Clustering Protocols for Energy Efficiency in Wireless Sensor Networks

Andrew Kefalas (6170560) and Mahesh Bakshi(6722083)
Concordia University
Montreal, QC, Canada
Comp7251
Prof. Lata Narayanan

Abstract–**One of the main issues concerning wireless sensor networks is energy efficiency. There are many clustering protocols that seek to extend the lifetime of a WSN by introducing new ways of choosing cluster heads and forming clusters. In this paper we research single-hop and multi-hop clustering algorithms that propose different ways to extend system lifetime. We will focus on how each protocol selects the cluster head, how the cluster is formed, and how routing to the base station is achieved. We will then implement these protocols in a network simulator to see how long the network survives and compare the results against each other and against the results found in the literature covered.**

Keywords – **Clustering, energy efficiency, power consumption, wireless sensor networks, LEACH protocol, EECS protocol, EEUC protocol, EEDUC protocol,single-hop routing, multi-hop routing.**

I. INTRODUCTION

One of the main challenges associated with wireless sensor networks is energy efficiency because of the inconvenience or impossibility or recharging and replacing nodes. Many algorithms and protocols have been proposed but it has been found that cluster based algorithms have better energy utilization as compared with non-cluster algorithms [2]. The concept of a cluster in wireless sensor networks is to have a cluster head that is responsible for collecting and forwarding the data from its cluster to the base station. There are 'normal' nodes that communicate only with their cluster head and cluster heads who relay the information from the normal nodes to either other cluster heads or the base station [3]. The idea is that the data retrieved by the sensors in the same cluster will have a high correlation and therefore the cluster head can aggregate this data and reduce the communication load [6]. The problem with this is that being the cluster head requires more energy and so we must have a way of choosing a cluster head in such a way that we try to maximize the life of the nodes and the overall system lifetime. So the protocols are divided into rounds, where in each round a new cluster head is elected according to the algorithm and then data is exchanged [1].

In this paper we will cover some algorithms that propose to be energy efficient and increase overall system lifetime by implementing some different techniques. The main focus of this paper will be on cluster head election, cluster formation and routing in each algorithm. We will then implement these protocols using network simulation software to see which gives the longest system lifetime and compare the results of each algorithm against each other and against the research papers that proposed the algorithms.

The remainder of this paper is organized as follows. Section 2 describes one of the first clustering algorithms, LEACH, and discusses the two phases included in each round and how a cluster is formed. Section 3 introduces EECS, which is a protocol based on LEACH but introduces load balancing to improve system lifetime. Section 4 describes the multi-hop algorithm EEUC, and look at some characteristics and challenges connected to multi-hop algorithms. Section 5 introduces another multi-hop algorithm, EEDUC, which is heavily based on EEUC but introduces a waiting timer mechanism to choose 'better' cluster heads. Section 6 describes our implementation method and parameters of simulation and Section 7 analyzes our simulation results. Section 8 concludes this paper and Section 9 describes the contributions by each member.

II. LEACH PROTOCOL

The first and most basic protocol that we will examine to manage clusters of wireless sensor nodes is Low-Energy Adaptive Clustering Hierarchy (LEACH). The purpose of this protocol is to use cluster based routing and low energy medium access control to increase the system's lifetime [1].This algorithm is one of the first successful clustering protocols and is referenced by many later protocols.

Clustering algorithms are broken down into rounds, each round starts with a set up phase where clusters are created and cluster heads are elected, and then a steady-state phase where the data is transferred from the nodes to the cluster head and on to the base station [1]. Protocols begin with the base station (BS) broadcasting a 'hello' message so that each node can determine the location of the base station as well as the distance from itself to the BS based on the strength of the signal [6].
LEACH assumes that each node can reach the base station and that the distance to the base station can be computed based on the initial hello message's signal strength [2]. This first assumption is important because the cluster head will receive the data from its cluster and forward it directly to the base station which simplifies routing. This is known as a single-hop from CH to the BS, and therefore LEACH is a single-hop algorithm.

The first and most naïve way LEACH can elect a cluster head is to use random election for where each node in the cluster will take turns being the cluster head based on probability algorithm. In this algorithm, the same node cannot be cluster head twice until all other nodes in the cluster have been elected. So a sensor i elects itself to be a cluster head at the beginning of each round r (which occurs at time t) with probability $P_i(t)$ [1]. The probability is explicitly chosen so that the expected number of cluster heads will be equivalent to the number of clusters in the network. In addition we want each node in the cluster to take turns being the cluster head to ensure that the same node does not become head multiple times before others have had the chance to become heads [2]. So we say that each node chooses to become cluster head at round r with the probability of 0 if it was recently a head [1]. The reason we do not want the same node to be head multiple times is because being cluster head consumes much more energy than being a normal node, so if the same node was head multiple times it will lose battery life quicker.

The problem with the naïve method is that it assumes that all nodes started with equal energy and will expend energy equally (that is they will all transmit data each frame). However if there is an unequal distribution of energy then we can create a simple heuristic that chooses the node with the most energy remaining to be elected as cluster head. This can be done in LEACH by setting the probability of becoming cluster head as a function of nodes energy relative to the aggregate energy remaining in the network [1]. The problem with this algorithm is that each node must be able to know the total energy of all nodes in the network which adds hardware overhead. This is a technique that many algorithms will use however and is a general assumption with today's hardware.

Once a cluster head has been elected the next step is to form the cluster. The method of cluster formation presented here is the base for many algorithms that follow LEACH, including the ones covered in this paper.

The cluster formation process can be seen in Figure 1. To form the cluster, the cluster head broadcasts an advertisement message containing his node ID using CSMA and waits for join requests. Then non-head nodes request to join the cluster head that requires the minimum energy to send to, which it can infer from the advertisement message [2]. Once that is finished the cluster head sets up a TDMA schedule so that there is no collisions within the cluster and so that nodes can turn off when it is not their turn to send data [1]. Here there are two ways to save energy; nodes use the exact amount of power needed to transmit to the head based on the advertisement message, and nodes sleep when it is not their turn to in the cluster. Then the nodes reach the steady-state period where data can be sent.

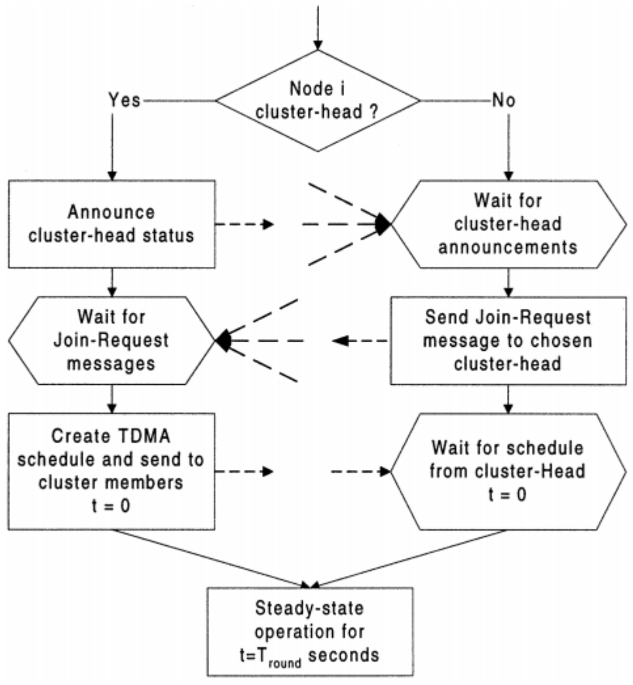
Node i cluster-head ?
Yes
No
Announce cluster-head status
Wait for cluster-head announcements
Wait for Join-Request messages
Send Join-Request message to chosen cluster-head
Create TDMA schedule and send to cluster members t = 0
Wait for schedule from cluster-Head t = 0
Steady-state operation for t=T_round seconds

Figure 1.LEACH Cluster Formation [1]

To prevent collisions and interference from neighbouring clusters, each cluster uses a unique spreading code from a predefined list so that the head can define which message to receive from the nodes in its cluster. Once the cluster head receives all the data to be sent, it compresses it and sends it to the base station using spreading codes and CSMA [1].

One of the reasons LEACH was so successful is that it drastically improved system lifetime of static clustering. The concepts that LEACH has introduced into clustering schemes greatly extended system lifetime and later algorithms also borrow some of these concepts and attempt to improve upon it. In our simulation results section we will see how LEACH compares to those newer algorithms.

