Data Collection with Battery and Buffer Consideration in a Large Scale Sensor Network

Deji Chen*, Aloysius K. Mok†, Jianliang Yi*, Mark Nixon*, Tom Aneweer*, and Rusty Shepard*  
*Emerson Process Management  
†Department of Computer Sciences  
12301 Research Blvd. Bldg. III  
Austin, TX 78759, USA  
{mok, jyi}@cs.utexas.edu  

Abstract

In a pure sensor network thousands of sensors are distributed over a large area. Sensor data is relayed from sensor node to node until it reaches the edge node, where it is then routed by wire to the host. In this paper we study the effect of data collection on the battery and buffer of a sensor node, and on the overall sensor network. We determine the relationship among battery life, sampling frequency, buffer size, etc. We claim that using identical sensors throughout a large scale sensor network may not achieve the best result. We suggest some guidelines for deploying a large scale sensor network. We then propose routing algorithms that are robust, truly distributed, and efficient in utilizing the sensor network.

1. Introduction

Large scale sensor network or mesh network could be applied to many areas. It connects battery-powered small sensors with low power consumption. With low sampling rate and outdoor environment, its wireless sensors could scatter over several acres or more and operate on battery for years.

Battery life and buffer size are two major challenges in a sensor network. In a large scale sensor network, it is difficult to replace battery and increase on-sensor memory. For the concept of smart dust, sensors are scattered around an interested area and used until their batteries die. Then they are simply left behind, never being re-used again. In future weather forecast, we could spray into monitored sever thunderstorm atmosphere a cloud of sensor dusts that could float in the air. In many cases, it may be difficult to deploy wire powered data collection device within the monitored area. To maintain a long working life, a sensor should sleep whenever possible and only wake up to sense and transmit data. Another feature of large scale sensor network is so called multi-path in which data from a sensor could reach destination via different hops of sensors. A sensor relaying data for other sensors needs buffer. The size of total data generated from the network is limited by sensor buffer sizes.

In this paper we consider data collection from a large scale sensor network. We shall discuss the limits imposed by the very nature of the sensor network. We assume a large scale area where sensors are evenly distributed. We shall show that the sensors will run out of battery earlier at the edge than those towards the center. Larger buffers are also required for edge sensors. Based on this and the fact that it is not possible for a sensor to have the knowledge of the whole network and routing information, we propose data transmission algorithms that are localized and also have good global performance.

We provide some assumptions and define some symbols in the next section. In Section 3 we draw some results based on the assumptions and give some suggestions based on the results. We then propose the data transmission algorithms in Section 4. Section 5 talks about ZigBee. Section 6 is the experiment. We talk about related work in Section 7. Section 8 is the conclusion.

2. Some assumptions and definitions

We make some assumptions. The results based on these assumptions provide useful guidance for sensor network in general.

The sensors and the sensor network The sensor network samples data and send to a central host that resides outside the sensed area. The sensor network is large enough. The sensors are mass produced, mass
configured, and deployed evenly over a large area. The sensors are thus identical with batteries and buffers. To save energy, sensors should generate data when necessary. Sensor data should be event triggered rather than time triggered. It may happen that at some period sensors in one dynamic sub-area generate more data than those in another static sub-area. If we assume that, over the long run, the monitored area has uniform dynamics, then the data generated should be evenly distributed throughout the sensor network as well. We could assume a constant average data rate per sensor throughout the network.

The power consumption The power consumption of the battery consists of two parts, one to maintain liveliness, and the other to transmit data. We assume a constant energy rate throughout the sensor’s life time to keep it alive. We assume constant energy consumption per data size for sending and receiving data. The overhead energy when transmitting data is different with data package size. It is better if a data package is fully packed with data. Given large scale network, we assume that there is always enough data to be packed into a data package. We shall assume full data package in our discussion and divide overhead energy among each data unit. Thus we could derive our value of energy per data size by dividing the energy of transmitting one full data package by the package size.

Data transmission We assume that the data sensed are transmitted from sensor to sensor until it reaches the edge of the network, where the data will be picked up by wired edge nodes and routed to the host of the sensor network. Data will not include any routing information except the source address. Because the sensors are low power devices, we assume that a sensor only knows its neighbors. It also knows from which neighbors it receives data and to which neighbors it sends data. This paper will not address the issue of establishing the sensor network. So we shall assume that a sensor’s neighbor configuration is already set up. We shall call a sensor a node in the context of relaying data.

