Energy Efficient Network Reconfiguration Algorithm Based on Cooperation for Wireless Ad hoc Network

Sachin Kumar Soni¹, Gajendra Singh²

¹M.Tech Scholar SSSIST Sehore
²HOD CE Department SSSIST Sehore

Abstract—In a wireless multi hop ad hoc network, communication between distant nodes is done via intermediate hops working as forwards. As forwarding data packets consumes energy, intermediate hops may not wish to cooperate in communication. In this paper we are dealing with two major issues of wireless ad hoc network namely, cooperation and energy efficiency. We propose an algorithm that increases the level of cooperation of a wireless ad hoc network in an energy efficient manner, working over topology control in network along with load balancing and sleep scheduling.

Keywords—Ad hoc networks, cooperation, energy efficiency, load balancing, sleep scheduling, topology control.

I. INTRODUCTION

A Wireless ad hoc network is an infrastructure less, decentralized network. It has no central object to govern the functioning of the network. In this network, each node also known as “hop” may act as router, forwarder, source and/or destination for a data packet. As compared to wired networks, ad hoc networks are more prone to malicious attacks and different kinds of failures. In order to manage the functionalities of an ad hoc network the nodes in it must be cooperative. Non-cooperative behavior of nodes affect throughput of the network. In an ad hoc network nodes communicate with each other by relaying data packets for other nodes when there is no direct link in between. To carry out this communication successfully the nodes must show their cooperation in relaying the packets. However, as nodes in ad hoc network have limited energy associated with them, they may choose not to forward any or all the packets for other nodes. But this misbehavior will consequently reduce the overall network throughput. Detection and penalization of misbehaving nodes is done by a protocol named CONFIDANT [3]. It detects malicious nodes by means of observation or reports about several types of attacks, thus allowing nodes to route around misbehaving nodes and to isolate them [3]. A wireless ad hoc network has finite energy, so to save energy and to increase the lifetime nodes may choose not to cooperate in relaying packet. This emphatically would bring down the throughput percentage. To manage the network in such a manner that the resources available, get properly utilized and nodes in the network do not drain out, energy efficient algorithms are needed. Along with it, sleep and wake-up strategy contributes to a level in saving a node’s energy.

In this paper, we are trying to improve the level of cooperation, considering local topology control and load balancing, in an energy efficient manner with sleep scheduling working in parallel.

The remainder of this paper is organized as follows. In Section II, we present related works on cooperation in ad hoc networks, topology control, load balancing and sleep scheduling. Section III includes formulated problem. Section IV includes proposed solution of the listed problem. In Section V, we will discuss the work to be simulated.

II. RELATED WORK

The idea of this paper is all-embracing and do not particularly fall into one category of wireless ad hoc networks. It covers several issues such as cooperation, energy efficiency, load balancing, topology control and sleep scheduling.

The idea of the project starts with cooperation in network and its measurement.

A. Cooperation and its measurement

Cooperative communication is a way to improve the performance of relay transmission system working over ever-challenging wireless medium. In this domain, the work did include strategies which mainly follow tit-for-tat in order to maintain cooperation among nodes. [1] Nodes are assumed to be rational, i.e., their actions are strictly determined by self interest, and that each node is associated with a minimum lifetime constraint. The optimal throughput that each node should receive is defined to be the rational Pareto optimal operating point. Using which a distributed and scalable acceptance algorithm called Generous TIT-FOR-TAT (GTFT) has been proposed [2]. A market-based approach has also been worked upon in which nodes charge a price for relaying data packets and then use these incentives to relay their own packets. In both these approaches, the nodes which are detected as non-cooperating are punished either by isolating them from rest of the network or their packets are not forwarded by rest of the nodes.

B. Energy Efficiency

Energy is an important asset in ad hoc networks, as the working of whole network relies on the availability of energy. Energy efficiency can be achieved in many ways. The ways include [4] distributed algorithm where each node makes local decisions about its transmission power and these local decisions collectively guarantee global
connectivity. Specifically, based on the directional information, a node grows its transmission power until it finds a neighbour node in every direction. The resulting network topology increases network lifetime by reducing transmission power and reduces traffic interference by having low node degrees.