III. EECS PROTOCOL

The next protocol we will examine is the Energy Efficient Clustering Scheme (EECS) protocol. This protocol is similar to LEACH but also introduces load balancing to balance the load in the nodes and especially cluster heads. It chooses cluster heads with a system that elects the cluster head as the node from the candidate pool with the maximum residual energy. According to Ye et al [5] it gives it about a 35% increase in time until first node death over LEACH.

Similarly to LEACH we make some assumptions about the network; each node can reach the BS, the nodes are stationary after being dispersed and each sensor can compute its distance to the BS based on the initial hello message's signal strength [5]. Therefore, from the first assumption, EECS is also a single-hop algorithm.

Cluster head election is done as follows: all nodes can become candidate nodes with a probability which is then compared to a threshold. The nodes that pass the threshold become candidate nodes and they broadcast a compete-head message to all nodes within range [5]. Each candidate node will check if there is another candidate node with more residual energy in its range. If there exists such a node, then that candidate node will not compete for cluster head anymore. If there is not a node with more residual energy in its range, then the candidate node will elect itself as cluster head [8].

Once a head is selected, it broadcasts the advertisement message to all the nodes and the normal nodes have to decide which cluster to join. Here is where EECS is different from LEACH. It makes the decision on which cluster to join based on balancing the workload for each cluster. It does this by assigning a cost value to each node to join a specific cluster head. This cost value is based on two values; the energy that needs to be consumed at the cluster head for receiving, aggregating, and sending data to the base station, call it F, as well as the distance of the cluster head from the base station, call it G. Based on these two cost values, a normal node will choose to join the cluster with the minimum cost [5]. In doing so, it also balances the work at the cluster head, especially those further away from the base station, because it considers how much work the cluster head will have to do based on the number of nodes in the cluster and the distance the data has to travel. According to Ye et al [5] an optimal value of T and W, which is the weighted factor for the tradeoff between F and G, can be computed to optimize this algorithm. Once a node has chosen which cluster head to join, the algorithm continues with the standard scheduling messages to form the cluster head exactly as in LEACH.

EECS introduces a little more overhead than the standard LEACH protocol. There are more messages being passed in the cluster head election phase, and there is more computation when choosing which cluster to join. The overall overhead however is negligible since the time of first node death will be increased and that is the goal of this algorithm. The claim of EECS is that it prolongs time until first node death by 35% over LEACH and has better energy utilization rate because it considers the load balance at cluster heads with its weighted function [4]. We will examine the results of the research paper and the results of our own simulation to see if they compare and to see if EECS does in fact improve time of first node death by 35% in our simulation results section.

IV. EEUC PROTOCOL

In this section we will present a different type of clustering algorithm; a multi-hop algorithm. The previous algorithms we discussed had one assumption that this algorithm does not; that each node can reach the base station. In multi-hop algorithms we do not make this assumption so cluster heads who are outside transmitting range of the BS must relay their data to other cluster heads until it eventually reaches the BS. Multi-hop routing raises a problem however, that the nodes closest to the base station will be under heavy traffic and thus die early, known as the hot spot problem [7]. The Energy Efficient Unequal Clustering (EEUC) algorithm is a multi-hop routing algorithm that seeks to solve the hot spot problem while at the same time extending system lifetime over single hop algorithms.

The EEUC algorithm is different from the previous algorithms we explored mainly because of its multi-hop routing. Multi-hop routing is more realistic than single-hop because WSN's can be dispersed across a vast amount of area and it is not realistic to assume that every node can reach the base station. Also multi-hop communication is usually more energy efficient than single-hop transmission because multiple small hops use up less energy than a single long hopbased on the nature of wireless communication and distance loss [8].

The problem with multi-hop routing is that it presents the hot spot problem. This problem states that the nodes closest the BS are not only sending their own data, but also the relayed data from all the other clusters in the network. So these nodes are overburdened as compared to other nodes and thus die faster [8]. The solution that EEUC proposes is to divide the nodes into clusters of unequal size with clusters closer to the base station having a smaller size than those farther away. The purpose of this is to reduce the energy used in intra cluster data exchange in those heavy traffic heads and thus keep more energy for relaying the data received by other cluster heads [8]. In addition to the unequal clustering, EEUC also presents an energy away routing protocol for choosing relayed nodes that prolongs system lifetime [6].

Figure 2 shows what the EEUC algorithm looks like. The circles represent the size of the cluster, which we can notice are smaller the closer they are to the base station. The dots are cluster heads and the arrows represent the relayed data. We can see that the cluster heads closest to the base station have the most data to send because they have to relay the data from the cluster heads further away and therefore these nodes consume more energy.

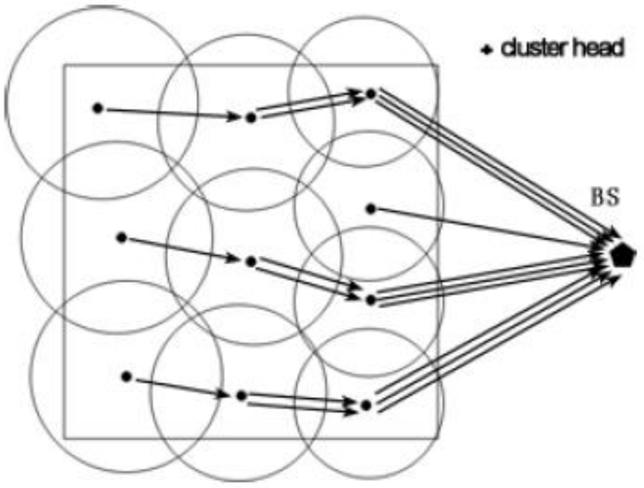

Figure 2. EEUC Algorithm [6]

Cluster head election in EEUC is done as follows. First candidate nodes are selected based on some probability T. Those nodes that pass the threshold become tentative cluster heads and have a competition range R which is based on the distance from the base station [6]. The closer any tentative node is to the base station, the smaller his competition range will be based on a specific formula that we will cover in our implementation section. Each tentative node also keeps a set of nodes S_{CH} which represents all the other nodes in competition range. Then all tentative nodes send out a compete_head_message which contains its competition range and residual energy. The cluster head will be elected as the tentative node with the most residual energy in its range. The node that finally becomes the cluster head will broadcast a final_head_message to inform all nodes that it has become the cluster head and all other nodes in the competition range will give up the competition and broadcast a quit_election message. The cluster head then broadcasts an advertisement message and normal nodes join the closest cluster head with the largest signal strength [6]. Scheduling is then done in the same fashion as described in the LEACH protocol.

In the previous algorithms we discussed, the cluster heads would receive the data from its cluster, aggregate it, and send it to the base station. In EEUC, the cluster heads have to choose which cluster heads to relay their data to in the multi-hop routing scheme. So cluster heads aggregate the data from the nodes and then relay it to other cluster heads. The first choice for a cluster head is whether to relay the data to another cluster head or to send it directly to the base station. So there is a threshold TD_MAX, where if the distance to the base station is less than TD_MAX, the head transmits the data directly to the base station. Otherwise the head needs to choose another CH to relay the data to [6].

The process begins with each cluster head broadcasting a message which contains its ID, residual energy and distance to the base station. Then each cluster head defines a set of adjacent nodes R_{CH} which represent the cluster heads that are 'adjacent' to the current cluster head and closer to the base station [6]. These 'adjacent' cluster heads can occur in EEUC since every node has a different competition range, so in CH1's range you could have CH2 but CH1 is not in CH2's range. In this case CH2 is 'adjacent' to CH1.

The cluster head will then choose to relay its data to the member of the adjacent set based on two values. The first is the residual energy of the relay node. The second is a distance cost based on the distance from the cluster head to the relay node to the base station [6]. This is important because if we did not have this second cost value the cluster head could continuously choose adjacent nodes with maximum residual energy but by choosing these nodes it will criss-cross across the network until it reaches the base station. Therefore the distance traversed by the data would be large. However with this added distance cost it will take into consideration a direct line to the base station even if the relay nodes have less residual energy and therefore reduce the distance travelled by the data. So the relay node is chosen as the node with the maximum residual energy from the smallest two distance costs in the set R_{CH}[6].