Now we define some symbols.
- \( d \): the average sensor density of the network.
- \( E \): the total battery energy of a sensor.
- \( L \): The time length when the sensor is alive.
- \( W_0 \): the constant energy rate to keep the sensor alive.
- \( W_1 \): the constant energy consumption per byte for data sensing and sending.
- \( W_2 \): the constant energy consumption per byte for data relaying.
- \( r_1 \): the sensor data generation rate.
- \( r_2 \): the data rate a sensor has to relay.
- \( D \): the maximum wireless data transmission distance a sensor supports.
- \( B \): the data transmission bandwidth of a sensor.
- \( q \): the percentage of time a sensor uses to send out data.
- \( b \): maximum data buffered in a sensor.
- \( m \): the number of a sensor’s neighbors from which it receives data.
- \( n \): the number of a sensor’s neighbors to which it sends data.
- \( G \): the graph representing the sensor network.
- \( G_d \): The directional graph of \( G \) in which the edge goes from the sending sensor to the receiving sensor.

The symbols will be introduced in different sections as they are used.

3. Some results

In this section we shall derive some results for large scale wireless sensor networks. We first look at how long a sensor can last in Subsection 3.1, then how much data could accumulate in the sensor buffer in Subsection 3.2. Subsection 3.3 will discuss the results and suggests guidelines for large scale sensor networks.

3.1. Sensor lifetime estimation

Let’s look at sensors in a circular band within the network as is shown in Figure 1. The radius of the inner circle is \( R \). The width of the band is \( D \). Any data generated within the circle has to be relayed by sensors within the band to reach outside due to the transmission limit \( D \). While \( D \) is fixed, \( R \) ranges from 0 to the radius of the whole sensor network. Now let’s calculate the average life time of a sensor within the band.

![Figure 1: Sensor Lifetime Calculation](image)
The sensor within the band spends energy in three ways, keeping alive, generating data, and relaying data. There are \( \pi R^2 d \) sensors within the inner circle. The total rate of data they generate is \( \pi R^2 dr_1 \); there are \( \pi(R+D)^2-\pi R^2 \) sensors within the band. The data relay rate of each sensor averages \( r_2 = \pi R^2 dr_1/((\pi(R+D)^2-\pi R^2)d) = r_1/(1+D/R)^2-1 \). The average lifetime of a sensor within the band is \( L = E/(W_0+W_1 r_1+W_2 r_2) = E/(W_0+W_1 r_1+W_1 r_1/((1+D/R)^2-1)) \).

It is expressed in Equation 1.

\[
L = \frac{E}{W_0 + r_1 \left( W_1 + \frac{W_2}{(1+\frac{D}{R})^2-1} \right)}
\]

Figure 2 is the graph of \( L \) versus \( r_1 \) and \( R \).

3.2. Sensor buffer usage estimation

Let’s look at a sensor in the band in Figure 1. It has to transmit two types of data, that from itself and that it relays. We assume the sensor has no control over when the relayed data will arrive, but it schedules when and how to transmit data out. A sensor should be able to send out any data eventually as long as its average output rate is bigger than data rate. Due to the irregularity of how the data is generated, buffer has to be used to store it until bandwidth is available to transmit it. A sensor will transmit data at the rate of \( B \) when it is active, but it will sleep most of the time during which no data could be transmitted. It saves the battery life during sleep. It also saves energy by combining data into big packages.

We have studied in [5] the effect on buffer due to data generation irregularity, where we assumed a constant sending rate. In this paper we have to assume a constant data generation rate due to the uncertainty of the arrival of the data a sensor has to relay. To estimate the buffer usage, we assume that the sensor will send out all accumulated data each time before it goes back to sleep. Let \( t_{max} \) to be the maximum time a sensor sleeps. On average the data accumulated through \( t_{max} \) is \( b = (r_1+r_2)t_{max} = (r_1+r_1/((1+D/R)^2-1))t_{max} \).

It is expressed in Equation 2.

\[
b = r_1 + \frac{1}{1+\frac{D}{R}} - 1 \]

Figure 3 is the graph of \( b \) versus \( r_1 \) and \( R \).