C. Load Balancing
As Wireless ad hoc networks is infrastructure less, nodes working on their own in coordination with other nodes which makes them consume energy for being on the route of different Path in the network. Some of the nodes may act as forwarder or router in multiple numbers of paths while some may idle or do the minimal. These conditions create a huge difference between the residual powers of the nodes. They require a procedure where it is made sure that there will not be any particular set of nodes that only participate in communication but the network as a whole must participate. This comes under load balancing and different load balancing algorithms in theory choose different kind of parameters that are measured and balanced.

By taking network node taking into account three factors, residual battery capacity, the number of hops on the path and the length of the interface queue which gives an estimate of load on a node, that is the average number of packets, queued up at the interface queue to be delivered to the destination. By estimating these values and averaging those over time give an approximate look of the traffic in the networks and energy left with nodes. These estimates are added in a manner such that the main contributing factor becomes hop count and average queue length when energy of the nodes is high and residual energy become the parameter of choice when energy gets depleted. The defined parameters are:

1. Route Energy (RE): The route energy of a path is the minimum of residual energy of nodes (r_ei) falling on a route. Higher the route energy, lesser is the probability of route failure due to exhausted nodes.
2. Traffic queue (t_q): The traffic queue of a node is the number of packets queued up in the node’s interface. Higher is its value, more occupied the node is.
3. Average Traffic Queue (ATQ): It is the mean of traffic queue of nodes from the source node to the destination node. It indicates load on a route and helps in determining the heavily loaded route.
4. Hop count (HC): The HC is the number of hops for a feasible path

D. Topology Control
Topology control is defined as a process where the topology of a network can be controlled by selective addition of nodes and links within a network. Due to varied usage of wireless ad hoc network in number of applications like battlefield, environmental modeling, emergency relief, topology control in ad hoc networks becomes a primary challenge where nodes can change their transmitting powers and can select different nodes for forwarding data. Topology control is defined as a process where the topology of a network can be controlled by selective addition of nodes and links within a network.

The basic goal of Topology Control is

- Maintain connectivity using the minimum transmission power.
- Maintain connectivity by moving some “router” nodes to fill in the hole.
- Enable nodes to self-organize themselves into clusters.
- Load balance with power consideration.

E. Sleep Scheduling
Research results have shown that radio energy consumption of listening or idling in ad hoc networks can be significant. Hence by turning off the transceivers of idle devices or making nodes sleep, redundant energy can be saved more efficiently. Node scheduling is an efficient power saving method but it is required to be in coordination with the routing algorithm. It should allow as many nodes as possible to turn their radio receiver off most of the time, since even an idle circuit can consume almost as much energy as an active transmitter. Also, it should forward packets between any source and destination with minimally more delay than if all the nodes were awake.

III. PROBLEM FORMULATION

A. Assumptions
Starting with a static ad hoc network with known topology, we assume that all the nodes are working as in Unit Disk Graph (UDG) i.e., with same initial energy. Nodes which are not busy being a forwarder or a source/destination will move to sleep mode from the active mode according to a sleep scheduling algorithm.

B. Problem Definition
For a static ad hoc network with known topology, the goal is to devise an algorithm that increases the cooperation level among the static nodes in an energy efficient manner.

C. Problem Description
At any instant of time, the number of nodes and the network topology is known. To increase the cooperation level, one of the factors which we have to consider is that the nodes in the network do not drain out too soon. For that, an energy efficient algorithm is to be applied and some techniques must be used which may save energy of a node when it is in idle condition.

Initially, we are assuming the network with some number of nodes in sleep mode while other nodes in active mode and the network is in running state in which routing has already been done.

We propose a technique which can increase the cooperation level in an energy efficient manner. Along with it sleep scheduling is running in parallel so that overall energy efficiency is increased. We have tried to resolve above stated upshots while deliberating issues like load balancing and topology control.

IV. PROPOSED SOLUTION

A. Assumptions
The proposed protocol will run between two successive routing updates. And the messages in the network should get delivered in real time i.e., there should be no delay.

