Flow Simulation Based Energy Efficient Clustering in Wireless Sensor Network

Yi JIANG
Dept. Of Communication Engineering, Harbin University of Science and Technology, China
jasonj@live.cn

Keywords: flow simulation, clustering, wireless sensor network.

Abstract. The paper introduces a flow simulation based clustering algorithm for wireless sensor network. The algorithm is fully decentralized. The proposed algorithm is inspired by simulating flow in the network. The algorithm clusters the nodes that are at distance of at most k hops from the cluster head while balance the energy dissipation among sensor nodes. By grouping sensor nodes into clusters, the performance of the mobile network, such as power dissipation, can be optimized. The preliminary simulation results show that the proposed algorithm is efficient.

Introduction

In wireless sensor network, the energy dissipation $e$ to transmit a packet to a receiver at distance $d$ can be estimated by formula [5]: $e_t = k_1 \times d + k_2 \times d^2$, where $c(2 \leq c \leq 4)$ and $k$ are constants for a specific wireless system. To receive a data packet at distance $d$ the energy can be estimated by formula: $e_r = k_1 \times d$. From those two formulas, we can see that:

- The energy used to transmit messages are larger than to receive messages.
- The energy used is greatly depend on the distance between the source and the destination of the communication channel.

To reduce the energy overhead and prolong the lifetime of the whole sensor network, it was proposed to partition the network nodes into clusters [5,7,8]. Suppose there is centroid node in each cluster that are closest to all of the clustered nodes within the same cluster, we call this node cluster head.

In our study, we propose a flow simulation based algorithm to cluster the wireless sensor node. The algorithm is inspired by [1] and [4]. The rest of the paper is organized as follows. First, the paper describes the algorithms of flow simulation, then provides the details about the clustering algorithm in wireless sensor network. Finally, a preliminary simulation about this algorithm will be presented.

Related works

MCL (Markov cluster algorithm) [4] is a centralized clustering algorithm for graphs based on flow simulation. In MCL algorithm, the graph $G$ is mapped to a Markov matrix. Then the set of transition probabilities is iteratively recomputed via expansion and inflation. Via expansion, nodes are able to see new neighbors. Via inflation, favored neighbors are further promoted and less favored neighbors are further demoted. The heuristic behind this algorithm is that a flow between sparsely connected dense regions evaporates after MCL process. Therefore, it is easy to detect dense regions in the original graph, and the dense regions are returned as clusters. MCL algorithm is considered a scalable and efficient networking clustering scheme, which can achieve high clustering accuracy.

CDC (Connectivity-based Distributed Node Clustering) [1] is a distributed and scalable algorithm for discovering connectivity-based clusters in peer-to-peer networks. The idea of the CDC scheme is to simulate flows in the network and is inspired by MCL. In CDC, the algorithm is starts from
initiating messages from set of nodes that are the originators of the clustering algorithm. The flow is simulated by messages sent from originator. We think this algorithm is scalable and it can handle dynamic network well.

But in wireless sensor network, the connectivity based algorithm is not well suitable. Because, in such environments, the network topology is changing because of node dying. Besides, the CDC algorithm consider the peer-to-peer network as an un-weighted graph, which is different to wireless sensor network. In wireless sensor network, the distance between nodes are important parameters when selecting the next hop route.

**The flow simulation based clustering algorithm**

The algorithm is inspired by MCL, and similar to CDC. But there are differences between the two algorithms. The main difference is that the algorithm considering the physical distance between nodes along with the hop distance. Let us first introduce the requirements and assumptions about the algorithm.

The cluster algorithm running in mobile sensor network must meet the following requirements:

1. Fully decentralized and asynchronized. This is the basic requirement. The asynchronized requirement is because that the mobile network is more dynamic than peer-to-peer network.
2. Avoid message overhead in network. Most of the mobile nodes are battery powered or energy limited device. The more the message forwarded, the more likely the node dead.
3. Heuristic and producing sub-optimal output. The cluster algorithm is running decentralized; only local information can be used to guide the algorithm. The heuristic algorithm is more suitable in this environment and produces sub-optimal output.
4. The selection of the cluster head node must consider the load balancing. The cluster head has high work load among all the cluster nodes, so it consumes energy faster than other nodes, if the cluster head selection algorithm cannot balance the work load among clusters, the cluster head will die quickly. As a consequent, the life time of the whole sensor network will be shortened.