The main contribution of this paper is concept of unequal clustering to solve the hotspot problem and its energy aware multihop routing protocol. The small interval between time of first node death and time of last node death implies EEUC has solved the hotspot problem [6]. The results from the paper show that EEUC improves the network lifetime over LEACH both for the time until first node death and until the last node dies [6].One of the reasons that EEUC and other multi hop algorithms greatly increase system lifetime is that in single hop algorithms cluster heads send data to the base station in 1 hop which can be very energy consuming if the distance is far. In multi hop, there are more transmissions but of a small range so the energy require to send them is low, so there is less energy consumed at the cluster head [8].

In our simulation we compare EEUC to LEACH and to EECS to see which has better system lifetime; the results will be discussed in our simulation results section.

V. EEDUC

The next multi-hop algorithm we will cover is an Energy-Efficient Distributed Unequal Cluster protocol (EEDUC). This algorithm is based on the EEUC algorithm we just covered. It uses the same unequal clustering algorithm to solve the hot spot problem but presents a different way to select cluster heads which improves the energy consumed at cluster heads over EEUC.

The algorithm begins as the other algorithms we have discussed; when the network is initialized the base station sends out a hello message and each node receives it and calculates its distance to the base station based on the strength of the signal. Here is where EEDUC adds some steps to the initialization phase. Each node broadcasts an advertisement message to count the number of nodes within 1 hop range, which are called neighbouring nodes. The BS then sends a time unit for clustering to each node [8]. Each node then individually sets a waiting time as a function of the number of neighbourhood nodes, maximum sensor nodes and the time unit sent by the base station. After each node has decided on its own waiting time, it decreases the waiting time based on a synchronized node timer [8].

Figure 3 shows an example of a network with each node’s waiting time after the decrement operation has reduce the waiting timer of some nodes to 0. Notice that each node’s waiting timer is different, since each node will have different factors in the waiting time equation.

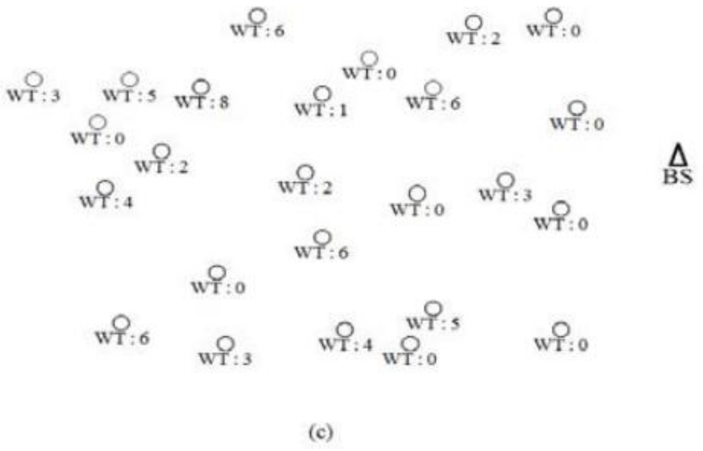

Figure 3. Waiting Time in EEDUC [8]

When the waiting time for a node become 0, it elects itself as cluster head and it broadcasts a hello message to a competition range which is a function of the distance to the base station, number of neighbourhood nodes and residual energy of the cluster head. Any normal node in the competition range then becomes a cluster member and sends a reply message back to the cluster head [8].

Figure 4 shows the resulting clusters formed by the waiting times in Figure 3. Notice that all the nodes with waiting time 0 become cluster heads. Also the cluster size is smaller as the distance to the BS is smaller. This shows that EEDUC implements the unequal clustering algorithm to solve the hot spot problem, exactly like EEUC.

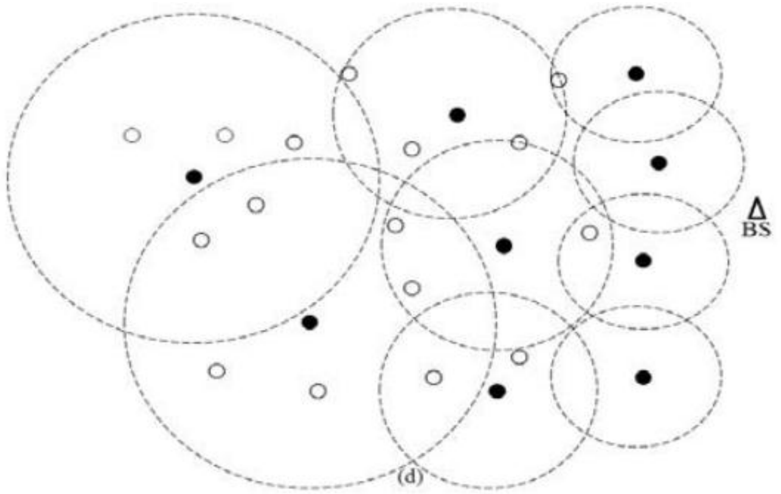

Figure 4. Resulting clusters from waiting time in Figure 3 [8]

The algorithm then sets up a TDMA schedule, and exchanges data in the same fashion as LEACH. In the next round the nodes will recompute their waiting time as a function of neighbouring nodes, the residual energy of the node, and maximum energy of the node [8]. This waiting time function is how EEDUC allows the election of cluster heads to be distributed across the system. Since the waiting time is based on residual energy and neighbouring nodes, a node that has low energy and many neighbours will have a high waiting time and thus is not likely to become the cluster head in that round [8].

In terms of choosing which cluster head to relay the data to, EEDUC uses a similar mechanism to EEUC, except it chooses the relay node from the set of neighbouring nodes instead of the set of adjacent nodes. The difference being that neighbouring nodes in EEDUC are specifically 1 hop neighbours, where adjacency nodes in EEUC are based on competition range. Also like EEUC, EEDUC does no data aggregation at the relaying nodes since the data correlation between different clusters is low [8].

As with other multihop algorithms EEDUC has much better energy consumption at cluster heads because of the short transmission ranges and therefore will have longer system lifetime than single hop algorithms. EEDUC also claims that the energy spent per cluster head is lower than EEUC which will prolong the lifetime of cluster heads and give greater overall system lifetime.

VI. Implementation

The purpose of the implementation part of our project is to recreate the algorithms covered above in our own environment. We will then compare our simulation results against each other, and against the results given in the research papers. For our simulations we used the network simulator tool NS-2 and the graphing tool xgraph. In this section we will outline the environments used for our tests, our parameters of simulation including formulas used, performance metrics, and challenges we faced during our implementation.

A. Setting up the simulation environment

For our simulation we decided to use the network simulator tool NS-2, more specifically the ns-allinone2.34 package found at [9]. We were able to successfully install it on both windows and Linux environment. To implement our clustering algorithms, we began with the LEACH patch that is available at [10], and is directly referenced in [1]. This patch had several errors installing and as such we were only able to install it on a Linux environment which is where we ran all our tests. This patch gave us a good framework to begin our implementation and using this as a base we made some edits to achieve our LEACH algorithm. Since EECS and LEACH are both single hop algorithms we were able to reuse most of the code to implement EECS while adding in our own code. Then using a mix of code from both LEACH and EECS we were able to construct a basis for our multi-hop algorithm, EEUC. The installing and change steps that are needed to run our code are documented in detail in Appendix A.

For our graphs we used the xgraph tool which comes in with the ns-allinone package and using scripts we extracted the output from our simulation and constructed the graphs from those outputs.

B. Parameters of Simulation

We make the same assumptions in our simulation as in the papers; in single hop algorithms all nodes can reach the base station, all nodes can calculate distance to transmitter based on the received message, every node begins with the same starting energy and that every node is static once it is placed. We ran our tests under 15 different topologies, which were each randomly generated using an NS-2 script. Below are two tables outlining our parameters of simulation and formulas used. Each parameter and formula and their values are then discussed below.

Parameter	Scene
Area:	100 x 100
Number of Topologies:	15 (5,5,5)
Number of nodes:	75,100,150
Initial energy:	2J
Threshold:	.95
Control Packet size:	25 bytes
Data Packet size:	500 bytes
Energy loss over distance formula:	E + (distToBS/MAXDist)^2*E

Figure 5.Parameters of Simulation

Parameter	Formula
LEACH Probability formula:	k / (N - k mod(r,N/k))
Competition Range in EECS:	$\sqrt{(A \div (\pi \times koptimal))}$
Weighted Cost Function in EECS:	Wxf(d(Pj, CHi))+(1-w)xg(d(CHi, BS))
Competition Range in EEUC:	(1-(((dmax- dist)/(dmax-dmin))*c))*RMAXcomp
TD_MAX:	150/2

Figure 6. Table of Formulas used in code

Area: We chose our area to be 100 x 100 and vary the number of nodes to test different network densities.

