Maximizing Lifetime Vector of a Wireless Sensor Networks using Distributive Progressive Algorithm

Saravanan[1], kiranprasath[2], snehalatha[3], jeyasudha[4]

[1]*student, software engineering, SRM University, Tamilnadu, India*

[2]*student, software engineering, SRM University, Tamilnadu, India*

[3]*Assistan professor, software engineering, SRM University, Tamilnadu, India*

[43]*Assistan professor, software engineering, SRM University, Tamilnadu, India*

ABSTRACT

Maximizing a operational lifetime of a sensor network is a critical problem in practice. Many prior works define the network's lifetime as the time before a first sensor in the network runs out of energy. However, when one sensor dies, the rest of a network can still work, as long as useful data generated by other sensors can reach the sink. More appropriately, it should maximize the lifetime vector of the network, consisting of the lifetimes of all sensors, sorted in a ascending order. For this problem, there exists only a centralized algorithm that solves a series of linear programming problems with high-order complexities. The paper proposes a fully distributed progressive algorithm which iteratively produces the series of lifetime vectors, each better than the previous one. Instead of giving the optimal result in one shot after lengthy computation, a proposed distributed algorithm has a result at any time, and the more time spent gives a better result. We show that when the algorithm stabilizes, its result produces the maximum lifetime vector. Furthermore, simulations demonstrate is the algorithm that able to converge rapidly towards the maximum lifetime vector with low overhead.

Keyword:-*DPA,network,sensor,wireless,lifetime vector.*

1.INTRODUCTION

What is exactly the lifetime of a sensor network? Many prior works [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] define the network's lifetime as the time before the first sensor in the network runs out of energy, or before the first loss of coverage [11]. This definition simplifies the problem of maximizing lifetime to a linear programming problem or an NP-hard non-polynomial programming problem if the sink is allowed to move [10]. However, in reality, the operational lifetime of the network is not limited to the smallest lifetime of all nodes. When one sensor dies, the rest of the network can still work, as long as useful data generated by other sensors can still reach the sink. It is not true that, since sensors around the sink forward others' data, they will always exhaust their energy first and prevent the rest of the network from reaching the sink. One can deploy more sensors around the sink, use larger batteries to boost the enrgy level of the network. Collecting the complete information about the network and uploading the complete forwarding policies to all nodes require significant amount of transmissions in the network.Thus the enrgy of the node does not drained easily. to maintain the energy levels of the network,the distributive progressive algorithm proposed.An appropriate definition for the lifetime of a sensor network should include the lifetimes of all sensors that produce useful data. A sensor's lifetime is the duration from the time when it begins to generate the first data packet to the time when it generates the last packet that is deliverable to the sink. The network's lifetime can be defined as the vector of all sensors' lifetimes sorted in ascending order, which is called the lifetime vector. The value of the lifetime vector is determined by the nodes' packet forwarding policies that specify how packets are forwarded from the sensors through the network to the sink. More specifically, for every node, its forwarding policy specifies the proportion of packets that should be forwarded on each outgoing link towards the sink.Hou et al. [12], [13] define the problem of maximizing a sensor network's lifetime as to find the packet forwarding poli-cies for all nodes that collectively produce the lexicographically largest lifetime vector, called the maximum lifetime vector. In less precise terms, it first maximizes the smallest lifetime of all nodes, then maximizes the second smallest lifetime of all nodes, and so on. Hou et al. show that this problem can modeled as a series of linear programming (LP) problems.

After solving the LP problems, the sink uploads the optimal packet forwarding policies to the sensors. Based on its forwarding policy, each sensor forward its packets. Such a solution is however a centralized one. It requires solving $O(|N|)$ LP problems of size $O(|E|)$, where $|N|$ is the number of sensors in the network, $|E|$ is the number of links, and LP has high-order polynomial complexity. The computation overhead can be prohibitively high for large sensor networks that need to be operational soon after deployment. Collecting the complete information about the network and uploading the complete forwarding policies to all nodes require significant amount of transmissions in the network, particularly for nodes around the sink. To avoid these problems, a distributed algorithm that spreads the overhead evenly on all nodes becomes important.This paper presents the first distributed solution for the problem of maximizing the lifetime vector of a sensor network. Our strategy is to design a distributed progressive algorithm that works in a series of iterations, each producing a result (in our case, a lifetime vector and its corresponding forwarding policies) that is better than the previous one. The sequence of results approaches to the optimal solution. A distributed progressive algorithm is practically attractive because a result is available at any time and is getting better as more time is spent. We show that when the algorithm stabilizes, its result produces the maximum lifetime vector. We have performed thousands of simulation runs on random networks of various sizes, and compared with Hou's centralized algorithm as well as other related algorithms. The results demonstrate that our algorithm rapidly converges to the maximum lifetime vector and its overhead is small. For networks of thousands of nodes, it produces near optimal results in 10 to 30 iterations — one iteration requires each node to transmit two small control messages. The algorithm scales well as its overhead increases slowly with respect to network size. When used as a centralized algorithm, it is two to three orders of magnitude faster than Hou's linear programming solution for random networks of thousands of nodes; the performance gap increases for larger networks. We also compare the proposed algorithm with other existing algorithms that maximize the smallest sensor lifetime in the network or perform minimum-power routing. DPA pro-duces much better lifetime vector.