We could also calculate the relationship between \( q \), the percentage of time a sensor uses to send out data, and \( R \). Assuming no data loss, the sensor sends out the same amount of data it has. So we have \( qB = r_1 + r_2 \). It is expressed in Equation 3.

\[
q = \frac{r_1 + \frac{1}{1+\frac{D}{R}}}{B} - 1
\]

Figure 3 is also similar to the graph of \( q \) versus \( r_1 \) and \( R \).

3.3. Analysis of the results

The obvious problem of using identical sensors is that they will die out sooner around the edge than at the center. Those sensors have to have bigger buffer to hold data, sleep less, and spend more time to relay more data. For sensors that do not relay data, they will live the longest at \( L = E/(W_0+W_1 r_1) \). \( L \) decreases when \( R \) increases or \( r_1 \) increases. \( r_1 \) has an upper limit in practice while people want to stretch \( R \) as large as possible. \( L \) approximates \( 0 \) as \( R \) increases. We either have to use more powerful batteries for sensors near the edge, or replenish sensors in those areas more often than the center.
Another issue is the buffer. With identical buffer sizes, we shall have sensors in the center with unused buffer and sensors near the edge starving for extra buffer. The good solution is to provide different buffer sizes for different sensors. Replacing sensors does not help.

There are many factors limiting the scale ($R$) of a sensor network, such as expected lifetime ($L$), data sampling rate ($r_1$), wireless bandwidth ($B$), and the imbedded buffer size ($b$). $L$ limits $R$ through Equation 1; $r_1$ limits $R$ through Equations 1, 2, and 3; $B$ limit $R$ through Equation 3 where $R$ is maximized when $q=1$; $b$ limits $R$ through Equation 2.

Equations 1, 2, and 3 provide some guidance when deploying a large scale sensor network. For example, if budget and area is fixed, we could consider reducing the sampling rate to achieve the targeted network lifetime.

We could deploy sensors more densely as $R$ increases to achieve a constant $r_2$, which results in constant lifetime, buffer usage, etc throughout the network. Let the average sensor density within the inner circle to be $d_0$; the average sensor density within the band to be $d_1$. We have $r_2=d_0/r_1((1+D/R)^2-1))$. For $r_2$ to be independent of $R$, $d_1/d_0$ must be a multiple of $1/((1+D/R)^2-1))$. As $R$ increases, $d_1/d_0$ must increase more than linearly. In other words, to keep common use of the sensors, the density of the sensors must increase at a rate more rapidly than linearly as $R$ increases.

In deriving the results, we do not require that the band in Figure 1 is centered in the network area. The data generated within the band has to cross the band to reach the destination. If the band is not in the center, then some sensors could relay more data than the others. Furthermore, we assume that any sensor in the band relays data. This may not be true in many situations. For example, edge nodes may be only deployed at some circumference sections of the network. Sensors at the edges where no edge nodes are deployed could not relay data. If only a fraction of sensors in the band could relay data or if the data is routed circuitously rather than straight to the edge, their life time will be decreased proportionally. So our results should be interpreted as the average in the optimal sense. In real applications the battery will last shorter.

We considered the case of a two dimensional area. In many applications, such as weather monitoring, sensors will be deployed in a three dimensional area. For this case the band becomes a hollow ball. The lifetime and buffer problem will be even worse, at a cubic rate versus square rate. Equations 3, 4, and 5 become $L=E/(W_0+W_1r_1+W_2r_1/(1+D/R)^2-1))$, $b=(r_1+r_1/((1+D/R)^2-1))t_{max}$, and $q=r_1(1+1/((1+D/R)^2-1))/B$.

4. Data routing algorithms

We analyzed a sensor network on an average case. If no data is relayed, the sensors will die of natural course, consuming energy to keep alive and sample data. Different data relaying method dictates the deviations of sensor live times. If data from one area is relayed via selected fixed paths to another area, then sensors on those paths will die sooner, and leave the areas covered by those sensors to be less sampled than other areas. We think it is beneficial if sensors die evenly. In other words, we think it is beneficial if data is relayed evenly among the sensors.

It is very difficult to have a global routing policy or a global controlling node. The challenge is how to achieve even routing when a sensor only knows about its neighbors. We propose local routing algorithms in the next subsections, one with request rejection and one without request rejection. We shall argue that the algorithms work and provide good global result.

Finally, we discuss how an initial routing configuration could be established.

