

Maximizing Video Sensor Network Lifetime through Genetic Clustering

Min Qin and Roger Zimmermann
Department of Computer Science
University of Southern California, Los Angeles, CA 90089

Abstract

In this paper we propose a novel communication protocol for studying the upper bounds on the lifetime of a video sensor network. Such networks are typically small due to their cost. Also, video sensors may have different transmission rates and data aggregation is difficult. Sensors are organized into clusters and a linear programming model is introduced for calculating a cluster head rotation schedule. Unlike most other clustering algorithms, our algorithm maximizes the network lifetime rather than minimizing the energy dissipation of sensors. Compared with previous models, our approach allows non-uniform transmission rates. In addition, the new protocol can maximize the K -of- N network lifetime, which has not been studied by most previous approaches so far. We use a genetic algorithm to compute optimal cluster formations. Simulation results show that our model can extend the lifetime of a sensor network up to five times over that of existing approaches.

1. Introduction

Video sensor networks [13, 18] have recently become feasible as an integral part of innovative new applications. They are generally small in scale and are carefully deployed. However, limited battery power hinders the development of video sensors. In the past several years, storage space and computing power of sensors have increased dramatically. To save energy dissipation, many routing protocols are devised for data sensor networks. For example, directed diffusion [10] is designed for querying data and keeping data within the sensor network. Because video sensors often do not have the computing power to analyze image/video data, data gathered from all sensors need to be extracted to an external node, called *base station* (sink). Generally, the base station is far away from all sensors. Data aggregation [7] is often used to save power in traditional sensor networks. However, it is hard to aggregate video data.

Most previous work [5, 8, 11, 14, 16] defines the lifetime of a sensor network as the duration from the deployment of the network to the time when the first sensor runs out of energy. We call this the N -of- N network lifetime. All sensors are required to work at a time. The definition of N -of- N network lifetime is very harsh for many applications. To be fault tolerant, some sensors may act as backups. In this paper we extend the N -of- N problem to the K -of- N lifetime problem, in which only K sensors are required to work at a time. To our knowledge, there have been few studies on maximizing the K -of- N network lifetime so far.

Algorithms for calculating a suitable base station location to prolong network lifetime are introduced in [17]. However, base station locations are often determined before all sensors are placed. Also, mobile base stations are not very common under current technology. For fixed base station locations, the upper bounds on N -of- N sensor network lifetime have been studied in [4, 6, 11]. In [11], the authors introduce a MLDR (Maximum Lifetime Data Routing) algorithm that models this problem into an integer problem with linear constraints. However, the MLDR model assumes that all sensors have the same transmission rate, which is not suitable for video sensor networks. Using MLDR will force sensors to split a single packet into several pieces and transmit them over different routes. In practice, it is hard to coordinate all sensors since packets from the same sensor are routed and fragmented differently. Therefore new routing strategies are needed.

Clustering [1, 2, 3, 5, 6, 8, 14, 15, 16] is a very useful approach to reduce energy dissipation in sensor networks. Data aggregation [7] is often coupled with clustering to further extend sensor lifetime. In Fig.1, for example, sensors are divided into 4 clusters. Nodes marked in black serve as the heads (local base stations) for each cluster. Data collected from sensors are sent to the cluster head first, and then forwarded to the base station. Compared to the base station, the cluster head is closer to sensors in the same cluster. Sensors can save transmission energy by sending data to the cluster heads instead of the base station. The cluster heads may aggregate data from different sensors to minimize the amount of data to be sent to the base station.

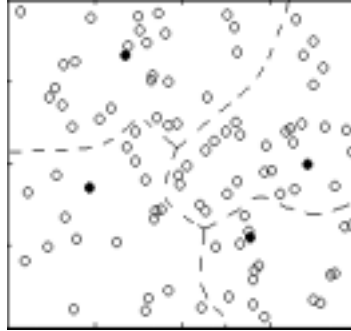


Figure 1. A sensor network with 4 clusters

Most of the existing clustering algorithms do not take the data transmission rate into consideration. In LEACH [8], each sensor has the same probability of becoming a cluster head per round. The formation of clusters may change over time. HEED [16] extends LEACH so that the probability of becoming a cluster head is dependent on residual sensor energy and degree cost. Both HEED and LEACH require re-clustering after a period of time, which may cause extra energy consumption. Similarly, WCA [5] and DCA [3] use weights associated with sensors to elect cluster heads. PEGASIS [14] uses a fixed cluster formation. Sensors within a cluster are organized into a chain. The cluster head rotates in a round robin fashion along the chain. All these protocols may work well when all sensors have the same transmission rate. For sensors with non-uniform transmission rates, those with a high transmission rate and low initial energy can easily become the bottleneck in calculating the N -of- N network lifetime.

In this paper, we introduce the **maximum lifetime sensor clustering (MLSC)** protocol for maximizing the lifetime of video sensor networks when clustering is used. We use a fixed cluster formation instead of re-clustering over time to reduce energy dissipation. This makes it possible to pre-determine the routing strategy. We do not suggest that localized clustering algorithms such as LEACH or HEED are not useful. Our MLSC protocol is preferable in cases where (a) the scale of the network is not very large (Communication range of sensors can cover most of the area), and (b) the routing strategy can be delivered to all sensors after they are deployed. Video sensor networks are a very good match for our assumptions. They are often small in scale due to the cost. Locations of video sensors are carefully chosen to reduce overlapping viewing angles. At the beginning stage, all sensors send their information to the base station or a designated special node. Upon receiving this information, the base station will calculate and send the clustering information back to all sensors. After that, the cluster formation is fixed.