Before describing the algorithm, there are some assumptions about the mobile network and mobile nodes.

1. The network has not been clustered.
2. Every mobile node knows its own position.

If node does not know its position, the node should have the ability to measure the signal strength with which node can determine the distance between the other node according to the Friis transmission equation.

As stated previously, the power consumption is largely related to the distance between local node and neighbors. In our study, the sensor node selects the one which is close to itself while have more residual energy as the cluster head. The main idea of the algorithm is simulating flows in the network as [1] does. Image the nodes in the network are intersection of the edges (links between nodes), and messages are flows. The message flow in the network, at each intersection, it drops some of its weight and continue its flow random direction along the links. Considering the following fact: a random walk in a graph that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited [4]. So we can infer that the nodes within the same density area will receive most of the weights. Then the node that one receives most weighted weights from can be selected as the cluster head.
The algorithm starts out by initiating messages from nodes randomly. In contrast to CDC algorithm, in the algorithm, there are no such THP (Two Hops Probability) process to determine the originator nodes. In our study, the aim of the algorithm is to reduce the energy dissipation, we think the process of the THP is not needed in sensor network, because the cluster message can be used to determine the originator as well. At first, nodes are not gathered into clusters. But if there are nodes needing to transfer packets to some other nodes, there are flooding in the network. We choose to piggy back the cluster message to every packets. In this way, there is no extra packet needed to transfer cluster message.

**Neighbors.** After power on, nodes turn on its radio transceiver. At this time, it broadcast a message through the wireless connection. All nodes in the area of the radio transmission respond to this message. In this way, nodes can know the neighbors in the sensor network. There are various reasons that a node will die without notification in sensor network, so the neighbors of node are changing at time. Nodes need to probe the network periodically to maintain the neighbors’ list. This is done by neighbor protocol. The neighbor protocol is very simple.

**Cluster message.** By neighbor protocol, node knows the nodes in the area of the radio transmission. Those nodes are the one hop neighbors of the local node. The algorithm will do is to gather all the nodes in one hop distance into clusters. Thus the sensor network can be partitioned into clusters according to the one hop distance. The cluster messages send from nodes to nodes are mainly within the one hop distance. Because there may be more than one node within the one hop distance from local node, the cluster message may be forwarded more than once. Similar to the data structures used in algorithm CDC, in the algorithm there are similar cluster message structure. Each cluster message is a tuple consisting of the following fields:

1. Source ID (SID): A field uniquely identifying the source of the cluster message.
2. Message ID (MID): A field distinguishing each message from all other messages from the same source.
3. Message Weight (MWeight): The weight carried by the message.
4. Previous Hop ID (PHID): This field indicating the most recent node the message traversed.
5. Max Hop Distance (MHD): This field indicating the max hop the message can be reached. This field is always set to 2 in our study.

**Message weight.** The algorithm considers not only the distances between nodes but also the energy distribution of the sensor nodes. Clustering according to the distances between nodes will reduce the energy dissipation in the sensor network. But the cluster head will out of power more quickly than the cluster member. After several iterations, the network may not alive because many of the nodes are died. Clustering according to the energy distribution only cannot make the energy usage optimal too. So we choose to cluster the sensor network on those two factors. The initial message weight in the algorithm is calculated from the following formula:

\[
MessageWeight = \frac{Energy_i}{NumNeighbors_i}
\]  

(1)

Where \(Energy_i\) is the energy level of the sensor node, which is a real number in \([0..1]\). \(NumNeighbors_i\) is the number of nodes within the one hop distance.

The message weight is designed to drop at the intermediate node \(j\) in the traversal path of the message. The dropped weight and the message weight can be calculated by the Eq.2 and Eq.3:
\[
DroppedWeight_i = \frac{MessageWeight}{2^{hops} \times e^{D_{i,j}/D_i}}
\]

\[
MessageWeight_j = MessageWeight_i - DroppedWeight_j
\]

where \(hops\) is the number of hops the message has traversed, \(D_j\) is the effective radio transmission diameter, \(D_{i,j}\) is the distance between sensor node \(i\) and \(j\). Because node knows its own position, and the position of the source node can be carried in cluster message, so the distance between nodes can be calculated simply.