4.1. The algorithm with request rejection

We propose a routing algorithm that is localized yet works well in the global scale. Each node routes data independently. A node only knows about its neighbor nodes that it could communicate with. Between two neighbor nodes, we only allow one-directional data transmission. There is no need to send the same data back and forth between two nodes. Figure 4 shows the configuration information for a node. It has $(m+n)$ neighbors. It receives data from $m$ neighbors and sends data to $n$ neighbors. We call them sending neighbors and receiving neighbors respectively.

![Figure 4: Routing configuration](image)

Here is the algorithm.
1. The node accepts any data from its sending neighbor.
2. The node sends data equally among its receiving neighbors. It could rotate through
its neighbors, sending a fix amount of data to each one per transmission period.
3. If any of its neighbors dies, it is remove from the neighbor list. If it is a receiving neighbor, the data will then be sent equally among the remaining neighbors.
4. If all of its receiving neighbors die, the node will request a direction reversal to all its sending neighbors. The request asks them to change to receiving neighbors. If a neighbor accepts the request, it then becomes a receiving one; if a neighbor rejects the request, it is removed from the neighbor list.
5. If all of its receiving neighbors die and all its sending neighbors reject reversal request, the node has nowhere to send data and ceases functioning by declaring itself dead.
6. A node accepts reversal request if it still has other receiving neighbors.
7. A node rejects reversal request if the requesting node is the only receiving neighbor it has. After rejecting the request, a node considers this neighbor dead and starts the procedure described in above Step 4.

Reversing sending node to receiving node may sound strange. You do not want to send data further away from the destination. But this may be the only way if the shorter path is broken by the dead nodes. In many cases, reversing the direction will cause data to be routed to another edge node, which may not necessarily be too costly.

There is a nice property of the algorithm. It will not produce loop in which data travels in a circle.

Data travels from its source to an edge node, then eventually to the host. The sensor network, including the edge node, is a directional graph \( G \) in which each sensor is a node and each sending/receiving pair represents a directional edge from the sending node to the receiving node. A data path in the sensor network corresponds to a directional path from the source to an edge node in \( G \). The path should not contain loops. Data traveling in a loop gains nothing and wastes energy. If no mechanism exists to break the loop, a single package could drain out batteries of the nodes in the loop. If no directional loop exists in \( G \), then no data will travel in a loop in the sensor network. We shall briefly prove that our algorithm will not create new directional loops if \( G \) does not contain a directional loop.

The algorithm will change \( G \) only in Step 4, when it either changes an edge’s direction, or removes an edge. Removing an edge will not introduce new directional loops. If the algorithm introduces a new directional loop by reversing an edge, the edge must be part of the new directional loop. This is impossible because the new edge has not predecessor edges. The node requesting the reversal will have no edge pointing to it afterwards. So we claim that if \( G \) does not contain a directional loop, no new directional loop will be created by the algorithm. As a matter of fact, the algorithm could break existing directional loops. In practice, a directional loop drains battery and the nodes on the loop most likely will die early and trigger Step 4. The algorithm turns to stabilize the network overtime.

One problem with this algorithm is that a node may declare itself dead even though it could reach an edge node. For example, consider a \( G \) of 6 nodes: \( E_1 \leftarrow A \leftarrow B \leftarrow C \longrightarrow D \rightarrow E_2 \) where \( E_1 \) and \( E_2 \) are edge nodes. After \( A \) dies, \( B \) has no receiving node. \( B \) sends a reversal request to \( C \). Since \( B \) is the only receiving neighbor of \( C \), based on step 7, the request will be rejected. Based on step 5, \( B \) will declare itself dead. Of course \( C \) will eventually send data to \( D \). Now \( G \) becomes: \( E_1 \) \( A \) \( B \) \( C \rightarrow D \rightarrow E_2 \). But actually \( B \) should be able to send data to \( E_2 \) through \( C \) and \( D \). This leads to the following algorithm.

### 4.2. The algorithm without request rejection

This algorithm is the same as the previous one except that a sending node never rejects reversal request. If it has no more receiving node after the request, it will in the next step initiate reversal request to all its sending nodes. A node never declares itself dead unless all its neighbors are dead.

Like the previous algorithm, this one will not create new loops. Besides, sensor data will be routed to the host as long as there is communication.