Number of Topologies, Number of nodes: We had 5 different topologies with 75 nodes, 5 different topologies of 100 nodes and 5 different topologies of 150 nodes. This is a total of 15 different topologies, each of them in a 100 x 100 area. We ran each topology at least once for each algorithm, 15 x 3 = 45 runs. On average let us say the length of a simuation was 30min so 45runs x 30min = 22.5 hours of simulation.

Initial energy: We chose the initial energy to be 2J because this is what is stated in [1] and [10]. All nodes begin with this amount of energy.

Threshold: Threshold was a very hard value to choose because of the impact of threshold on network lifetime observed in [5]. If we wanted to achieve optimal system lifetime, we would have had to change threshold in each algorithm, for each density. Since our goal was to compare the algorithms to each other and not achieve optimal lifetime in each topology, we kept threshold constant throughout all our tests. We experimented with several values but saw that .95 threshold gave us the most consistent and comparable results.

Packet Size: Packet size taken from the LEACH patch [10].

Energy loss over distance formula: The energy loss formula we computed ourselves because initially the code would take away a fixed energy value, no matter the distance of transmission. So we constructed an exponential function that would remove a fixed amount of energy plus more if the transmission range was longer, and less if it was short.

LEACH Probability formula: The probability of a node becoming a cluster head in LEACH is described by this formula from [1], Pi(t) = k / (N - k mod(r,N/k)). k is the expected number of clusters per round, N is the total number of sensor nodes in the network and r is the number of rounds that have already passed. We use exactly this formula from the LEACH patch [10].

Competition Range in EECS: In EECS every node has the same competition range, so we need to calculate the range for all nodes once. According to [5], we must use the formula $\sqrt{(\mathrm{A} \div (\pi \times \mathrm{koptimal}))}$ where A is the area of the network and koptimal is the optimal number of clusters. From the LEACH paper and code [1][10] we assume koptimal = 5 and the area of our network is 100x100 so the result of our formula ≈ 25m competition range.

Weighted Cost Function in EECS: Each node in EECS uses this formula to balance the workload at the cluster head and uses it to choose the 'best' cluster head to join when it receives multiple cluster head advertisement messages. Here we use a simpler version of the formula presented in [5]. Our formula is 0.8*DistanceFromMeToCH + 0.2*DistanceFromCHtoBS. We choose w=0.8 based on our experiments since no formula is given to compute w.

Competition Range in EEUC: In EECS every node has the same competition range but in EEUC each node calculates its own competition radius based on its distance to the base station. We use the same formulas as presented in the table which is taken from [6]. RMAXcomp is the maximum size of a node's competition range. In [6] they have it as 90m in a 200 x 200 area, so we divide by half to fit our area. So our RMAXcomp = 45m. dmax is the maximum distance between any two nodes in the topology, dmin is the minimum distance between any two nodes in the topology, and dist is the distance from this node to the base station. These are computed by our algorithms when we read the topology file so they are already available to use.

TD_MAX: There is no specific formula for TD_MAX, so we based ourselves off of the simulations done in [6]. They have an area of 200 x 200 and use TD_MAX as 150m. Since in our simluation we have an area of 100x100 which is half of theirs, we decided to use a TD_MAX of 75m which is half of theirs.

Finally, our performance metrics were time of first node death and overall system lifetime. These are the main two factors in determining system lifetime. In our simulation results section we will present the data accumulated from our simulations, analyze the results and make comparisons to the results presented in the papers.

C. Challenges

Our first and main challenge was installing NS-2 and the LEACH patch. Installing NS-2 went well and we were able to install it on both Windows and Linux environments. However the LEACH patch was severely out of date and every time we tried to install it, we had different errors. Our solution was to search for these errors one by one and solve them individually until it worked. Finally after days of working on it, we finally got the LEACH patch to run on Linux only.

Our second main challenge was to understanding how to read and code in TCL. The LEACH patch gave us a very good starting point but it came at the cost of having code that was more complicated than we could understand at the time. As such, we spent more time trying to understand how this code was working than we had anticipated. Also the version of LEACH in the patch did not have all the variables and behaviour we wanted, ex: it would remove energy from a node based on a fixed value without taking into account the distance of transmission. So by going through all the code we finally got to understand it all, fix any errors that existed and put our own formulas and variables in place of the existing ones.

Some other challenges we faced was that the LEACH code would sometimes just stop in mid-simulation with no explanation, forcing us to rerun the simulation and with bigger simulation taking up to an hour, it was time consuming. Another challenge is that our EECS code removes half the energy of cluster head's that are elected in the first round, which was mysterious because we used some of the same code in LEACH and EEUC and this did not occur. These are the kind of inconsistencies that were frustrating to deal with. If we had more time we would have changed our decision to use the LEACH patch and would have begun with our own code.

VII. Simulation Results and Comparisons

In this section we will describe the results from our simulation, analyze them, compare them against each other and against the results presented the in research papers covered. It is important to note that we used the same threshold for every topology. As discussed in our Implementation section, we keep threshold the same to achieve comparable, but not optimal, results for every topology.

A. Simulation Results

The following 3 graphs are the results of our simulations. We ran each algorithm on 15 different topologies, 5 for each 75, 100 and 150 nodes. Each line for each graph is the average of the 5 runs on each topology for that #node graph. Eg: the line for LEACH in Figure 7 is the average of runs on all 5 different topologies with 75 nodes.