Compared to previous works, our algorithm assumes that sensors may have non-uniform transmission rates and initial energy. This is a very important assumption for video sensor networks. Video sensors may produce with different frame rates and resolutions. Our routing strategy incorporates a novel genetic algorithm for clustering when sensor information is available. It is robust to temporary communication failures. For each cluster, the cluster head rotation schedule is provided through a linear program. Another contribution of our MLSC protocol is that it can solve the K -of- N lifetime problem, which has not been widely studied so far.

The rest of the paper is organized as follow. In Section 2, we introduce the communication model for the sensor network and the parameters we are using. The MLDR model is modified to calculate the upper bound of the network lifetime. A linear programming model for calculating the cluster head rotation

strategy is introduced in Section 3. Section 4 presents a genetic algorithm based clustering approach. In Section 5, experimental results are reported. Finally, we summarize the contributions and provide suggestions for further research.

2. Modeling for sensor communications

2.1 Energy consumption model for sensors

Table 1 lists all the terms and definitions used in this study. We assume that the transmission rate and location information of all video sensors are available to us. There exists only one base station (sink) in our analysis. If two sensors cannot communicate directly, we set the distance between them as infinite. For simplicity, our experiments assume that all video sensors can communicate with each other directly, including the base station.

To model energy dissipation, we use a d^2 energy loss estimation for channel transmission. In order to transmit data from the i th sensor v_i to the base station directly, the power consumption rate is modeled as

$$p_t(r_i, d_i) = r_i(\alpha_1 + \alpha_2 d_i^2) \quad (1)$$

where r_i is the transmission rate of sensor v_i . To receive a message of length r bits, the energy required is

$$p_r(r) = r\alpha_\gamma \quad (2)$$

We use the radio model described in [8], in which $\alpha_1 = \alpha_\gamma = 50$ nJ/bit, and $\alpha_2 = 100$ pJ/bit/m². The N -of- N network lifetime is defined as the duration from the deployment of the network to the time when the first sensor runs out of energy. If all sensors communicate directly with the base station, we have

Table 1. List of terms used and their definitions.

Term	Definition	Units
α_1	Energy dissipated to run the radio transmitter	J/bit
α_2	Energy dissipated at the transmit amplifier	J/bit/m ²
α_γ	Energy dissipated to run the receiver circuitry	J/bit
K	Number of sensors required to work at the same time (including the cluster head)	
N	Total number of nodes	
v_i	The i th sensor ($1 \leq i \leq N$)	
d_i	Distance from sensor v_i to the base station	m
d_{ij}	Distance from sensor v_i to sensor v_j	m
r_i	Transmission rate of sensor v_i	bit/round
e_i	Initial energy of sensor v_i	J
S	An $n \times n$ square in which all sensors $v_1 \dots v_n$ are located	m ²
$Bdist$	The distance from the center of S to the base station	m
T_L	The network lifetime	Round
L_i	Lifetime of the i th sensor v_i	Round
M	Number of clusters in the network	

$$T_L = \min\{L_i\} = \min\left\{\frac{e_i}{r_i(\alpha_1 + \alpha_2 d_i^2)}\right\} \quad (3)$$

Thus sensors with low initial energy and a high transmission rate are the bottleneck in determining the N -of- N network lifetime.

2.2. Upper bound on N -of- N network lifetime

Kalpakis et al [11] introduced a MLDR algorithm for calculating the upper bound of the N -of- N network lifetime. The MLDR algorithm generalizes sensor energy constraints into a linear programming

model. However, the original MLDR model assumes uniform transmission rates for all sensors. Here we extend their model to networks with non-uniform transmission rates.

Let x_{ij} denote the total number of bits sensor v_i sends to sensor v_j . For simplicity, the base station is regarded as sensor v_{N+1} . The energy constraint on sensor v_i ($1 \leq i \leq N$) can be represented by the following formula:

$$\sum_{j=1}^{N+1} x_{ij}(\alpha_1 + \alpha_2 d_{ij}^2) + \sum_{j=1}^N \alpha_r x_{ji} \leq e_i \quad (4)$$

$$x_{ij} \geq 0, \quad i=1 \dots N, j=1 \dots N+1 \quad (5)$$

Here we assume that there is no data aggregation in the video sensor network. In Equation (4), $x_{ij}(\alpha_1 + \alpha_2 d_{ij}^2)$ represents the total energy spent on forwarding data to sensor v_j . Similarly, $\sum_{j=1}^N \alpha_r x_{ji}$ is the total energy consumption for receiving data from other sensors. For sensor v_i ($1 \leq i \leq N$), its incoming data plus the data it generated itself should be equal to the outgoing number of bits. Thus we have

$$r_i T_L + \sum_{j=1}^N x_{ji} = \sum_{j=1}^{N+1} x_{ij} \quad (6)$$

Equations (4), (5) and (6) constitute a linear programming model and our objective is to maximize T_L .