\( G \) represents the communications graph of the network. \( G \) is the same as \( G \) only that the edges have no direction. Initially \( G \) must be connected. Otherwise we could simply assume the connected sub graphs to be separated sensor networks. A connected \( G \) does not mean all sensors have directional paths to edge nodes in \( G \). A simple example is when a sensor has no receiving node in \( G \). We shall prove that our algorithm will resolve this problem eventually if we assume unlimited battery life.

Let’s call a node without receiving neighbor a sink node. Edge nodes are not sink nodes. We shall prove that no sink node exists if we run the algorithm long enough. By definition, our algorithm will turn all sending nodes of a sink node into receiving nodes. This may create new sink nodes in its neighbors, which may turn the original sink node back to sink node again. However, this does not happen forever. Let’s say the path from the sink node to an edge node in
non-directional $G$ has $i$ steps. The nodes on the path are $v_0, v_1, v_2, \ldots, v_i$, where $v_0$ is the original sink node, $v_i$ is the edge node. For $v_0$ to flip back to sink, $v_l$ has to become sink first. Once $v_l$ gets out of sink, $v_i$ has to become a sink before $v_l$ becomes a sink again. Similarly, $v_{i-1}$ has to become a sink before $v_{i-2}$ becomes a sink again. Since $v_l$ is never a sink, $v_{i-1}$ could only be a sink once. Consequently, $v_{i-2}$ could only be a sink twice, in the end, $v_0$ could only be a sink $i$ times, after which $v_0$ has at least a directional path to $v_i$. This argument does not care if $G$ has directional loops. If not, any data will be eventually routed to the host with our algorithm. If $G$ has directional loops, any node in a loop will never be a sink as it always has a receiving node along the loop. So assuming no battery depletion, our algorithm will get rid of all sink nodes regardless if there are loops or not in $G$. The is the second property of the algorithm: unless all communication paths from a node to the edge nodes are broken due to dead nodes, the algorithm will eventually route its data to the host. This is true even if there is no directional path to an edge node in $G$; this is true even if there are loops in $G$.

As nodes run out of battery, $G$ may become disconnected. This algorithm keeps a node alive as long as it has neighbors, even though it is in a disconnected subgraph. This consumes more energy than the first algorithm. Since $v_0$ could only be a sink $i$ times if no node dies, we could set an upper limit $l$. If a node switches in and out of sink for more than $l$ times, it declares itself dead. At this time the node in corresponding $G$ most likely becomes disconnected from any edge nodes. On the other hand, there may be no need to save the batteries on the subgraph, which may be discarded anyway.

We hope to have all nodes share the load of relaying data by sending data evenly to receiving nodes. This is probably better than sending to one receiving node over and over again until it dies or requests direction reverse. We could not say how good the algorithm is in this regard. However, it reduces the risk that one node gets too much data than its transmission limit $B$. By evenly distributing data, we also hope to run sensor batteries at an even rate. We think it is undesirable if a particular region loses data availability earlier than other regions because the sensors die out earlier. Another drawback of fixed path is that we have to re-establish a new path whenever a node dies.

Our algorithm does not attach routing information with the data, yet it guarantees the arrival of the data at its destination. This saves battery energy. The data contains only the source information and the sensed value. A node does not keep copies of relayed data. Data is lost only when the node with the data dies or declares itself dead.

4.3. Initialize the directional graph

Initializing the sensor network with a $G$ avoids the need to initialize the network with data paths. Both $G$ and routing table enable routing sensor data to the destination. Or we could say $G$ is a collection of many possible data paths.

As long as $G$ is connected, our algorithms could handle any kind of $G$. Still, it would be better if initial $G$ contains neither loop nor sink nodes. In this subsection, we discuss some approach to create $G$.

If a sensor has a unique number, a simply way to determine the edge direction is to use the ordinal values of the numbers. No loops will be created as the node numbers in any path either increases or decreases, never goes back to previous values. If we could deploy sensors so that any node has a receiving node if its number is not adjacent to that of edge nodes, we could also avoid sink nodes.

Different sources could be used to represent this unique number as long as we use it consistently among the sensors. It could be some hardware unique value like MAC (Medium Access Control) address. ZigBee network has up to 65,536 addresses. Sensor address does not reflect the sensor locations. A better value could be the measured distance from the area center or from the closest edge node. Those measured values may not be accurate, which is tolerable for this purpose. We need a consistent way to break the ties among measured values. One candidate would be the MAC address. If we deploy the sensors from center to the edge, or vice versa, we could also use the deployment time as the unique number.