1.1 NETWORK MODEL AND PROBLEM DEFINITION

Let N be the set of sensor nodes, among which the subset *S* that generate new data are called data sources, which may be the aggregation nodes representing local clusters [12], [13]. Let g_i, $i \in N$, be the source rate at which node *i* generates new data packets. $g_i > 0$ if $i \in S$; $g_i = 0$ if i _∈S. We assume that the source rates are set low enough to not cause congestion in the network. The sink may consist of multiple geographically dispersed base stations. Assume the base stations are externally connected to a data collector. It makes no difference which base station a data packet is routed to.Two nodes are neighbors if they can receive packets from each other (to support DATA/ACK exchange). There may be multiple routing paths from each node to the sink. Let D_i be the set of neighbors that node *i* use as the next hops to the sink. They are called downstream neighbors of node *i*. $\forall j \in D_i$, (i, j) is called an outgoing link of *i*. Let U_i be the set of upstream neighbors, which use *i* as the next hop on their routing paths to the sink. $\forall k \in U_i$, (k, i) is called an incoming link of *i*. If *i* is a downstream neighbor of *k*, then *k* must be an upstream neighbor of *i*. Let $E = \{(i, j) \mid \forall i \in N, j \in D_i\}$. We call the graph consisting of all these links as the routing graph of the sensor network, which contains all routing paths from data sources to the sink.

1.2 Volume Schedule

The volume $v(i, j)$ of a link (i, j) is defined as the number of packets transmitted on the link over the lifetime
of the sensor network. The source volume $v(i)$ of a node *i* is defined as the number of new data packets generated by *i*. All link volumes and source volumes together form a volume schedule. There are many possible volume schedules, but not all of them can be actually realized. A volume schedule is feasible only if it satisfies the following energy and volume conservation constraints. Let e_i be the energy available at node *i*. Let α be the amount of energy that a node spends on receiving a data packet from an upstream neighbor, β_i be the amount of energy that node *i* spends on producing a new data packet, γ_i be the amount of energy that node *i* spends on sending a packet. The *energy constraint* is given below.

$$\sum_{k \in U_i} \alpha \times v(k, i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i, j) \le e_i, \quad \forall i \in N \qquad (1)$$

We say a node *i* is *exhausted* if

$$\sum_{k \in U_i} \alpha \times v(k, i) + \beta_i \times v(i) + \sum_{j \in D_i} \gamma_i \times v(i, j) = e_i.$$

The volume conservation constraint depends on the application model. If the application requires raw data to be delivered from sources to the sink, then the number of packets sent by a node is equal to the number it receives, i.e.,

$$\sum_{j\in D_i} v(i,j) = v(i) + \sum_{k\in U_i} v(k,i), \quad \forall i \in N. \qquad (2)$$

If it requires periodic measurement of min/max/avg among readings from sources that have not exhausted yet and remain reachable to the sink, then a node will send a packet for each set of packets received from its upstream neighbors or generated locally. The constraint becomes

2.NECESSARY AND SUFFICIENT CONDITIONS FOR MAXIMIZING LIFETIME VECTOR

This section establishes the theoretical foundation of our distributed algorithm for maximizing the lifetime vector.The volume of a (directed) path is defined as the minimum volume of the links on the path. A path in the routing graph