There are several problems in applying this linear programming model to sensor networks. The result for the linear program might not be an integer solution. Even if all x_{ij} happens to be integers, the sensors have to fragment all incoming packets and send them along different routes. This makes packet tracing and routing control impractical. The base station needs to assemble data fragments from all sensors, which is very complicated since different sensors do fragmentation arbitrarily. The protocol is also vulnerable to communication failures. Any temporary failure can disrupt the routing strategy.

Since there are N^2+1 variables (x_{ij} and T_L) in the linear program, the computational complexity for solving the linear problem is $O(N^7 L)$ by using the interior point algorithm [12, 19], where L is the size of the input matrices. From Equations (4),(5) and (6), we have $L=O(N^3)$. So the total complexity of this model is bounded by $O(N^{10})$. In practice, the interior point algorithm converges very fast. When $N=100$, for example, it only takes several seconds to solve this linear program using MATLAB.

3. Clustering sensors inside the network

Because cluster heads are closer to all video sensors than the base station, sensors save their energy by reducing transmission power when sending to their cluster heads. Clustering also makes the routing control easier. Sensors only need to know their current cluster head and deliver all data to it.

3.1. Locating cluster head for fixed clusters

First we solve the case when there is only one cluster in the sensor network. In this case, $v_1 \dots v_N$ all participate in the same cluster. If we choose sensor v_k as the cluster head, the lifetime of all sensors becomes:

$$L'_i = \begin{cases} \frac{e_i}{r_i(\alpha_1 + \alpha_2 d_{ki}^2)} & , i \neq k \\ \frac{e_k}{(\alpha_1 + \alpha_2 d_k^2)R + \left(\sum_{i=1}^{k-1} r_i + \sum_{i=k+1}^N r_i \right) \alpha_r} & , i = k \end{cases} \quad (7)$$

where $\mathbf{R} = \sum_{i=1}^N r_i$. In Equation (7), $(\alpha_1 + \alpha_2 d_k^2) \mathbf{R}$ represents the energy consumption rate for forwarding data to the base station from v_k . Similarly $(\sum_{i=1}^{k-1} r_i + \sum_{i=k+1}^N r_i) \alpha_r$ is the energy consumption rate for receiving data from sensors. Here no data aggregation is available. However, we can easily extend Equation (7) to incorporate data aggregation if sensors are able to analyze video data.

Because the cluster head consumes more energy than other nodes, adhering to one cluster head may drain its energy quickly. We need to rotate the cluster head after a period of time. Let t_k denote the total number of rounds that sensor v_k serves as the cluster head. According to Equation (7), for each sensor v_k , we have

$$t_k ((\alpha_1 + \alpha_2 d_k^2) \mathbf{R} + (\mathbf{R} - r_k) \alpha_r) + r_k \sum_{i=1, i \neq k}^N t_i (\alpha_1 + \alpha_2 d_{ik}^2) \leq e_k \quad (8)$$

$$t_i \geq 0, \quad i = 1 \dots N \quad (9)$$

Our goal is to maximize $T_L = \sum_{k=1}^N t_k$. Again, Equation (8) and (9) constitute a linear program, which can be solved in polynomial time. The input size of the linear program is $O(N^2)$ and the computational complexity to solve it is $O(N^{5.5})$. This is much lower than that of the modified MLDR model. Thus clustering is very helpful for a large N .

3.2. Cluster head rotation strategy

The cluster head rotation strategy can be pre-determined by the linear program introduced in the above section. Initially, all sensors send their energy and transmission information to the base station or a special designated node. After solving the linear program, the base station sends the rotation strategy back to all the sensors.

The linear program returns a real number solution. In order to coordinate multiple sensors, we derive a near-optimal integer solution from it so that the cluster head is rotated after a number of rounds. Let (t_1, t_2, \dots, t_N) denote the optimal solution returned by the linear program. Initially, sensor $v_1 \dots v_N$ are assigned $t_1 \dots t_N$ time slots, respectively. After each round, the current cluster head consumes one time slot. We then select the sensor with the maximum time slots available as the next cluster head. Algorithm 1 shows this procedure.

Algorithm 1: Cluster head rotation strategy

Begin

1. Use the linear programming model to calculate the optimal solution $t_1 \dots t_N$.
2. While $\max\{t_i\} \geq 1$ do
3. Calculate i that maximizes t_i among $t_1 \dots t_N$
4. Select sensor v_i as the cluster head for the current round
5. After this round, let $t_i = t_i - 1$
6. End while

End begin

Since the actual lifetime of the sensor network is an integer, it is slightly lower than the theoretical value. Figure 2 compares the N -of- N network lifetime generated by the integer solution with that by the real number solution. The upper bound obtained by the modified MLDR model is also shown in Figure 2.

For each sensor, the simulation starts with a random energy from 1J to 5J and a random transmission rate between 2k to 20k bits per round. The clustering algorithm is covered in the next section. Our MLSC algorithm can achieve a network lifetime close to the upper bound. For example, when the cluster number is 5, the N -of- N network lifetime of the integer solution model is equal to 90% of the upper bound.