5. ZigBee – a possible candidate for large scale sensor network

ZigBee [24] is so far the only standard for low power, long battery life wireless network. In this section, we talk briefly about its topology and how it could be used for large scale sensor network. ZigBee defines “Reduced Function Device” that does not relay data. There is a “Full Function Device” connecting to each reduced function device to relay its data. Routers and Coordinator are full function devices. Coordinator coordinates routers and devices. In small scale, routers are wire powered and wired to the host. Figure 5 shows three basic topologies in ZigBee. In a star topology, sensors are connected to a central router/coordinator. In a cluster tree topology, routers
form a tree. Sensors connect directly with tree nodes. In a mesh topology, stars and cluster trees are connected via their routers. A large scale ZigBee network is a mesh.

![ZigBee topology models](image)

**Figure 5: ZigBee topology models**

Our analysis applies to large scale sensor network such as smart dust, mote, etc. Tiny motes could be thrown into a large area to monitor the environment. Before these applications become practical, our results could be readily applied to guide the deployment of large scale ZigBee mesh where routers are battery powered. The routers correspond to the sensors in our discussion. Data from a router’s reduced function devices could be considered data generated from the router. When a router dies, all reduced function devices connected to it are considered dead. When a router’s reduced function device dies, its data generation decreases.

6. Experiment

We simulate our algorithms to see the statistics on sensor lifetime, buffer usage, etc. Our simulator assumes that sensors are evenly distributed in a round area of the radius $R$. Sensors within the distance $D$ (maximum wireless data transmission distance) to the edge are edge nodes. All nodes generate data at the rate of $r_i$, and relay data for neighbor nodes. The maximum sending rate is limited by the wireless bandwidth of sensors. To keep a sensor alive, a certain amount of energy needs to be consumed per time unit. Also to generate, receive and send data, a certain amount of energy will be consumed per data size (e.g. byte).

For sensor lifetime and its relationship with the distance to the center, we do the following experiment. Let $R$ be 5000 distance unit. Sensors in this area are classified into ten different groups, based on their distance to the center of the round area. The average lifetime of each group is calculated. From the analysis of the algorithms we know that sensors close to edge will die first. After that inner sensors will be disconnected and claim to die soon, even they still have energy. Or they may keep generating and relaying data in a disconnected sub-network without any edge nodes. In this situation, the maximum buffer size will go extremely large and sensors will use up all energy very soon. So in order to estimate their lifetime, we obtain statistic data at a relatively stable moment, before edge nodes start to die. Based on the energy that has already been consumed, we can calculate their expected lifetime. In order to evaluate the relationship of lifetime and data generation rate, we do the above experiment on different data generation rates (20, 40, 60, 80, 100 bytes per time unit).

The simulation results are in Figure 6. The $x$ axis represents different distance groups in the sensor network. The $y$ axis represents expected lifetime. Different curves represent different data generation rates. From the graph we can clearly see that sensor lifetime decreases when distance or data generation rate increase. This graph is based on the algorithm in Section 4.2. We get very similar results when using the algorithm in Section 4.1.

![Sensor lifetime](image)

**Figure 6: Sensor lifetime**

We also do experiment on maximum buffer size. The settings are the same as the above experiment for lifetime evaluation. Results are in Figure 7. Here the $y$ axis represents maximum buffer size. We can see that the maximum buffer size increases when distance or data generation rate increase. And we get similar results on $q$, the percentage of time a sensor uses to send out data.

![Maximum buffer size](image)
Figure 7: Sensor buffer usage

We also compare the difference between sending data evenly and unevenly to receiving nodes. As we expected, sending data evenly shows much better results in terms of maximum buffer size, since all nodes can share the load of relaying data. As described above, the following simulation data are obtained at relatively stable moments during the simulation. Here $r_1 = 100$.