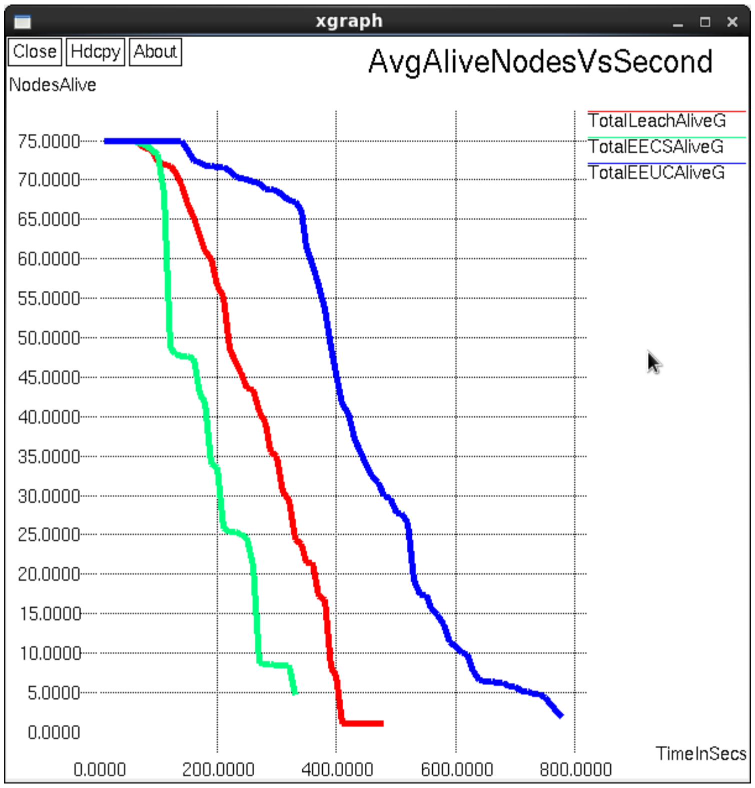

Figure 7. 75 node topology average results

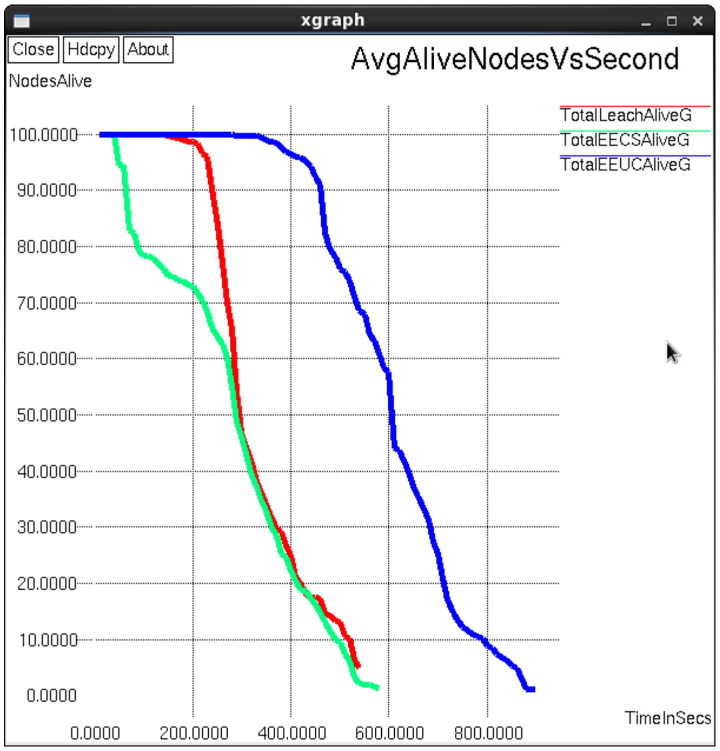

Figure 8. 100 node topology average results

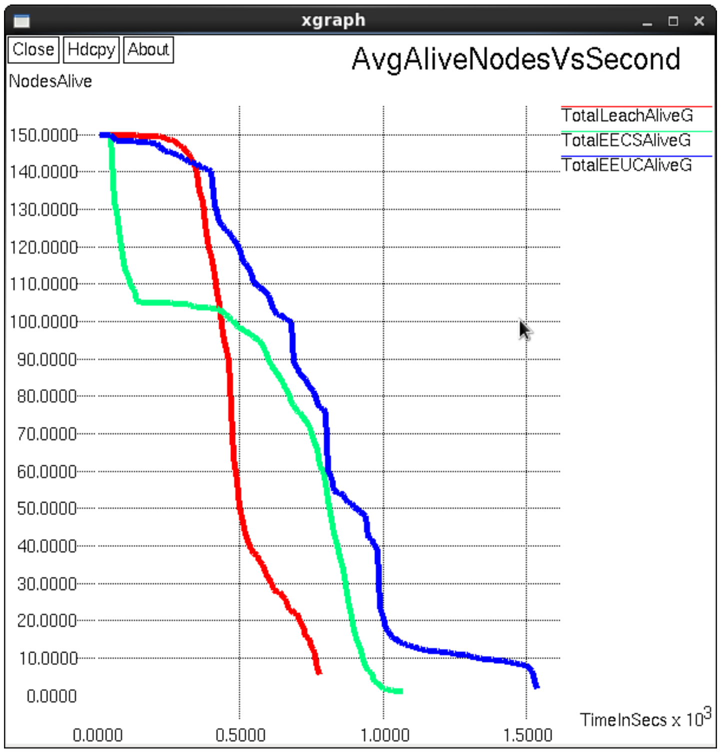

Figure 9. 150 node topology average results

Algorithm	# nodes	Average Time of first Node Death	Average Overall system lifetime
LEACH	75	80	480
EECS	75	110	290
EEUC	75	150	780
LEACH	100	180	540
EECS	100	50	580
EEUC	100	350	860
LEACH	150	240	780
EECS	150	50	1000
EEUC	150	190	1500

Figure 10. Simulation Results from Figures 7, 8, 9 (in seconds)

Looking at Figure 7, we can see that LEACH has the time of first node death, and second overall system lifetime. EECS ranks second in first node death and last in overall system lifetime. EEUC outperforms both other algorithms in both aspects.

These simulations gave the expected results from looking at the papers. EECS has expected 35% increase of time of first node death over LEACH according to [4]. If we use the values from Figure 10, in Figure 7 EECS has a 37% increase over time of first node death over LEACH. So these values are comparable. According to [6] "EEUC clearly improves the network lifetime (both the time until the first node dies and the time until the last node dies)", while they do not gives specific values, this is also occurring in our simulation. EEUC has an 87% increase over time of first node death and a 62% increase in total system lifetime over LEACH. Finally EEUC has a 36% increase over EECS in time of first node death and 168% increase in overall system lifetime.

One of the reasons EEUC does much better than the single-hop algorithms for both time of first node death and overall system lifetime is that it uses small hops to get the base station, rather than one long transmission. We have an energy distance loss function (described in the Implementation section) which takes into account the distance that the transmission must go, and it exponentially increases distance increases. Therefore small hops take less energy than long hops.

Looking at Figure 8, EECS has a very quick time of first node death, and the others perform as in the previous simulation. The reason for this is that there seems to be a bug in the EECS code that removes around half the energy of all the cluster heads elected in round one. That is why we see a quick dropoff and then it stabilizes around time 150 and then behaves as expected. The fact that it stabilizes after a certain time shows that the algorithm runs correctly but only after the first few rounds where the bug is occurring. As far as EEUC goes, it improves time of first node death by 94% over LEACH and a 59% increase in overall system lifetime which is very similar results in what we saw in Figure 7. So we can conclude that these two algorithms are working as expected. It is important to note that there is some randomness in these algorithms which is why the values are not consistently the same, so we give a margin of flexibility of about +-10%.

Looking at Figure 9, the EECS bug is even more obvious where there are many nodes dying very quickly and then the algorithm levels off and stabilizes around time 150. The other important note is that EEUC does not outperform LEACH in time of first node death in this high density scenario as we have seen in the previous two Figures. Here are our thoughts as to what caused this. First, because of the node density there will be many more collisions during the candidate cluster advertisement message, so nodes could be wasting energy there. In addition our threshold is very high, which will result in lots of nodes becoming candidate cluster heads, which will lead to more candidate cluster head messages colliding. Secondly, because of the density again, the data could be taking a large amount of small hops and these hops result in a decrease in performance because of the nature of our energy distance loss formula (we subtract a fixed base amount along with and addition to an exponential amount that increases with distance). So if there are many small hops, it will trigger the fixed base amount many times and may not actually save energy as compared to a long hop. So once the area becomes less dense by nodes dying, the algorithm then goes back to being 'better' and in the end has a better overall system lifetime. So what we see here is similar in what we see in EECS, at the beginning it does not work well in high density areas, but once the area becomes less dense the algorithm stabilizes and overall it increases overall system lifetime. In Figure 9, EEUC has a better overall system lifetime by 92% over LEACH, which is more than we had previously seen but makes sense because of the number of nodes in the network.

B. Comparisons to research papers

Let us look at the simulation results presented in the LEACH research paper [1] and in Figure 11 and compare them against our own results in Figure 12.

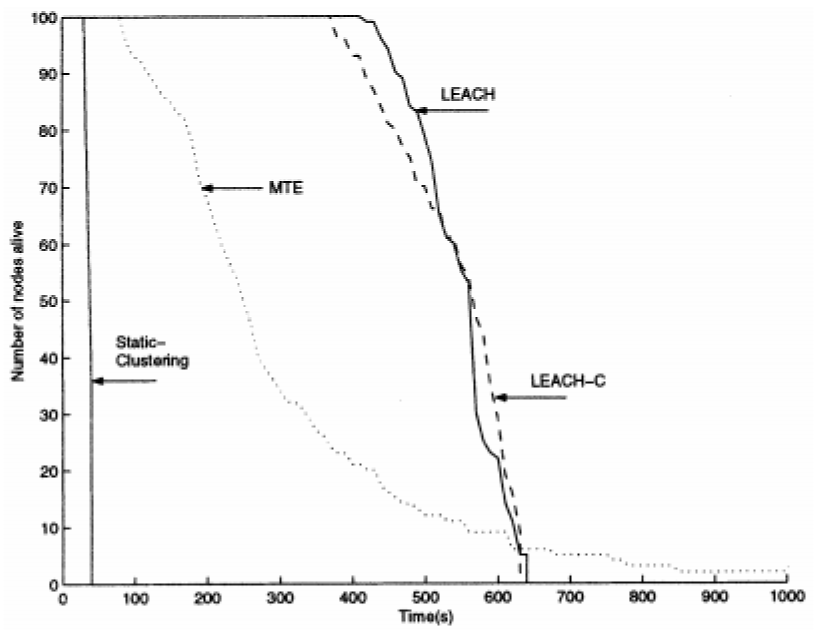

Figure 11. Simulation Results from LEACH paper [1]