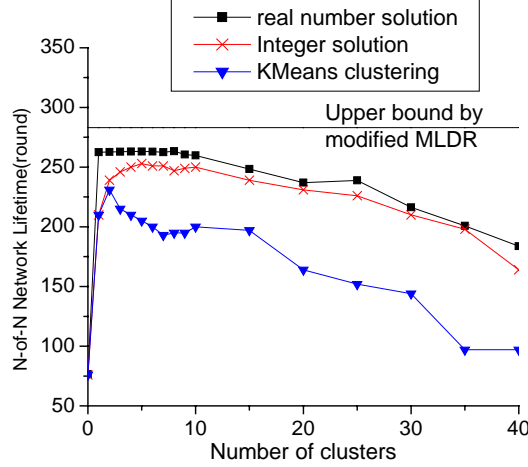


Figure 2. N -of- N network lifetime generated by our algorithm as a function of the number of clusters. $N=100$, $S=20*20$ m², $B_{dist}=100$ m.

Algorithm 1 is robust to communication failures. If the communication fails between a sensor and the cluster head, sensors can check the rotation schedule and switch to the next cluster head in the schedule. Because one cluster head is skipped for one round, the total lifetime of the sensor network is affected by at most 1 round in this procedure.

3.3. Solving the K -of- N lifetime

In a video sensor network, viewing angles of sensors may overlap. To save energy, many applications do not require all video sensors to work all the time. Some sensors are turned off and serve as backups. In this case, the lifetime is defined as the duration from the deployment to the time when less than K sensors can function properly. This is called the K -of- N lifetime problem. How to choose a suitable K is application dependant and therefore not the focus of this paper.

In order to extend the N -of- N lifetime problem to the K -of- N lifetime problem, let t_{ij} denote the rounds that sensor v_i serves as the cluster head while sensor v_j subscribes to it. From the definition, t_{ii} represents the total duration that sensor v_i serves as the cluster head. Since K sensors (including v_i) need to work at the same time when v_i is the cluster head, for each i ($1 \leq i \leq N$), we have

$$\sum_{j=1}^N t_{ij} = K t_{ii} \quad (10)$$

$$t_{ii} \geq t_{ij}, \quad 1 \leq j \leq N \quad (11)$$

We further observe the following energy consumption constraints for each sensor v_i

$$\begin{aligned} & (\alpha_1 + \alpha_2 d_i^2) \sum_{j=1}^N t_{ij} r_j + \alpha_r \sum_{j=1, j \neq i}^N t_{ij} r_j \\ & + r_i \sum_{j=1, j \neq i}^N t_{ji} (\alpha_1 + \alpha_2 d_{ij}^2) \leq e_i \end{aligned} \quad (12)$$

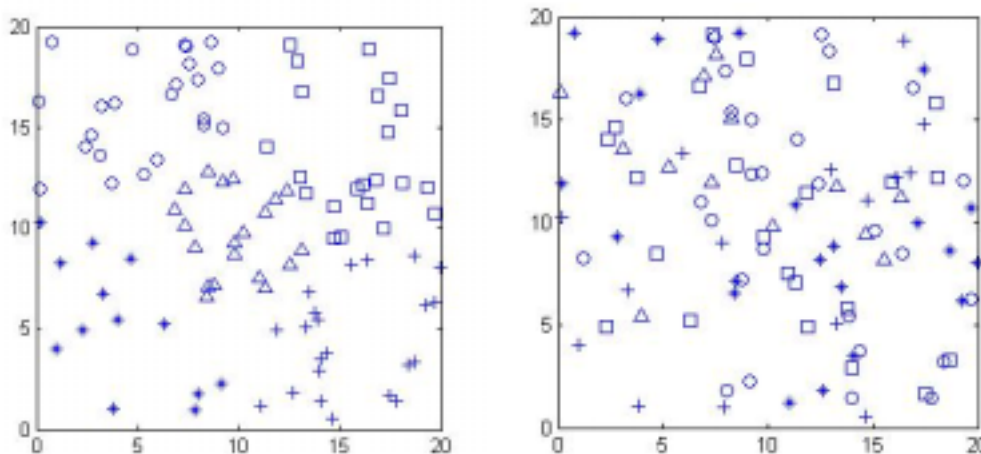
Again, our goal is to maximize $T_L = \sum_{i=1}^N t_{ii}$. The computational complexity for solving this linear program is $O(N^{11})$, higher than that of the N -of- N problem. The major challenge of the K -of- N lifetime problem with a single cluster is the space complexity. The input to the linear program solver is a sparse matrix with more than $N*N$ rows and $N*N$ columns. When N is large, the memory required to store this matrix increases dramatically. When $N=100$, for example, the memory required for storing this matrix will be 800MB if all variables use double precision. To lower the memory requirement, we use a divide and conquer approach to divide the problem into several sub-problems. First we use clustering algorithms to split a sensor network into several clusters. By applying the N -of- N or K -of- N linear programming model to each cluster, we can greatly lower the memory requirement and computational complexity. For the K -of- N lifetime problem, we proportionally divide K among all clusters according to their size. Details of our clustering algorithm are discussed in the next section.

4. Genetic clustering algorithm

We studied the single cluster problem in the above section. In order to save energy, more than one cluster is needed. In Figure 2, for example, using only one cluster may not yield the maximum integer solution. The space and computational complexity is also high for solving the K -of- N lifetime problem. Dividing the network into smaller clusters can reduce the size of the input to the linear problem solver.