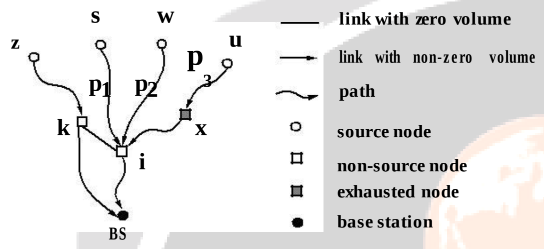

Fig. 1: There is no exhausted node on P_1 or P_2; nodes *s* and *w* are unrestricted feeding sources of *i*. There is an exhausted node *x* on P_3; node *u* is a restricted feeding source of *i*. There is no forwarding path from *z* to *i*; node *z* is a potential source of *i*. is called a forwarding path if its volume is greater than zero. Otherwise, it is called a non-forwarding path.Node $s \in S$ is a *feeding source* of node $i \in N$ if there is a forwarding path from *s* to *i*. Furthermore, node *s* is a restricted feeding source of node *i* if there is an exhausted node on every forwarding path from *s* to *i*. Node *s* is an unrestricted feeding source of node *i* if there is no exhausted node on at least one forwarding path from *s* to *i*, where the *path* referred in this definition includes *s* but excludes *i*. Node *s* is a potential source of node *i* if it is not a feeding source of *i*, but there exists a non-forwarding path from *s* to *i*, and the path has no exhausted node.We will establish the necessary and sufficient conditions for maximizing the lifetime vector in a theorem below. Below we explain a basic technique used in the proof, called volume shift. Understanding this technique will also help one to understand the design of the algorithm.Consider the routing graph in Fig. 1. Suppose *s* and *w* are two unrestricted feeding sources of node *i*. Let P_1 and P_2 be two forwarding paths that do not have any exhausted node. We show that the lifetime of an unrestricted feeding source can be increased at the expense of the lifetime of another. To do so, we simply decrease the source volume of *s*, then decrease the volumes on the links of P_1, increase the source volume of *w*, and finally increase the volumes on the links of P_2, all by the same tiny amount, which should be small enough such that its addition on P_2 does not violate the energy constraint. The above operation is called a volume shift from *s* to *w* with respect to *i*. It is easy to see that, after volume shift, the volume schedule remains feasible and the lifetime of *s* is decreased, the lifetime of *w* is increased, while the lifetimes of all other sources remain unchanged. It is obvious that, to improve the lifetime vector, we shall always perform a volume shift from a node with a larger lifetime to a node with a smaller lifetime.Not only can a volume shift be performed between two unrestricted feeding sources, but also it can be performed from a restricted feeding source *u* to an unrestricted feeding source *s*, or from an unrestricted feeding source *s* to a potential source *z*, but not the other way around — more specifically, i) *a* volume shift cannot be performed from an unrestricted feeding source s to a restricted feeding source *u* because we cannot add any additional volume to P_3 that has an exhausted node *x*;volume shift cannot be performed from a potential source z to an unrestricted feed source *s* because the volume of any path from *z* to *i* is zero and thus nothing can be shifted out.

Theorem 1*:* A feasible volume schedule produces the maxi-mum lifetime vector if and only if the following conditions are met:

1) There is an exhausted node on every path from a source to the sink.

2) All unrestricted feeding sources of a node must have the same lifetime, which should be no less than the lifetimes of the restricted feeding sources of the same node, and no greater than the lifetimes of the potential sources of the same node.

The proof is omitted due to space limitation. The above theorem gives us some guideline for designing a distributed algorithm that generates a volume schedule to maximize the lifetime vector. Below we give intuitive interpretation.Based on the first condition, data sources should aggressively set their source volumes to the highest values that their paths to the sink allow.The lifetime of a source *s*, which is $v(s)/g_s$, can be inter-preted as the average volume assigned to each unit of rate. The second condition requires that each unit of rate received by a node *i* from an unrestricted feeding source deserves the same amount of volume allocation. In other words, for unrestricted feeding sources, node *i* should allocate volumes in proportion to their rates (that *i* receives and forwards). However, each unit of rate from a restricted feeding source (which encounters an exhausted node on its forwarding path) may receive less volume allocation at node *i*. Moreover, a source should always direct its packets to paths that have highest volume allocation per unit of rate.