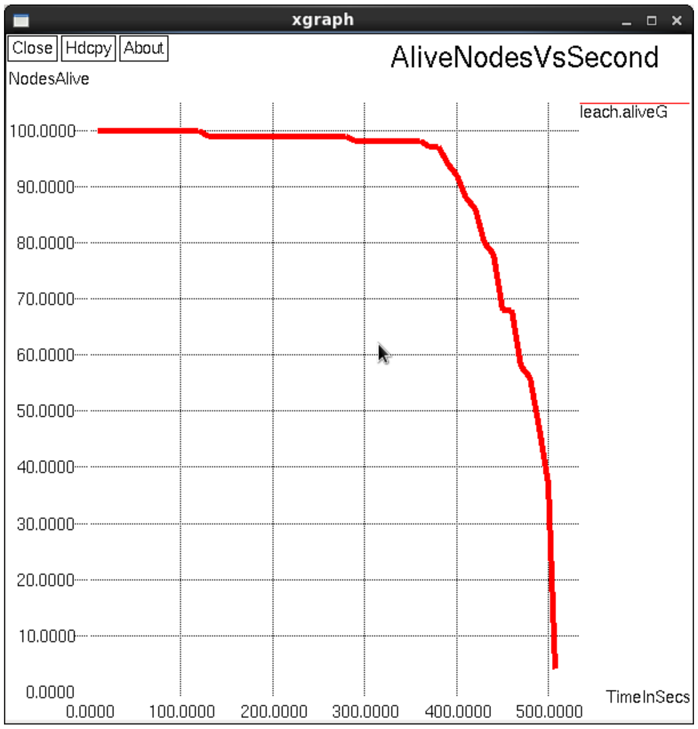

Figure 12. 100 nodes LEACH with optimal threshold

If we compare the results, with 100 node topology our simulation gave 120 seconds for time of first node death and 520 seconds of overall system lifetime. If we approximate from the Figure 11, [1] has the results of ~400 seconds for first node death ~660 for overall system lifetime. The gap between time of first node death is what jumps out (120 to 400), but if we take a look at Figure 12, we can see that we are losing very few nodes until ~380 seconds, and then there is a big drop-off. So in fact the graphs do look very similar.

We stated above that we use the LEACH patch that is based off of this paper and that we used this LEACH patch as a basis for our code which is why we get similar results. However we modified some the energy loss formula which is why we get different results. We introduced an exponential energy distance loss formula which the patch did not. The patch just deducted a fixed amount of energy when sending a transmission. Therefore it did not matter how far or close the transmission was, it would always remove a fixed amount of energy. In our simulation we have the exponential distance loss function so in fact we are removing more energy when the transmission is far. This is one of the reasons we have different results.
Also it is important to note that the difference between Figure 12 and our results in Figure 8 is that we used an optimal threshold in Figure 12 in order to compare to the results presented in the LEACH paper.

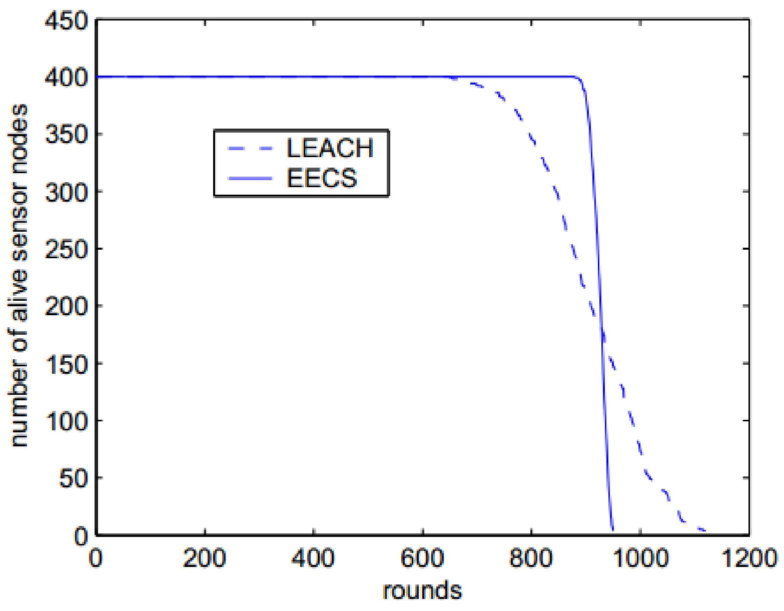

Figure 13. Simulation Results from EECS paper [5]

Figure 13 presents the results from the EECS research paper [5]. In their simulations they use 400 nodes as a test case with .5J of energy. They claim to have a 35% increase of time of first node death than in LEACH [5]. We will compare these results to the results presented in Figure 7, because those are the most comparable when our EECS code does not seem to be affected by the bug. In our results from Figure 7, we had an increase of 37% which matches the 35% increase presented in the paper. In addition LEACH outperforms EECS in overall system lifetime by about 33% in Figure 13 which also occurs in Figure 7. If we look at Figure 7, LEACH dies at 480 but is just on the border of dying at 400. Let us use the latter value because there is some randomness in the algorithms and in another 5 runs it could have died at 400. So if we use 400 as our time of death we get 37% increase in overall system lifetime which is almost what is presented in Figure 13.

The reason EECS outperforms LEACH in time of first node death is because of the load balancing done among cluster heads. This is due to the fact that normal nodes calculate the best cluster based on the weighted cost function mentioned above.

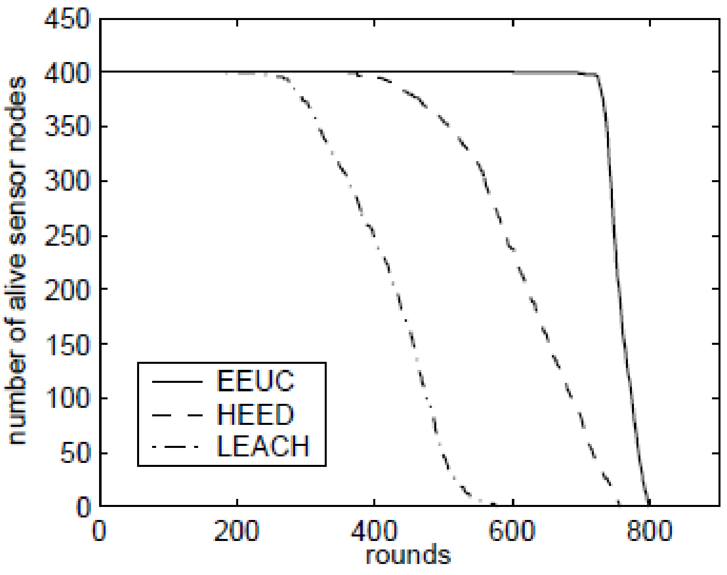

Figure 14. Simulation Results from EEUC paper [6]

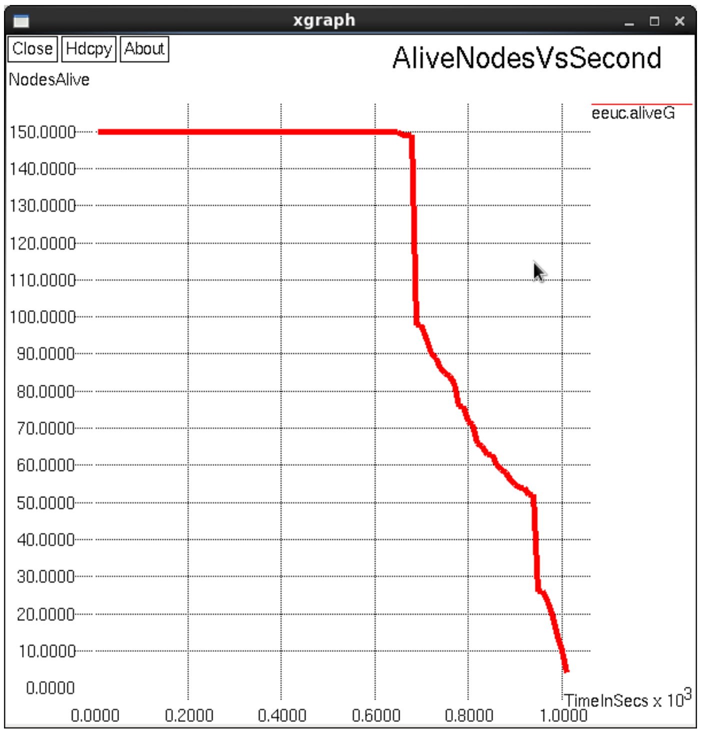

Figure 15. 150 nodes EEUC with optimal threshold

Figure 14 shows the results from [6] using a 200 x 200 area with 400nodes having .5J of energy. Figure 15 represents our comparison results for our EEUC algorithm chosen with an optimal threshold for 150 nodes. We chose 150 nodes because there were errors running 400 nodes including “MAX meta data hit error” which basically prevents us from running a simulation at such a high density. If we look at time of first node death they occur around the time same. The major difference is the gap between time of first node death and overall system lifetime. In Figure 14 the gap is very small, around 100 seconds, which [6] says proves that EEUC has solved the hot spot problem. In Figure 15, the gap is quite large at around 300 seconds. This may be because of the different size of areas and since the formula used to calculate competition range takes into account the maximum and minimum distances in the network, the area may have an effect.

