 12 take time and energy $T(n/2)$ and $E(n/2)$. As discussed above, Step 3 takes time and energy $O(n)$. Simple analysis shows that rest of the steps also take time and energy $O(n)$. The time and complexity of the algorithm is given by:

$$T(n) = 2T(n/2) + O(n) = O(n \log n)$$

$$E(n) = 2E(n/2) + O(n) = O(n \log n)$$

No sensor is awake for more than $O(1)$ time steps in any single recursive call. There are $O(\log n)$ recursive calls to the algorithm. Thus, any sensor is awake for at most $O(\log n)$ time steps. The algorithm is energy-optimal and energy-balanced.

5.4. Implementation in a p -channel sensor network

Next, we consider the implementation of algorithm *BSORT* (n) in a p -channel, single-hop sensor network, where $p \leq n^{1-\epsilon}$, and $0 < \epsilon \leq 1$. In order to communicate, both the transmitting and the receiving sensors 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 without additional signalling.

We first discuss the implementation of the algorithm *INDEX_l* (n) on a p -channel sensor network (Note the analysis of procedure *SHIFT_l* (n , k) is similar).

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 $0 < \epsilon \leq 1$.*

An algorithm for a p -channel implementation of prefix sum discussed in [7] can be adapted for this algorithm. For simplicity we assume n to be divisible by p .

Consider the case, where $p \leq \sqrt{n}$. The algorithm *INDEXP_l* (n) implements algorithm *INDEX_l* (n) in time $O(n/p) + O(p) = O(n/p)$. Each sensor is awake for at most $O(1)$ time steps. The energy complexity of the algorithm is $O(n)$.

INDEXP_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 $(t = 1, t < p, t++)$
Sensor with index n/p in group $g = t$ 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.
-

Figure 7. Pseudocode for *INDEXP_l* (n)

Next we consider $p = n^{1-\epsilon}$, where $0 < \epsilon \leq 1$. Let $2^c = (1 - \epsilon)/(1 - 2\epsilon)$. Consider implementation of the algorithm *INDEXG_l* (n , p , c) described in Figure 8. Total time taken for the algorithm is $O(n/p + p^{1/2} + \dots + p^{1/2^c})$

$INDEXG_l(n, p, c)$.

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.
-

Figure 8. Pseudocode for $INDEXG_l(n, p, c)$

$= O(n/p + p^{1-1/2^c}) = O(n/p + p^{\epsilon/(1-\epsilon)}) = O(n/p + p^{(1-\epsilon)/p}) = O(n/p)$. The energy dissipation is $O(n)$ and each sensor is awake for at most $O(1)$ time steps.

Lemma 5.2 *The algorithm $BSELECT(n, i, d)$ can be implemented in a p -channel, single-hop sensor network of n sensors 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 $0 < \epsilon \leq 1$.*

The proof is by induction on the size of the problem n and a detailed proof has been discussed in [13].

Steps 2, 4, 6-8, and 10-11 are computed in time $O(1)$. Step 3 can be simulated in time $O(n/p)$ in a p -channel network as follows. Let a sensor with index j belongs to group g , where $g = (j - 1)/5 + 1$. 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, 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 procedures $INDEX_l(n)$ and $INDEX_h(n)$. Using Lemma 5.1, we conclude this step can be performed in time $O(n/p)$ and energy $O(n)$, with each sensor awake for $O(1)$ time steps. Step 12 calls $SHIFT_l(n, k)$ and $SHIFT_h(n, k)$. Step 1 of algorithm $SHIFT_l(n, k)$ is similar to function $INDEX_l(n)$ and can thus, be implemented in a p -channel network in time $O(n/p)$. 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 $m/p = O(n/p)$.

Steps 5 and 12 make recursive calls to the procedure $BSELECT(n, i, d)$ 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 $T(n/p) = O(n/p)$.