B. Protocol
Consider a static ad hoc network with known topology having “N” number of nodes, where in routing has already been performed. In the network, out of N nodes, some
nodes are in sleeping state (say $N_s$) while the rest are active (say $N_a$) (Figure 1). So,

$$N = N_s + N_a$$  \hspace{1cm} (i)

where,

$N$ = Total number of nodes
$N_s$ = Number of nodes in sleeping state
$N_a$ = Number of nodes in active state

The protocol requires value of the cooperation to be precompiled at every of the node. For that, we will be using queues. Each node in the network is working with two queues attached with them, which are:

1) Input queue, $Q_i$ - used to store incoming packets,
2) Output queue, $Q_o$ - used to store the packets which are to be transmitted by the node.

Cooperation of a node can be calculated using two queues associated with the nodes. Cooperation can be defined as number of packets forwarded by a node.

$$Q_i = PF_i + Pr$$
$$Q_o = PF_o + Po$$
$$PF_o = Q_o - Po$$

Where

$PF_o$ = No. of Packets forwarded
$Po$ = No. of self generated packets sent
$PF_i$ = Packets to be forwarded
$Pr$ = No of packets of received packets
$PF_o$ indicates the level of cooperation.

The protocol can be divided into five modules as following:

**Module 1: Two-hop Communication**

A hop is a component part of a signals journey from source to destination. In particular, when only two hops are used, the communication is known as 2-hop communication.

**Module 2: Cluster Formation**

As the protocol starts each of the node distributes its level of cooperation in two-hop proximity. This is done through “2-Hop Prefatory Packet” which carries the node ID along with calculated cooperation value and energy remaining with the node.

Figure 2: Two Hop Prefatory Packet

Module 3: Inflicting Penalty

Each of the cluster head has some of the responsibilities associated with them. A cluster head has to:

1) Monitor the cooperation level of every node in its cluster,
2) Detect non-cooperative node and penalize them by forcefully sending them to sleep state for a certain period of time, and
3) Find alternate paths when nodes are penalized.

For this, every cluster head maintains a following table:

<table>
<thead>
<tr>
<th>Node_ID</th>
<th>Cooperation Value</th>
<th>Status</th>
<th>Time to Sleep</th>
</tr>
</thead>
</table>

Status field can have three values:

1. Node is in active State.
2. Node is forcibly sent to sleep state.
3. Node voluntary sent to sleep state.
   It compares the value of cooperation of each node with every other and finds a node with lowest cooperation value. It then sends a “Penalty Message” to that node which includes TTS i.e. time to sleep.

**Module 4: Identification of New Route**

Whenever a node with lowest cooperation level is punished and is forcibly sent to sleep state, cluster head manages the topology of the network. Suppose if there is any path from source to destination which had the punished node, cluster head has to find an alternate path. This may infix two conditions:
1) There may be one efficient path formed by active nodes. If it is so, then use it.
2) There are more than one alternate paths. This may further introduce three situations:
   a) One of them is efficient out of all the paths. If so, then use it.
   b) None of them is efficient. If it is so, then use load balancing mechanism to create an efficient path.
   c) If no node is in the proximity and even load balancing could not find an efficient path, then in this case routing algorithm is triggered by the cluster head.

**Module 5: Sleep Scheduling**

This module can be divided into two cases.

**Case I: When node voluntarily goes to sleep state**

In ad hoc network it may happen that a node remains idle for a certain period of time. In that case, it voluntarily goes to sleep state to save energy. The fixed time interval after which node goes to sleep state is referred as “Idle Time”. Whenever a node sleeps it sends an “Alert Message” to its neighbors that include time for which node would be sleeping. This sleeping time is referred to as “Sleep Interval”.

![Figure 6: Sleep Alert](image)

When in sleep state, a node periodically checks its sleep buffer which stores the packets received when the node is asleep. Sleep buffer is actually input queue but in case of sleeping nodes, we are referring it as “sleep buffer”. This periodic interval is named as “sleep buffer checking interval”.

**Requesting to cooperate**

It may happen that during topology control and load balancing i.e., while identifying a new route any sleeping node may be required to act as intermediate hop. For that, cluster head sends request to the required hop. The request gets stored in the sleep buffer of the node. But as advertised before that the clusters are overlapping, a sleeping node may receive requests from multiple cluster heads. This problem is resolved on first come first serve basis.

![Figure 7: Requesting message](image)

**Case II: When node is forcibly sent to sleep state**

The node with lowest cooperation value is punished and is forcibly sent to sleep state. When a punished node completes its punishment it sends an introductory packet to every node informing its availability.