To cluster sensors into several clusters, a traditional approach is to cluster adjacent sensors together. For example, the K-means algorithm can group nearby sensors. Figure 3(a) shows the cluster formation generated by the K-means algorithm using the same network configuration as Figure 2. There are a total of 5 clusters. Sensors in the same cluster are labeled with the same marker. For each cluster, we calculate the N -of- N lifetime using the linear program introduced by Equations (8) and (9). The minimum lifetime of any of these clusters is the actual N -of- N lifetime of the network.

The result of using the K-means algorithm is included in Figure 2. The N -of- N lifetime generated by K-means drops sharply when the number of clusters increases. This is due to the fact that sensors with low energy or a high transmission rate may be grouped into one cluster if they are geographically close to each other. Thus clustering nearby sensors together may not achieve a maximum network lifetime. We need to try different cluster formations in order to get the optimal network lifetime.



(a) K-means clustering, N -of- N lifetime=205 (b) Genetic clustering, N -of- N lifetime=253

Figure 3. Cluster formation generated by K-means and our genetic algorithm when cluster number is 5, sensors in the same clusters are labeled with the same marker.

The complexity of classifying N sensors into M clusters using complete enumeration is $O(M^N)$. It is computationally impractical to solve this problem when N is large. To pursue a solution, we use a novel genetic algorithm based approach. First, Q different random cluster formations are generated to constitute the initial solution set. Then, through a series of crossover and mutation functions, the offspring of these initial solutions progresses towards the optimal solution. In each iteration, the 2 solutions with the longest lifetime from the solution set are combined to form 2 new solutions using crossover. These two new solutions replace the worst 2 in the solution set.

For each solution $P_i (1 \leq i \leq Q)$, we calculate the lifetime of each cluster by using the linear program. The minimum of the lifetime of all clusters is the actual network lifetime of P_i . Let $x_i (1 \leq x_i \leq M)$ denote the cluster ID sensor v_i belongs to. Each cluster formation can be represented by $x_1 x_2 \dots x_N$. Let $P_1 = p_{11} \dots p_{1N}$ and $P_2 = p_{21} p_{22} \dots p_{2N}$ be 2 solutions with the longest network lifetime in the solution set, in which $p_{ij} (1 \leq p_{ij} \leq Q)$ equals to the cluster label of sensor v_j in solution $P_i (1 \leq i \leq 2)$. To perform the crossover function, we first perform a permutation on P_1 so that the resulting P_1' has the same value as P_2 on most positions. In order to achieve this, we use the following greedy algorithm. First we build a $Q \times Q$ matrix H , so that

$$H_{ij} = |\{k \mid p_{1k} = i \text{ and } p_{2k} = j\}|$$

Let H_{lm} be the maximum element of H . We select H_{lm} from H and mark all the elements on row l and column m as -1 . Then we select the next largest element in H and repeat this procedure again until all the elements in H are -1 . For all the H_{ij} we have selected, we set $p'_{1k} = j$ if $p_{1k} = i$ for all the $1 \leq k \leq N$.

To perform the crossover, we let child C_1 and C_2 keep the same value at position k if $p'_{1k} = p_{2k}$. If $p'_{1k} \neq p_{2k}$, we randomly select a child $i (1 \leq i \leq 2)$ and set $c_{ik} = p'_{1k}$. For the other child, we set its value to p_{2k} on position k . Mutation is performed by randomly choosing one sensor, and assign it to a random cluster.

We simulated our algorithm with the GAOT toolbox [9] using Matlab. When the cluster number equals to 0 or N , the algorithm is equivalent to letting all sensors send to the base station directly. As shown in Figure 2, a suitable cluster number would be around 5% of the total sensor number.

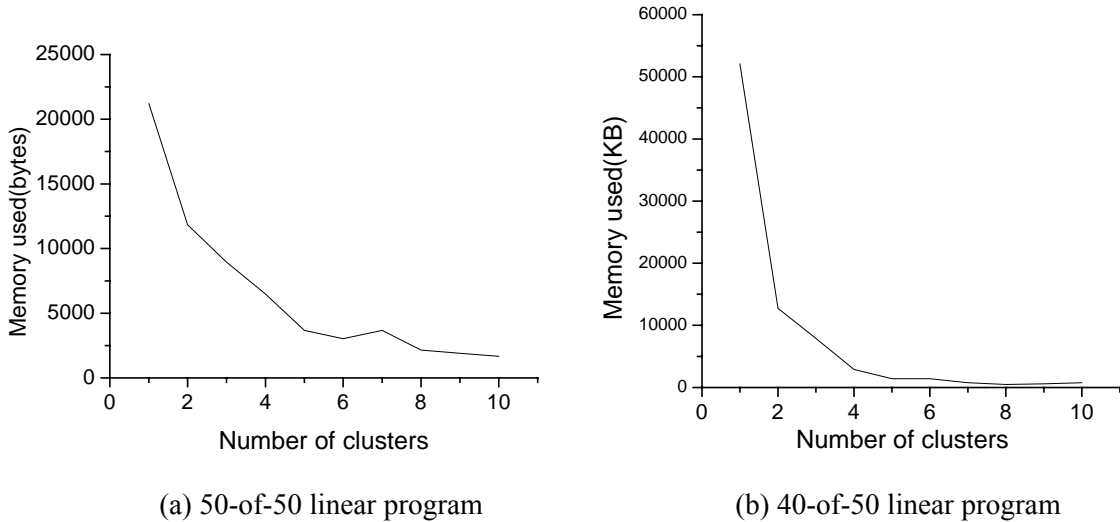


Figure 4. Memory requirement for the linear program solver as a function of the number of clusters. $N=50$, $S=20 \times 20 \text{ m}^2$, $B_{\text{dist}}=100 \text{ m}$.