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

Theorem 5.1 *Given a single-hop sensor network with p channels and n randomly distributed nodes, sorting can be performed in time $O(n/p \log n)$ with energy $O(n \log n)$, and no sensor awake for more than $O(\log n)$ time steps. $p \leq n^{1-\epsilon}$, where $0 < \epsilon \leq 1$.*

Proof: Partition the set into groups of size n/p as follows. Find the median of the set using $BSELECT(n, (n/2 + 1), 0)$ and p channels in time $O(n/p)$. Next partition the elements into two sets, those with elements larger than median, and the others with elements smaller than or equal to the median. Assign $p/2$ channels to the first set and the remaining channels to the second set. Recursively repeat the above steps on both the partitioned sets in parallel (using distinct channels) till the size of the resultant set is n/p . The above procedure requires time $O(n/p \log p)$, energy $O(n \log p)$, and each sensor is awake for $O(\log p)$ time steps. Invoke procedure $BSORT(n/p)$ on each of the subsets. Since all the subsets have distinct channels, they can sort in parallel. Time taken is $O(n/p \log n/p)$, energy $O(n \log n/p)$ and no sensor is awake for more than $O(\log n/p)$ time steps.

Total time taken for the algorithm is $O(n/p \log n)$, energy $O(n \log n)$ and each sensor is awake for at most $O(\log n)$ time steps. \square

6. Simulation Results

In Section 5, we discussed three algorithms for sorting. The algorithms $MSORT(n)$ and $BSORT(n)$ are asymptotically energy optimal. $BSORT(n)$ is also energy balanced. The aim of our simulations is to compare the energy performance of these two algorithms.

The simulations were performed for problem size n ranging from 10 – 4000 for the following algorithms. (a) $MSORT(n)$ as discussed in Section 5.2. (b) $USORT(n)$, which is $QUICKSORT(n)$ with median as partitioning element. The median is found using $SELECT(n, i)$ discussed in [3], and is thus unbalanced. (c) $BSORT(n)$ as presented in Section 5.3.

Three types of data sets have been used for comparing the performance of the three algorithms. The first data set contained already sorted data elements. In the second data set, the elements were reverse sorted, and the third data set