<table>
<thead>
<tr>
<th>Distance Group</th>
<th>&lt;500</th>
<th>1000-1500</th>
<th>2000-2500</th>
<th>3000-3500</th>
<th>4000-4500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even</td>
<td>65</td>
<td>295</td>
<td>509</td>
<td>795</td>
<td>1024</td>
</tr>
<tr>
<td>Uneven</td>
<td>203</td>
<td>11412</td>
<td>20883</td>
<td>21031</td>
<td>20617</td>
</tr>
</tbody>
</table>

7. Related work

There has been much work on the capacity limit of wireless network lately [1, 2, 8, 10, 11, 13, 16]. Most of them show that the network capacity decreases as it scales up. The decrease rate varies for different topologies and measurements. Ours is the first to derive general relationships among several network parameters such as battery life, bandwidth, buffer size, etc. in randomly deployed larger scale sensor network where data is to be transmitted towards the edge. We do not consider exact model of the network. Our limits apply to any network topology.

[10] tries to answer the question “What is the operational lifetime of a particular wireless sensor network under the control of optimal power management schemes?” The operational lifetime is defined with regard to the percentage of failed nodes. Like our discussion, it considers all sensors as data source. It uses the concept of a cut similar to our band through which all data passes towards a single destination. [8] considers identical randomly located nodes. It studies the throughput capacity between any two nodes, which also decreases with scaling. [13] considers detail topology to derive bounds and bounds on one-one path. So is [2]. [1, 16] discuss fundamental capacity limits on real-time information transfer in multihop wireless networks.

It is intuitive that with identical sensors the ones towards the destination have to relay more traffic hence deplete more rapidly than other nodes. Our paper derives some mathematical equations to quantify the problem. In our model, even though the number of sensors increases as $R$ increases, the average lifetime still decreases at a fast rate. Researches like [19] try to get around scaling limit by sending a mobile unit through the sensor network to collect data, which avoids the sensors’ burden of relaying data.

Due to the limitations, routing becomes an important topic to make best use of a sensor network [2, 3, 4, 6, 15, 16, 17, 21, 22]. [2] analyzes the square grid and torus grid based networks and provides optimal routing algorithms for which the rate per node is equal to the network capacity. [15] discusses routing for real-time data and sampling frequency assignment based on its RICH architecture. It costs to maintain and re-establish data paths. In our algorithms, data is transmitted based on local information only and the data path is implicit.

There are other researches related to various aspects of our work. [12] etc. studies how to reduce $W_1$ and $W_2$; [9] studies how a sensor knows its location; [20] studies how a sensor knows about the time; [14, 18] discuss issue similar to $G$’s connectivity. In [23] all sensor data is transmitted to the center of the network where the host resides. This asks for less number of sensors towards the center to relay more data.

Our analysis is based on some assumptions, which simplifies the actual problem. Some of them are:

- We assumed constant energy consumption for data transmission between two nodes. The distance between two nodes also dictates energy level. This is considered in many literatures mentioned above. If $e(A, B)$ is the cost per data size from node $A$ to node $B$, Then $e(A, C)$ may be different from $e(A, B)$ because of the distance differences. And $e(A, C)$ is probably also different from $e(A, B)+e(B, C)$. On the other hand, longer distance usually means less hops on the data path, hence less energy consumption in this perspective.
- We did not consider cases where a wired edge node is intentionally deployed inside the monitored area to reduce the burden of relaying data by the outer circle sensors. This practice would be the recommended based on our results. We shall point out that in this case, our analysis could still be applied to the smaller areas that do not contain such wired node. [8] recommends breaking large network into smaller separate ones. Also the edge nodes may be deployed in only selected places around the sensor network, not necessary evenly around the circumference.
- The only failure for data transmission we consider is when a node dies. Lots of interferences need to be considered. Retry costs extra energy. We should point out that it
is commonly assumed that the energy consumption could be based on the amount of data transmitted.

- We considered only sensors that push data out. The dynamics could be more complex if the host is allowed to solicit data from a particular sensor, or the nodes could be actuators and the host issues control command to them.

8. Conclusion

In this paper we looked at the theoretical limitations a large scale wireless sensor network has. In this network sensor data is sampled and relay towards the edge, where it is picked up by wired edge nodes. We calculated the average life time and buffer usage of a sensor. We analyzed the results with some suggestions for deployment. We proposed local routing algorithms that perform nicely in the global scale. The algorithms drain power evenly among the sensors, are robust against initial network setup and sensor depletion. Actually, our algorithms do not require routing table or path. It only stipulates one directional data transmission between two adjacent nodes. The simulation supports our claims.

We compared our work with other researches in Section 7, where we also listed further improvements. Those would be the areas of our future studies.

9. References