To analyze the computational complexity, we assume that sensors join clusters randomly in all solutions generated throughout all the iterations. This is true for cluster formations in the initial solution set. However, it might not be *truly* random for the clusters obtained by the crossover and mutation function. In each generation, the two new child cluster formations are dependant on the two best parents in the solution set. However, since the first generation is random and performing permutations might

introduce randomness to the solution, our assumption is an approximation of the actual cluster formation. For a random cluster formation, the probability that a sensor joins a particular cluster is $\frac{1}{M}$. So the probability that a particular cluster has k sensors is

$$p_k = \binom{N}{k} \left(\frac{1}{M}\right)^k \left(\frac{M-1}{M}\right)^{N-k}$$

According to Section 3.1, the complexity for solving the linear programming model of k sensors is $O(k^{5.5})$. Thus the complexity for calculating the lifetime of a network of M clusters is bounded by $O(M \sum_{k=1}^N k^{5.5} p_k)$. For each generation, there are at most two child solutions generated. If we have G

generations in our algorithm, the total complexity for this algorithm is $O((Q + 2G)M \sum_{k=1}^N k^{5.5} p_k)$. The actual complexity is much lower since the interior point algorithm converges very fast.

As we have analyzed in Section 3.3, the space complexity of the linear program model is exponential to the number of sensors. To reduce the space complexity, our genetic algorithm can divide a large linear program into several small-scale linear programs. For example, the K -of- N lifetime problem in Section 3.3 needs more than 800MB space to store the input matrices when $N=100$. If we use 10 clusters and sensors are evenly distributed into each cluster, then we might only need a more manageable 100KB of memory to solve the linear program for each cluster. Figure 4 shows the actual memory used for solving the 50-of-50 lifetime and 40-of-50 problem as a function of the number of clusters. As shown in Figure 4, the K -of- N lifetime has a much higher memory requirement than the N -of- N problem. Figure 3(b) shows the resulting cluster formation by using the genetic algorithm. Sensors close to each other are not grouped into the same cluster. However, this may introduce transmission collision and scheduling problems. To solve this problem, a TDMA (Time Division Multiple Access) schedule is needed among video sensors. How to devise the TDMA schedule is beyond the scope of this paper.

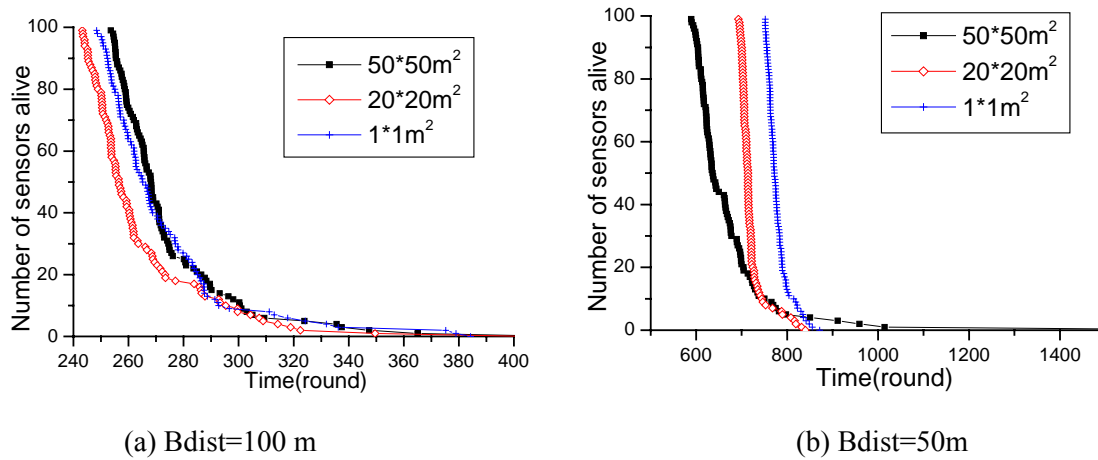


Figure 5. Effect of sensor area on the N -of- N network lifetime, cluster=5, $N=100$.

Figure 5 shows the effect of the area in which sensors are located on the lifetime of all sensors. We continuously run the MLSC algorithm until all sensors in the network expire. In Figure 5(a), sensors are far away from the base station. A larger portion of the energy is spent on communication between cluster heads and the base station. Thus distances between sensors and the cluster heads do not have much impact on the N -of- N network lifetime. When network size is small, the communication cost inside a cluster is low. Thus a network of size $1*1 \text{ m}^2$ has a longer lifetime than a network of size $20*20 \text{ m}^2$. However, when the network size continues to grow, part of the sensors become closer to the base station

since the center of the network remains fixed. This can save energy consumption rate when electing those sensors as cluster heads. For example, in Figure 5(a), sensors in a $50*50\text{m}^2$ square area have a longer N -of- N lifetime.

As shown in figure 5(b), when sensors are closer to the base station, a large sensor area results in a shorter lifetime. Cluster heads tend to be far away from all sensors when the network size grows. Thus large sensor area consumes more energy in order to communicate with the cluster heads. The N -of- N lifetime of the sensor network is decreased by 170 rounds as the sensor area increases from $1*1$ to $50*50\text{m}^2$.