2.2.DISTRIBUTED PROGRESSIVE ALGORITHM

Generate data packets and deliver them to the sink, a distributed progressive algorithm (DPA) is executed to produce a volume schedule, based on which the data packets will be forwarded. on the second condition in Theorem 1. From the volume-bound distribution, it sets a volume schedule, based on which it will in turn derive a new rate schedule. This completes the first iteration of the algorithm. As shown in Fig. 2, in each subsequent iteration, DPA repeats the above computation of a new volume-bound distribution (based on the rate schedule from the previous iteration), then a new volume schedule, and finally a new rate schedule. Each iteration produces a better volume schedule whose lifetime vector is larger than the previous one.The rate schedule, volume-bound distribution, and volume schedule are stored and computed in a fully-distributed way. Each node only maintains the rates, volume bounds, and volumes of its adjacent links with a space complexity of $O(|D_i|+|U_i|)$. Because each directed link is shared by a pair of upstream-downstream nodes. Some properties of the link will be set by the upstream node and then sent to the downstream node, while other properties will be set by the downstream node and then sent to the upstream node. Details are given below.Node *i* will set its *outgoing rates*, $r(i, j)$, $j \in D_i$, by distributing the total incoming rate among the outgoing links. It will learn the *incoming rates*, $r(k, i)$, $k \in U_i$, from upstream neighbors *k* who set those rates. (We want to stress that the link rates here are auxiliary variables used to facilitate the computation of volumes. They have nothing to do with the actual data-packet rates on the links at the time when DPA is executed. In fact, DPA can be executed at the beginning of the deployment before any data packets are transmitted.)Node *i* will set its *outgoing volumes* $v(i, j)$ by distributing the total incoming volume among the outgoing links. It will learn the *incoming volumes* $v(k, i)$ from upstream neighbors *k* who set those volumes.Node *i* will set its incoming volume bounds $b(k, i)$ by dis-tributing its forwarding capacity among the incoming links. It will learn the outgoing volume bounds $b(i, j)$ from downstream neighbors *j* who set those bounds.In the rest of the section, we will describe the details of DPA, which consists of Initialization phase and iterative phase with each iteration having two steps. The first step computes volume bounds based on link rates. The second step determines link volumes from volume bounds and then computes new links rates, which sets the stage for the next iteration

3.SIMULATION

The first simulation studies how quickly DPA converges its lifetime vector to the maximum lifetime vector (MLV), which is computed numerically based on Hou’s centralized algorithm [13]. Consider the lifetime vector V_x produced by DPA after the *x*th iteration. We measure how much V_x deviates from MLV by the following two metrics. Let $t_x(s)$ be the lifetime of source *s* in V_x. Let $t_*(s)$ be the lifetime of *s* in MLV. The max deviation of V_x is defined as the avg/max deviations of lifetime vectors pro-duced by DPA on 500-node sensor networks. The deviations drop quickly to an insignificant level after a small number of iterations. For example, the avg/max deviations are merely 0.066 and 0.013 respectively after 20 iterations — that means, in the worse case, the lifetime of any source deviates from its optimal value by no more than 6.6%, and on the average case, the lifetime of a source deviates from the optimal by 1.3%.

3.1.SCALABILITY OF DPA

We evaluate the scalability of DPA on random networks of 500 to 3,000 nodes (with 20% being sources). We set a *target* (avg or max) deviation to be 0.025, 0.05, 0.075 or 0.1. We then count the number of iterations that DPA has to perform in order to produce a lifetime vector whose deviation is bounded by the target value. The simulation results are presented in Fig. 5. It shows that the *o*verhead for DPA to satisfy a target deviation, which is measured by the number of iterations, grows slowly with the network size. Recall that a node sends at most 2 smalL control packets in each iteration. Even for a network of 3,000 nodes, only 12 iterations are needed to achieve an avg deviation of 5%, and 32 iterations are needed for a max deviation of 5%.

4.CONCLUSION

We have proposed a distributed progressive algorithm for maximizing the lifetime vector in a wireless sensor network, the first algorithm of its kind for this problem. The design of the algorithm was based on the necessary and sufficient conditions that we have proved for producing the maximum lifetime vector. Simulations are performed to demonstrate the performance of the algorithm

5. REFERENCES

[1] D. Luo, X. Zhu, X. Wu, and G. Chen, "Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks," in Proc. IEEE INFOCOM, *Apr. 2011, pp. 1566–1574.*

[2] H. Wang, N. Agoulmine, M. Ma, and Y. Jin, "Network lifetime optimization in wireless sensor networks," IEEE *J. Sel. Areas* Commun., vol. 28, no. 7, pp. 1127–1137, Sep. 2010.

[3] Y. Yun and Y. Xia, "Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications," IEEE Trans. Mobile Compu., vol. 9, no. 9, pp. 1308–1318, Sep. 2010.

[4] G. Zussman and A. Segall, "Energy efficient routing in ad hoc disaster recovery networks," in Proc. IEEE INFOCOM, 2003, vol. 1, pp. 682–691.

[5] A. Sankar and Z. Liu, "Maximum lifetime routing in wireless ad-hoc networks," in Proc. IEEE INFOCOM, 2004, vol. 2, pp. 1089–1097

.