If the reset of the message weight in this cluster message is greater than zero, then the message will be broadcasted to all the neighbors of node \(j\) with message weight as denoted by Eq.4.

\[
MessageWeight_j = \frac{MessageWeight_i}{NumNeighbors_j}
\]

The algorithm. The pseudo code of the algorithm is shown in Fig.1. The main process of the algorithm is similar to CDC. The differences are the weight computation and the amount of the weight to drop. Obviously, message from sensor node with high energy level carry more weight than others. Those messages drop more weight on nodes it traverses. The node far away from the candidate cluster head node receives less weight from the message. Nodes within the radio transmission range of the high energy level sensor nodes will choose the closest high energy node as cluster head. Those cluster head nodes will use up their energy more quickly than the other nodes. The algorithm will always choose the high energy node as the cluster head. So the energy dissipation is balanced in the whole network, and prolongs the lifetime of the sensor network.

```
1 TotalWeight ← TotalWeight + Msg.Weight
2 Msg.Weight ← Msg.Weight + \frac{Message.Weight}{2^{hops} \times e^{D_{i,j}/D_i}}
3 Msg.Hops ← Msg.Hops − 1
4 Msg.Range ← Msg.Range − D_{i,j}
5 if TotalWeight(Msg.SourceID) ≥ Threshold then
6     Clusterhead ← Msg.SourceID
7 else
8     if Msg.Hops > 0 and Msg.Range > 0 then
9         for all n such that n ∈ Neighbors_i do
10             Msg.Weight ← \frac{Msg.Weight}{NumNeighbors_i}
11             forward Msg to n
12         end for
13     end if
14 end if
```

Fig.1 The algorithm executed on each sensor node when cluster message received

Candidate cluster heads. This is the last section of the algorithm. This section describes the selection of candidate cluster heads. The algorithm described in Fig.1 is executed by the candidate cluster heads. If all the sensor nodes send cluster messages to all the neighbors, there will be message overhead in the network. Before the execution of the cluster algorithm, the sensor nodes need to identify itself as the
candidate. The node has more residual energy and low degree is the candidate of the cluster head, and the more residual energy and the lower degree a node has, the more probability the node will be cluster head. Choosing the node with low degree may not wise for energy-constrained sensor network because it penalizes specific nodes in the network irrespective of their battery lifetimes [6,7]. Choosing the node with high degree as the cluster head may result in quickly draining the battery of larger-degree nodes. In our study, Eq.5 is used to identify the candidate cluster head. Where $RadioRange$ is the radio transmission range of the sensor node. Eq.5 balances the energy level and the density of the sensors.

$$\text{CandidateProbability}_i = \frac{\text{Energy}_i}{\text{NumNeighbors}_i \times RadioRange^2}$$  \hspace{1cm} (5)

In the algorithm, node whose $\text{CandidateProbability}_i = \text{Max}_{j \in \text{Neighbor}_i} \text{CandidateProbability}_j$ is the cluster head candidate.

Performance evaluations

The simulation uses a network in which 100 nodes are located randomly in $100m \times 100m$ dimensions, and assumes that the energy consumption is symmetric between two arbitrary nodes in simulation. The base station is located $100m$ away from the closest sensor node and the base station is considered to be high-energy node.

We use the same radio model as [5] used. That is $E_{\text{elec}} = 50nJ / \text{bit} / m^2$ and $E_{\text{amp}} = 100pJ / \text{bit} / m^2$. Each node was initially given $0.5J$ of energy, and each message are $2000$ bits.

Fig.2 shows the result of the simulation, the first sensor died at 850 rounds, the last sensor died at 1257 rounds. The simulation states that the proposed algorithm has comparable performance to LEACH.

Conclusion

The paper introduces the flow simulation based clustering for energy efficient data gathering algorithm for wireless sensor network. The algorithm extends the generic connectivity based algorithm for peer-to-peer clustering to accommodate the wireless sensor network environments. The algorithm in this paper considers not only the distance the cluster member away from cluster head but
also the energy level of each sensor node. The cluster based communication in wireless sensor network can save a lot of energy in wireless sensor network, thus prolongs the lifetime of the system. The simulation shows that the flow simulation based clustering has good performance.

References