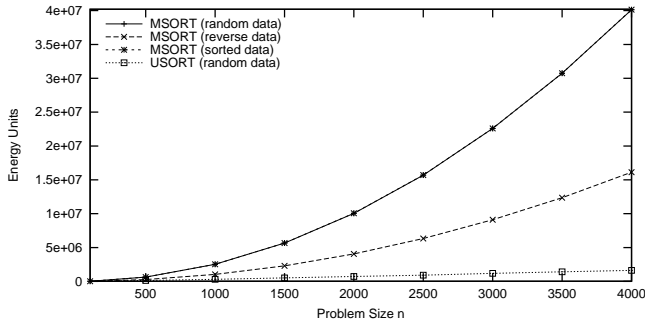


Figure 9. Energy Dissipation

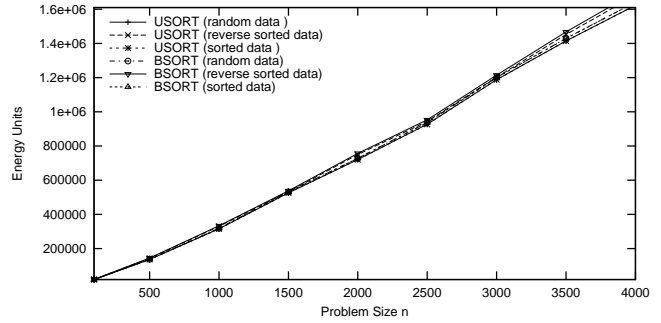


Figure 13. Energy Dissipation

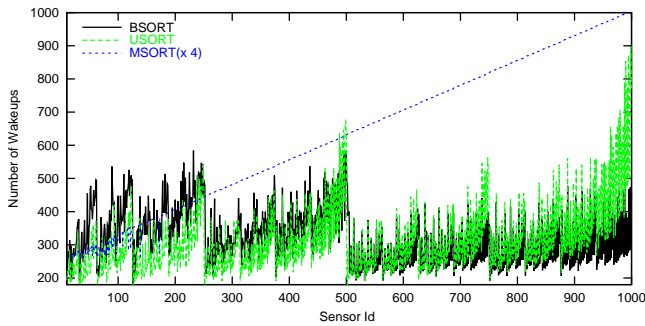


Figure 10. Number of Wakeups

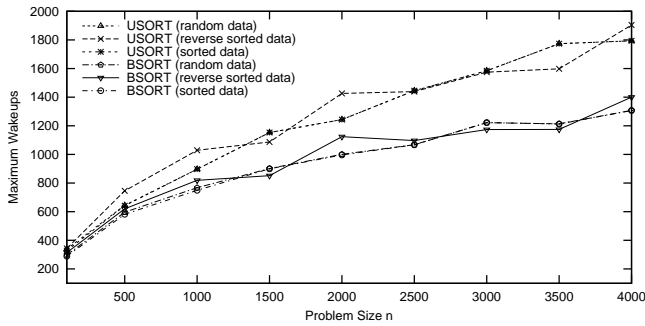


Figure 11. Maximum Wakeups

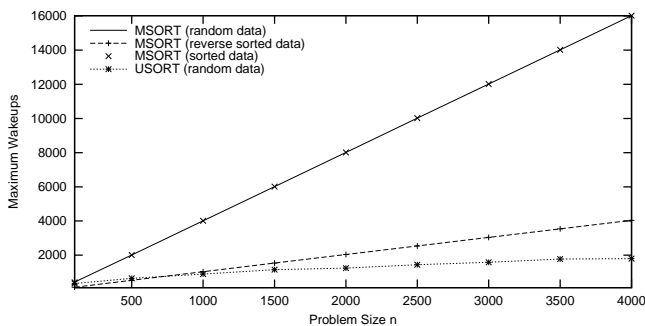


Figure 12. Maximum Wakeups

was randomly generated. The results presented in this paper have been averaged over 1000 iterations with 95% confidence interval and 0.1% (or better) precision.

We compare the energy dissipation for the algorithms for the three data sets. We observe that $MSORT(n)$ performs the worst (Figure 9). The energy dissipation using this algorithm is least if the data is reverse sorted and highest if the data is already sorted. Algorithms $BSORT(n)$ and $USORT(n)$ show little variation in results for the three data sets (see Figure 13). Moreover, the energy dissipation for the two algorithms is similar, which shows that data reshuffling overheads for algorithm $BSORT(n)$ are low.

Next we compare the energy-balancedness of the three algorithms. Figure 10 depicts the number of wake ups for various sensors in a network of 1000 nodes for randomly generated data set. Note the results for algorithm $MSORT(n)$ have been scaled down by a factor of four. We observe that the sensors in algorithm $MSORT(n)$ are not energy-balanced. A sensor with larger id is awake for a longer duration. The algorithm $BSORT(n)$ is most energy-balanced as the variation between the number of wake ups in various sensors is least. Lastly, we compare the energy-balancedness of algorithms for various problem sizes in Figure 11 and Figure 12. The results are illustrated by plotting the maximum number of time steps any sensor is awake for a given problem size for a chosen algorithm.

The results show that algorithm $BSORT(n)$ is most energy-balanced. The overall energy dissipation is also lower for this algorithm as compared to algorithm $MSORT(n)$.

7. Conclusion

In this paper, we discussed an energy-optimal and energy-balanced algorithm for sorting in a single-hop wireless sensor network. Along with overall system energy reduction, we emphasized on uniform energy dissipation among the sensors and defined the property of energy-balancedness.