If we compare Figure 15 to Figure 9, we will see that it is much more stable in 15 because of the optimal threshold chosen. This comparison in fact shows the impact of threshold on the algorithm. Because in 15 we chose an optimal threshold, it behaves more closely to what we expected than in Figure 9. In Figure 9 the time of first node death is very quick which should not happen in EEUC, but in Figure 15 the time of first node death is much later.

C. Data received comparisons

In reading the research papers we realized that LEACH is the only paper that compares how much data is actually received by the base station. LEACH says it deliver more data than static clustering and MTE [1]. However the EECS and EEUC papers say that they are more energy efficient than LEACH but never speak about how much data is actually received by the base station. In this section we will show our results for how much data is received by the base station by each algorithm on average for each set of topologies and analyze the results.

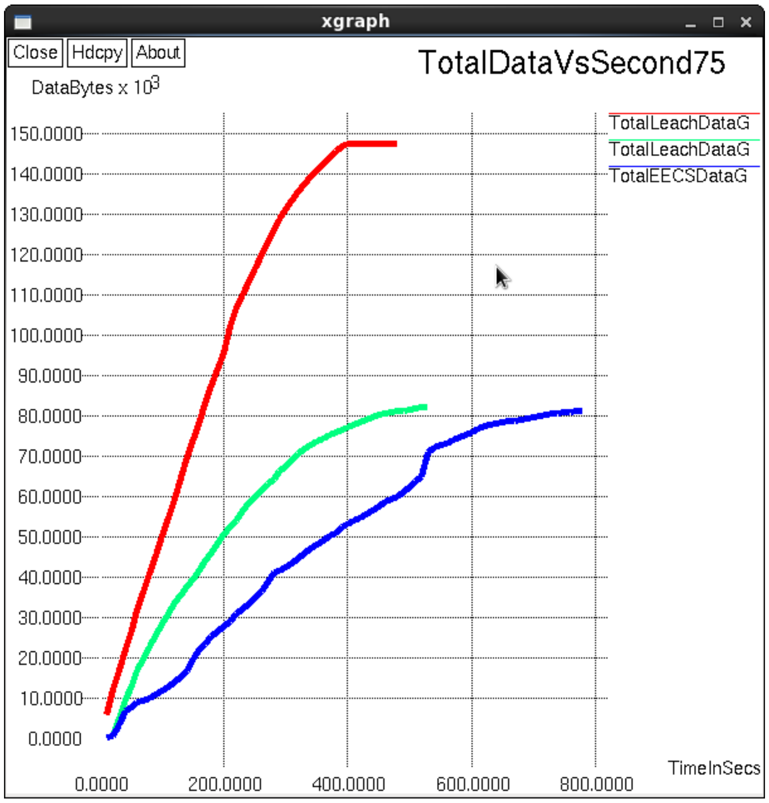

Figure 16. 75 nodes Total Data received by BS

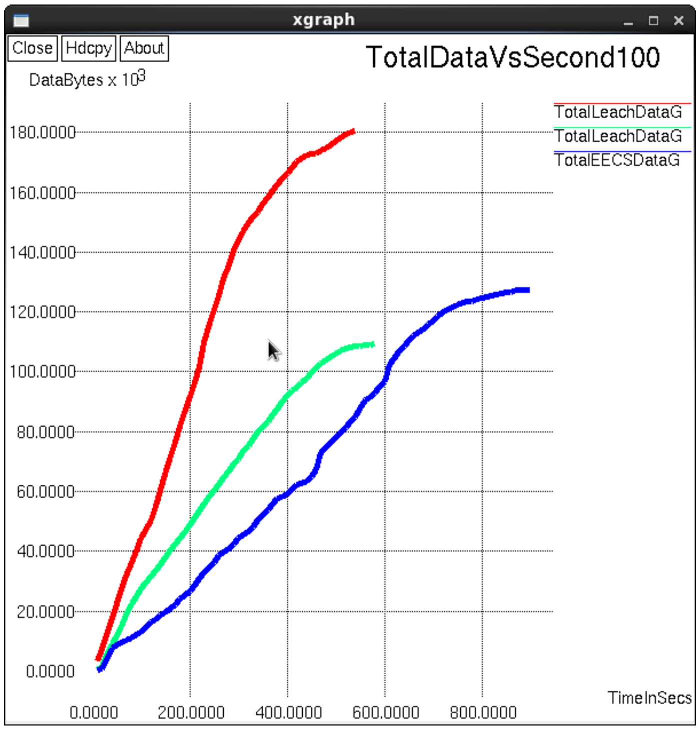

Figure 17. 100 nodes Total Data received by BS

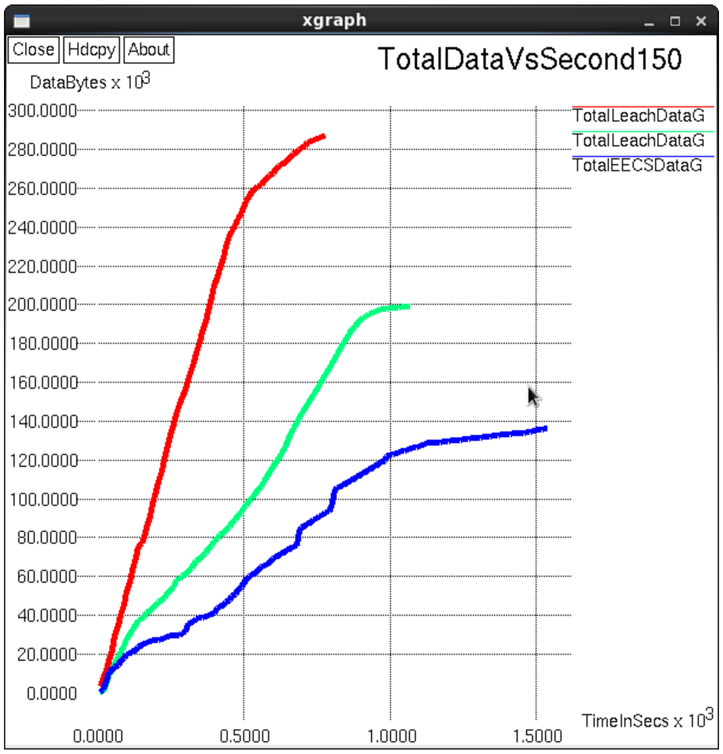

Figure 18. 150 nodes Total Data received by BS

In Figures 16, 17 and 18 we basically see the same trend, that LEACH sends much more data to the BS than EECS and EEUC. In Figure 16 LEACH sends ~130,000 bytes as compared to ~80,000 bytes in EECS and EEUC, an increase of 62%. LEACH sends even more than EEUC even though EEUC is alive much longer than LEACH. Here are some possible explanations as to why this occurs.