5. Experimental results

To verify the performance of our approach, we compared our MLSC protocol with direct transmission, LEACH and PEGASIS. For LEACH, the cluster number is set to 5% of the number of sensors in the network. We use a two-level hierarchy for the PEGASIS protocol. Sensors are organized into ten clusters and a round-robin rotation strategy is used for each cluster. At the second level, cluster heads are also organized into a chain with a similar head rotation strategy.

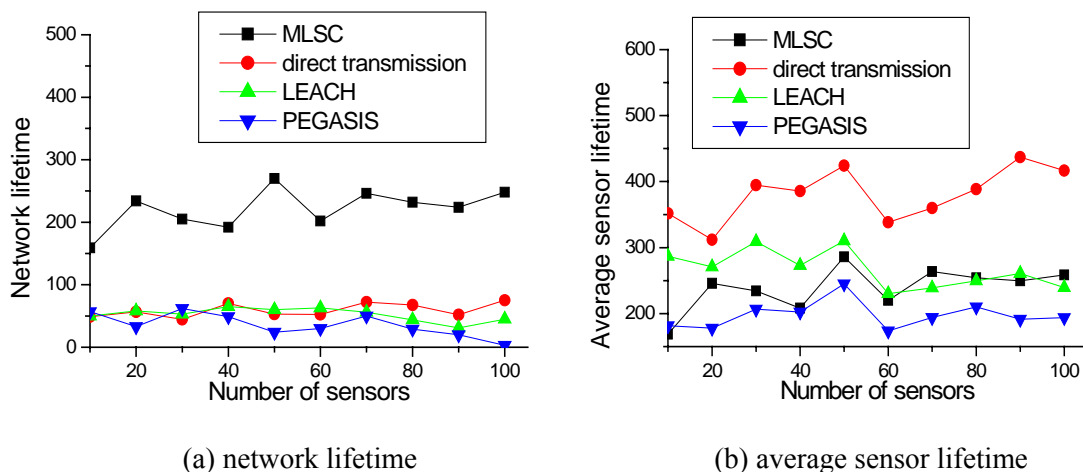


Figure 6. Comparison of video sensor network lifetime, $N=100$, $B_{\text{dist}}=100$, cluster=5.

Figure 6 shows both the video sensor network lifetime and the average sensor lifetime generated by these three algorithms. Compared with the other three approaches, our MLSC algorithm performs very well in extending the network lifetime. In Figure 6(a), the MLSC algorithm can achieve a network lifetime of up to 5 times longer than that generated by the other three algorithms. PEGASIS works poorly compared to other algorithms. Because data are forwarded along a chain until they reach the cluster head, the same data may get transmitted multiple times among sensors. Sensors with low initial energy can easily become a bottleneck and the N -of- N network lifetime is greatly reduced.

In Figure 6(b), we continuously run the four algorithms until all sensors drain their energy. Then we calculate the average of all sensors' lifetime. Direct transmission has the longest average sensor lifetime among all four algorithms. For the other three clustering algorithms, the network needs to consume additional energy to sacrifice the power of some sensors and extend the lifetime of others. Thus the average sensor lifetime is diminished. It is very likely that clustering algorithms may introduce additional energy dissipation in the sensor network. However, the network lifetime is prolonged by sacrificing the energy of some sensors to help others.

Figure 7 compares the number of sensors alive as a function of the time passed. The total number of sensors in the network is 100. In Figure 7(a), sensors are assigned a random transmission rate from 2k to 20k bits per round. By using the MLSC algorithm, all sensors die nearly at the same time. To test the

effect when video data processing and data aggregation is feasible, we use a uniform 2kb/round transmission rate for all sensors in Figure 7(b). Under both situations, the MLSC algorithm can achieve a longer network lifetime (when all 100 sensors are alive) than the other three algorithms. PEGASIS behaves quite differently with or without data aggregation. With data aggregation, each sensor sends and receives only one message. This saves a large amount of energy. PEGASIS outperforms MLSC in the last 70 nodes' lifetime when using N-of-N linear programming model. However, if we let 70 nodes work together by using the K-of-N lifetime model, the 70-of-100 network lifetime is much longer than the time when the first 30 nodes die in PEGASIS. Thus choosing a suitable K is very important when comparing results with other algorithms.