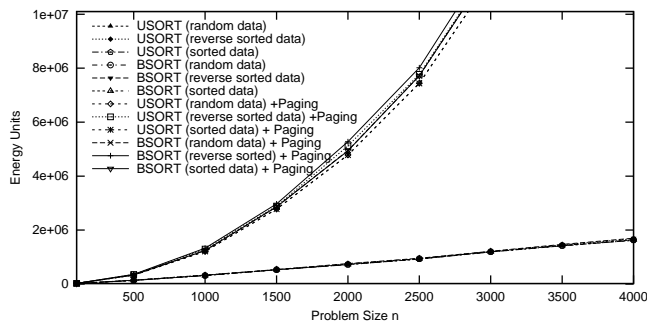


Figure 14. Paging Channel Overheads

Our analysis is for a single-hop network, and the size of these networks is limited by the transmission range of the radios. The results shown in this paper are for problem sizes larger than 80. Thus, they are applicable to either dense single-hop clusters in a multihop network, or to large single-hop sensor networks. A general belief is that sensor radios must be designed to be low power with short transmission range. However, it has been argued in [6] that if robustness is critical, increasing the radio range to facilitate single-hop transmissions, may reduce overall energy dissipation.

We have assumed a low cost paging channel for implementation of our algorithm. Figure 14 illustrates the energy overheads for paging, when the power of the paging channel is assumed to be one-hundredth of the communication channel. To save energy, the wake up mechanism can be implemented by equipping sensors with low-power hardware timers for self-initiated wake up. Our algorithms can be easily adapted to work with internal timers without the paging channel. However, it would increase the overall execution time of the algorithm and require more accurate time synchronization between the sensors. For example, consider execution time for Step 5 in *BSELECT* ($\lceil n/5 \rceil, \lceil n/10 \rceil, 0$). The time is $O(n/5)$, which implies that it is less than $cn/5$ for some constant c , but the exact time depends on the data. If only timers are used, Step 6 must be scheduled assuming the maximum time which is $cn/5$. On the contrary, with the paging channel the wake up can be scheduled immediately after completion of Step 5.

We presented an energy-balanced and energy-optimal algorithm for sorting that takes time and energy $O(n \log n)$. The time-optimal implementation [2] requires time $O(n)$ and energy $O(n^2)$. As our future works we would like to investigate if it is possible to design a sorting algorithm that is energy-balanced and energy-optimal but takes less time than $O(n \log n)$.

References

- [1] P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In *International Conference on Mobile Data Management (MDM)*, January 2001.
- [2] J. L. Bordim, K. Nakano, and H. Shen. Sorting on single-channel wireless sensor networks. In *International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN)*, May 2002.
- [3] T. H. Cormen, C. H. Leiserson, and R. L. Rivest. *Introduction To Algorithms*. The MIT Press, 1990.
- [4] 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.
- [5] 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.
- [6] B. Krishnamachari, Y. Mourtada, and S. Wicker. The energy-robustness tradeoff for routing in wireless sensor networks. Technical Report TR02-01, Dept. of Electrical Engineering-Systems, University of Southern California, September 2002.
- [7] K. Nakano. Time and energy optimal list ranking algorithms on the k-channel broadcast communication model. In *International Computing and Combinatorics Conference (COCON)*, August 2002.
- [8] K. Nakano and S. Olariu. Energy-efficient initialization protocols for radio networks with no collision detection. *IEEE Transactions on Parallel and Distributed Systems*, 11:851–863, 2000.
- [9] 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*, 10:1276–1289, 1999.
- [10] K. Peng and J. Kieffer. Embedded image compression based on wavelet pixel classification and sorting. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2002.
- [11] C. S. Raghavendra and S. Singh. Pamas-power aware multi-access protocol with signaling for ad hoc networks. *Computer Communications Review*, July 1998.
- [12] M. Singh and V. K. Prasanna. System level energy tradeoffs 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.
- [13] M. Singh and V. K. Prasanna. Optimal energy-balanced algorithm for selection in a single hop sensor network. In *International Conference on Communications (ICC), Workshop on Sensor Network Protocols and Applications*, May 2003.
- [14] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *International Conference on Mobile Computing and Networking (MOBI-COM)*, July 2001.