The first explanation is that in EECS and EEUC there are many more control messages being passed around because of the candidate cluster head advertisement messages. So if nodes are out of synchronization and some are sending candidate cluster head messages at the same time that other nodes are sending cluster head advertisement messages, it could result in nodes not joining a cluster at all. For example let use have Node A, Node CCH and Node CH where A is a normal node, CCH is a candidate cluster head and CH is a cluster head. Assume A only has CH in its range and is waiting to hear an advertisement from CH. Suppose that CH's advertisement message collides with CCH's advertisement message. Then A will never get CH's advertisement message and thus not join any cluster. If we assume the algorithm is EECS, then that node will try to send data directly to the base station which may or may not reach the BS because of collisions. If many nodes are trying to send directly to the base station they may collide with each other or with data being sent from cluster heads. If we assume the algorithm is EEUC, A will only try to send data directly to the base station is the BS is in its transmission range, which again may or may not reach the BS, or if it is not in transmission range it will not send its data at all. This leads to our second explanation.

In single-hop algorithms we assume all nodes can reach the BS, so if a node for one reason or another does not join a cluster it may try to transmit to the BS. Like discussed previously it may or may not reach the BS. In EEUC, which is a multi-hop algorithm, we do not assume that all nodes can reach the BS so if a node does not have a cluster and is outside transmission range of the BS it will not even try to send data, which is why we have less received data by the BS in EEUC.

So our results are interesting because it seems to show that while EECS and EEUC seem to have better energy efficiency, they send less data. This is an important result because it changes what algorithm you may want to choose when implementing your wireless sensor network. While EECS and EEUC do offer better energy efficiency, according to our results they do it at the price of lower delivery rates.

VIII. Conclusion

In this paper we compared four algorithms, two single hop algorithms, LEACH and EECS, and two multi hop algorithms, EEUC and EEDUC. We compared them to see which was a more energy efficient clustering solution for wireless sensor networks. We then detailed each algorithm and compared their similarities and differences. It was clear that multi hop algorithms had an advantage over single hop algorithms because a couple of small hops was more energy efficient than one long hop. We then proceeding to implement three of these algorithms in a network simulator and compared them against each other in different density networks. Our results confirmed the results of the research papers covered, and we proved that multi hop algorithms tend to use energy more efficiently than single hop algorithms but seem to lower data delivery rates.

IX. Contributions

Name	**Contribution**
Andrew Kefalas	Report, presentation slides, algorithm implementation in code and general coding, revision and editing of report
Mahesh Bakshi	Report, Installing and general debugging of ns-2, coding and understanding underlying structure of code, general revising and editing of report

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Microsensor Networks,” *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.

[2] Y. Li, N. Yu, W. Zhang, W. Zhao, X. You, M.Daneshmand, “Enhancing the Performance of LEACH Protocol in Wireless Sensor Networks”, *In Proceeding of “M2MCN-2011”,* pp. 223-228, 2011.

[3] H. Chen and S. Megerian, “Cluster Sizing and Head Selection for Efficient Data Aggregation and Routing in Sensor Networks,” *in Proc. IEEE WCNC 2006*, pp. 2318–2323, 2006.

[4] M. Ye, C. Li, G. Chen and J. Wu, “An Energy Efficient Clustering Scheme in Wireless Sensor Networks”,*in Proc. of the 2nd IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS),Washington, DC*, Nov. 2005.

[5] M. Ye, C. F. Li, G. H. Chen, and J. Wu, “EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks”, in *Proceedings of IEEE Int’l Performance Computing and Communications Conference (IPCCC)*, pp. 535-540, 2005.

[6] C. F. Li, M. Ye, G. Chen and J. Wu, “An energy-efficient unequal clustering mechanism for wireless sensor networks”,*in Proc. of the 2nd IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), Washington, DC*, Nov. 2005.

[7] D. Wei ,Y. Jin and S. Vural,K.Moessner, R. Tafazolli, ”An Energy-Efficient Clustering Solution for Wireless Sensor Networks”, *IEEE Transactions on Wireless Communications*, vol. 10, no. 11, pp.3973-3983, 2011.

[8] S. Lee, J. Lee, H. Sin, S. Yoo, S. Lee, J. Lee, Y. Lee, and S. Kim, “An energy-efficient distributed unequal clustering protocol for wireless sensor networks,” *in PWASET*, pp. 1274–1278, December 2008.

[9] *The Network Simulator,*The University of Southern California, 2011. Available fromhttp://www.isi.edu/nsnam/ns/

[10] *LEACH Patch,* Microsystems Technology Laboratories: Massachusetts Institute of Technology, 2011. Available from http://www-mtl.mit.edu/research/icsystems/uamps/cadtools

Appendix A

Here we give steps on how to run our code and how to run the scripts to generate the graphs. We installed ns-2.24 on Scientific Linux 5 OS, so some commands are specific to this OS.

Step 1. Install ns-2.24
Step 2. Merge ns-2.24_Clustering/ns-2.24 directory with ../ns-allinone-2.34/ns-2.24 directory (including all sub directories)
Step 3. Go to terminal ../ns-allinone-2.34/ns-2.24 folder
Step 4. Compile our code changes - re-make ns2 file. (make clean and then make)
Step 5. To run leach, you need to run ./test in terminal folder location ../ns-allinone-2.34/ns-2.24 folder.
Step 6. To run leach, you need to run ./testeecs in terminal folder location ../ns-allinone-2.34/ns-2.24 folder.
Step 7. To run leach, you need to run ./testeeuc in terminal folder location ../ns-allinone-2.34/ns-2.24 folder.

Location of files.Consider EECS:
input parameters will be in file: ../ns-allinone-2.34/ns-2.34/eecs_test
input topology will be in file: ../ns-allinone-2.34/ns-2.34/mit/uAMPS/sims/100nodes.txt
main code of leach will be in file: ../ns-allinone-2.34/ns-2.34/mit/uAMPS/ns-eecs.tcl
out files after running "./testeecs" will be in files: ../ns-allinone-2.34/ns-2.34/mit/leach_sims/eecs.out, ../ns-allinone-2.34/ns-2.34/mit/leach_sims/eecs.energyand ../ns-allinone-2.34/ns-2.34/mit/leach_sims/eecs.data

You can generate multiple topologies using file: ../ns-allinone-2.34/ns-2.34/mit/uAMPS/sims/genscen. (get to folder /root/ns-allinone-2.34/ns-2.34/mit/uAMPS/sims in terminal and run command 'genscen')

To generate graphs: In terminal run below script -

```
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' leach.energy>temp
sort -t" " -k1 -nu temp>leach.energyG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' leach.data>temp
sort -t" " -k1 -nu temp>leach.dataG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' leach.alive>temp
sort -t" " -k1 -nu temp>leach.aliveG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eecs.energy>temp
sort -t" " -k1 -nu temp>eecs.energyG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eecs.data>temp
sort -t" " -k1 -nu temp>eecs.dataG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eecs.alive>temp
sort -t" " -k1 -nu temp>eecs.aliveG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eeuc.energy>temp
sort -t" " -k1 -nu temp>eeuc.energyG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eeuc.data>temp
sort -t" " -k1 -nu temp>eeuc.dataG
awk '{a[$1]+=$3}END{for (i in a) print i,a[i]}' eeuc.alive>temp
sort -t" " -k1 -nu temp>eeuc.aliveG
```

Manually 5 different topologiesleach.alive to TotalLeach and use below scripts to get average alive nodes graph:

```
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalEECS>temp
sort -t" " -k1 -nu temp>TotalEECSAliveG
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalLeach>temp
sort -t" " -k1 -nu temp>TotalLeachAliveG
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalEEUC>temp
sort -t" " -k1 -nu temp>TotalEEUCAliveG
```

```
xgraph -x TimeInSecs -y NodesAlive -lw 5 -bg white -fg black -t
AvgAliveNodesVsSecondTotalLeachAliveGTotalEECSAliveGTotalEEUCAliveG
```

Manually 5 different topologoesleach.alive to TotalLeach and use below scripts to get total data reached at base station graph:

```
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalEECSData>temp
sort -t" " -k1 -nu temp>TotalEECSDataG
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalLeachData>temp
sort -t" " -k1 -nu temp>TotalLeachDataG
awk '{a[$1]+=$2}END{for (i in a) print i,a[i]/5}' TotalEEUCData>temp
sort -t" " -k1 -nu temp>TotalEEUCDataG
```

```
xgraph -x TimeInSecs -y DataBytes -lw 5 -bg white -fg black -t
AvgDataVsSecondTotalLeachDataGTotalEECSDataGTotalEEUCDataG
```