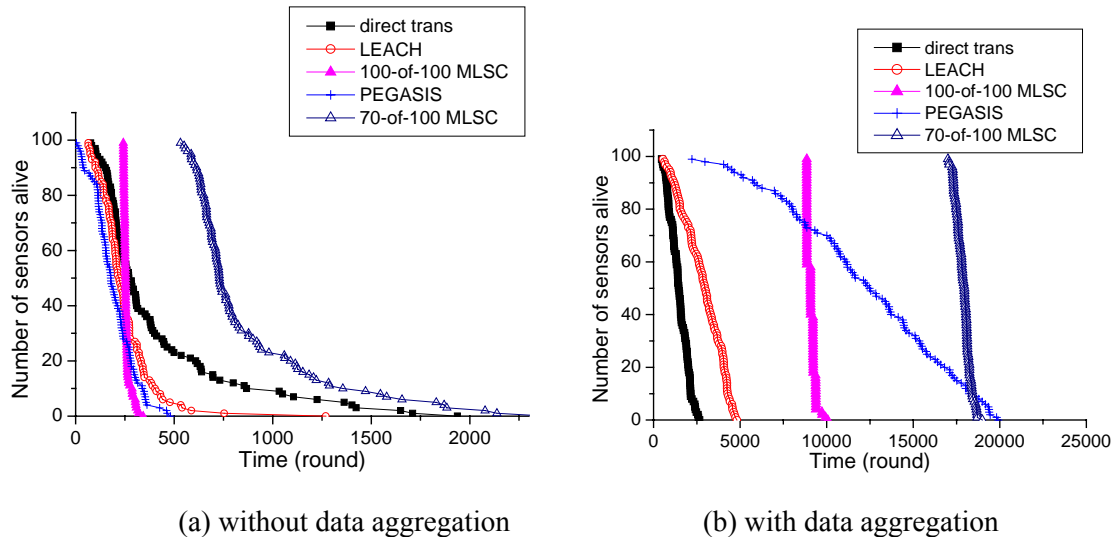


Figure 7. Comparisons of living sensors over time with/without data aggregation.

6. Conclusions

In this report, we first introduced a linear programming model for maximizing the lifetime of a video sensor network. The algorithm adopts non-uniform transmission rates and a fixed cluster head rotation strategy, which makes it a suitable match for video sensor networks. It is possible to pre-determine the cluster formation and routing strategy. Subsequently, we described a genetic clustering approach that results in a robust and efficient cluster head rotation strategy. The algorithm maximizes the sensor network lifetime rather than minimizing the overall energy consumption. Sensor network lifetime achieved by our MLSC approach is three to five times longer than that of existing approaches. If data aggregation is permissible, the network lifetime can be increased even further. Unlike conventional approaches, our algorithm not only solves the N -of- N lifetime problem, but also the K -of- N problem.

Currently our linear programming model does not take transmission scheduling and collision issues into consideration. We are going to address these problems by introducing a timesharing scheme. We plan to extend our work to mobile video sensor networks. In such networks, mobile devices are not fixed at specific locations. Currently our algorithm allows only one hop clusters. It will be extended to support multi-hop clusters. Furthermore, unexpected node failures are not currently handled in MLSC but will be considered as part of our future work.

References:

- [1] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, "Max-Min D-Cluster Formation in wireless Ad Hoc Networks," in *Proc. IEEE INFOCOM*, March 2000.

- [2] S. Bandyopadhyay and E. J. Coyle, An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks, in *Proc. IEEE INFOCOM*, San Francisco, April, 2003
- [3] S. Basagni, "Distributed Clustering for Ad Hoc Networks", in *Proc. International Symposium on Parallel Architectures, Algorithms and Networks*, June 1999. pp. 310-315
- [4] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks," in *Proceedings of ICC 2001*, June 2001.
- [5] M. Chatterjee, S. K. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks", *Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking*, No. 5, 2002, pp. 193-204.
- [6] C.F. Chiasserini, I. Chlamtac, P. Monti, A. Nucci, "Energy efficient design of wireless ad hoc networks", in *Proceedings of European Wireless*, February 2002.
- [7] D. Hall, "Mathematical Techniques in Multisensor Data fusion," Artech House, Boston, MA, 1992.
- [8] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks." *Proc. Hawaiian Int'l Conf. on Systems Science*, January 2000.
- [9] C. Houck, J. Joines, and M. Kay, "A Genetic Algorithm for Function Optimization: A Matlab Implementation," *NCSU-IE TR 95-09*, 1995.
- [10] C. Intanagonwiwat, R. Govindan and D. Estrin "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000, Boston, Massachusetts.
- [11] K. Kalpakis, K. Dasgupta, and Parag Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," In *Proc. of the 2002 IEEE International Conference on Networking (ICN'02)*, Atlanta, August, 2002. pp. 685-696.
- [12] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, *Combinatorica*, 4 (1984), pp. 373-395.
- [13] W. Kumwilaisak, Y.T. Hou, Q. Zhang, W. Zhu, C.-C. J. Kuo and Y.-Q. Zhang, "A Cross-layer quality of service mapping architecture for video delivery in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 10, pp. 1685-1689, December 2003.
- [14] S. Lindsey, C. S. Raghavendra, "PEGASIS: Power Efficient Gathering in Sensor Information Systems," *2002 IEEE Aerospace Conference*, March 2002, pp. 1-6.
- [15] A. B. McDonald, and T. Znati, "A Mobility Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1466-1487, Aug. 1999.
- [16] O. Younis and S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", in *Proc. IEEE INFOCOM 2004*, Hong Kong, China, March, 2004.
- [17] J. Pan, Y.T. Hou, L. Cai, Y. Shi, and S.X. Shen, "Locating base-stations for video sensor networks," in *Proc. IEEE Vehicular Technology Conference (VTC 2003)*, Orlando, Florida, October 4-9, 2003.
- [18] Wu-chi Feng, Brian Code, Edward C. Kaiser, Mike Shea, Wu-chang Feng: "Panoptes: Scalable low-power video sensor networking technologies". *ACM Multimedia 2003*: pp. 562-571.
- [19] Y. Zhang,, "Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment," *Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, Technical Report TR96-01*, July, 1995.