![Figure 8: Introductory packet](image)

**V. SIMULATION**

The protocol designed is simulated in NS2 with the above drawn topology and tested the requirements of being called an energy efficient algorithm. The nodes will increase gradually as the network goes bigger.

**Initial Topology**

NAM is a TCL based animation tool for viewing network simulation traces and real world packet traces. It is mainly intended as a companion animator to the ns simulator. NAM provides a visual interpretation of the network topology created. In our protocol, we are assuming an initial network topology with 25 nodes. Following is the scenario for our protocol which contains 25 static nodes:
Sensing-
Every node in an ad hoc network transfers data packets via its neighbors. For discovering the neighbors, a node has to sense the environment for relaying the packets. In the following figure, it is shown how a node senses its environment for discovering neighbors.

Packet Transmission-
The following figure depicts how packet transmission occurs in the network. In the below exhibited figure, node 15 is transmitting a packet to node 7 and node 17 is transmitting a packet to node 9.

Simulation Result
Initially, in the network some of the nodes are in sleep state while others are active. Following snapshot lists the awake nodes out of 25 nodes.

Before running the EECR protocol, a list of neighbors within the two-hop vicinity is extracted from Traffic Generation file, stated above. Here, modified list of neighbors is shown, that excludes the sleeping nodes.

After finding active nodes, the detail of packet transmitted, received, forwarded and dropped is fetched from the trace file. Based on which cooperation level and remaining energy of each node is calculated separately.

Similarly, for all the nodes value of cooperation and remaining energy is calculated. And then average of both, the cooperation level and energy level is figured.
Now based on the level of cooperation, the cluster-heads are chosen and clusters are formed. Also, examining the value of cooperation can yield the nodes to be punished.

On penalizing the nodes, the resultant neighbor list is shown below.

Penalization of nodes engendered broken links. So, to continue the traffic load balancing and topology control is to be applied.

After performing load balancing, the punished nodes are released and hence they change their mode from inactive to active.

Similarly, the above simulation is carried out with different number of nodes in the network.

Simulation Result Analysis
The EECR protocol is designed to increase the average cooperation level of a static wireless ad hoc network in an energy efficient manner. After simulating working of network without EECR and with EECR, following results came out.

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Cooperation values without EECR</th>
<th>Cooperation values with EECR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.20000</td>
<td>0.20000</td>
</tr>
<tr>
<td>10</td>
<td>0.66959</td>
<td>0.67333</td>
</tr>
<tr>
<td>15</td>
<td>0.66991</td>
<td>0.77354</td>
</tr>
<tr>
<td>20</td>
<td>0.76309</td>
<td>0.78117</td>
</tr>
<tr>
<td>25</td>
<td>0.772950</td>
<td>0.82148</td>
</tr>
</tbody>
</table>

Graphically,
Analysis-
By studying the above graph, we can conclude that
1. As the number of nodes increases, average cooperation level both, before and after running the EECR algorithm, shows a general increase.
2. After running the EECR algorithm, the overall cooperation level of the network showed an increase, as intended.
3. When number of nodes are 5, there is no change in overall cooperation level of the network.
4. When number of nodes are 15, a substantial change in overall cooperation level of the network is encountered.

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Energy values without EECR</th>
<th>Energy values with EECR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>18387</td>
<td>18387</td>
</tr>
<tr>
<td>10</td>
<td>19542</td>
<td>19729</td>
</tr>
<tr>
<td>15</td>
<td>19892</td>
<td>19716</td>
</tr>
<tr>
<td>20</td>
<td>19410</td>
<td>19529</td>
</tr>
<tr>
<td>25</td>
<td>18507</td>
<td>19893</td>
</tr>
</tbody>
</table>

Graphically,

![Figure 21: Comparative Analysis of Energy Level](image)

VI. CONCLUSION
For an ad hoc network with 2-hop connectivity the design of this algorithm was aimed to increase the cooperation level of nodes while taking into consideration the energy constraint. It can be concluded from the simulation results that the EECR is resultant protocol that can increase the cooperation level in an energy efficient way. This is collectively accomplished with the help of many factors like topology control, load balancing and sleeps scheduling. In future, the devised protocol can be extended for mobile ad hoc network.

REFERENCES