dLEATC: Defined Load and Energy Aware Topology Control in Wireless Ad-hoc Networks

Aungon Nag Radon
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, V5A 1S6
Email: aradon@sfu.ca

Abstract—Holistic power management cannot be fully realized without load awareness in topology control problem for wireless ad-hoc networks. Finding both load and energy aware relays of any node in the network while maintaining connectivity is a computationally hard problem. Though LEATC [1] solves the problem by finding static multiple relay set of each node and rotating among them for data forwarding purpose, the term load is unspecified. Moreover, multiple relay set strategy consume huge amount of memory. In this paper a distributed memory efficient algorithm specifying load as delay is proposed that constructs single dynamic relay set of every node while maintaining connectivity of the network i.e., any random topology generated by the algorithm is a connected subgraph of the original graph.

I. INTRODUCTION

Power management is a big issue in wireless ad-hoc networks. Hence energy aware topology control problem got big attention among researchers in past decade. Energy aware topology control problem basically asks to find energy saving information transmission approach in connected wireless ad-hoc networks. The philosophy of energy aware topology control algorithm is to replace longer communication link with one or more shorter links. Each node identifies relay and non-relay nodes among its neighbors and removes link from its non-relay nodes for avoiding direct communication with them. So, a node keeps direct communication link with only its relay nodes to reach all non-relay neighbors using one or more relays.

Among many energy aware topology control algorithms SMECN [2] and MECN [3] are famous for their performance. These algorithms find static single relay set of each node of the network. Author’s previous work [1] highlights the necessity of having multiple relay set of each node instead of fixed single one because of following fact: Fixed single relay set strategy may choose one particular relay several time making that relay heavily loaded and ultimately creates congestion in the network.

LEATC [1] algorithm finds multiple relay set of each node using 2-hop neighbor information and dynamically rotate those relay set for packet forwarding duty. Although this algorithm is both load and energy aware but the term load was not specified. Moreover, fixed multiple relay set of each node will consume huge memory. So, how can we define the term load and save memory requirement for each node? Author’s research finds an interesting solution of this big question.

We can formulate the problem as follows: Given a graph $G = (V, E)$ where $V$ stands for nodes in the network and $E$ stands for all possible communication links within each node’s maximum power range, our goal is to generate load and energy aware subgraph $G_{LE}$ of original graph $G$. $G_{LE}$ can be generated by finding load and energy aware relay set of each vertex of $G$ and keeping link from every source node to only its relay nodes. So, the problem reduces to finding load and energy aware relay set of every node of original topology. The reduced problem actually fits with well known Set Cover problem as we have to pick suitable relays from whole bunch of neighbors of any node $s$ so that $s$ can reach any neighbor using its relays.

In this paper, author proposes a distributed algorithm that only uses 1-hop neighbor information to solve the above mentioned problem. Delay is used as a parameter to define the term load. If any relay node exceeds the cut-off value $C_f$ (initialized with 100 ms) for replying the message of source node, it is assumed that the relay node is loaded. Every node of the network keeps two types of relay set; one is active and the other one is inactive. The idea is to shift the loaded relay node from active set to inactive set after certain delay (say 50 ms), the least loaded relay from inactive set is transferred back to active relay set. The connectivity is ensured by replacing loaded relay node for example $v$ by duplicate relay node $w$ from source node’s neighbor set. The property of $w$ is such that it can be used as a relay to reach those neighbors of source node $s$ for which $v$ was the first choice for relaying purpose. If we cannot find any such $w$, we are bound to keep $v$ in active relay set for maintaining the connectivity of the network. In a nutshell, each node dynamically updates its relay set according to the load status of its relays. Dynamic single relay set strategy guarantees huge memory savings for each node of the network.

II. DEFINITION

A. Cover Set of a Node

Cover set of a node $i$ denoted by $C^s(i)$ consists of those nodes for which node $i$ is used as relay of source node $s$ to reach those nodes. So, each node of $C^s(i)$ lies in the relay region of node $i$ denoted by $R_{s\rightarrow i}$ [2]. Formally, $C^s(i)$ can be defined by following equation.

$$ C^s(i) = \bigcup_{v \in N(s)} v \in R_{s\rightarrow i} $$

(1)
III. Proposed Algorithm

A distributed algorithm referred as Defined Load and Energy Aware Topology Control or dLEATC shown in Algorithm 1 is proposed in this section. Node \( s \) is assumed as the source node. Author denotes \( A^*_R \) and \( I^*_R \) as the active and inactive relay set of \( s \) respectively. At first, \( s \) broadcasts neighbor discovery message (NDM). The nodes which response form the neighbor set \( N(s) \). Then, we construct relay graph, \( G^*_R \) of \( s \) which is described in LEATC [1]. After that relay set, \( A^*_R \) of \( s \) is formed using the idea of MECN [3] algorithm. Without loss of generality, we can guarantee that first iteration is loaded node \( A^*_R \). With rare possibility it can happen that none of the nodes of \( A^*_R \) are loaded. If this is the case then \( A^*_R \) is our desired relay set. For simplicity we can assume that relay nodes will be loaded at some time. So, we follow the strategy of relaxing the loaded relay nodes of \( A^*_R \). After forming \( A^*_R \), we construct cover set, \( C^*_s \) of every node of \( A^*_R \). Then we can run an infinite loop and inside that loop we can dynamically construct our desired \( A^*_R \). Delay of 50 ms is used in order to get the load status of every node of \( A^*_R \). The load status of every node of \( A^*_R \) is stored in a table \( T^s \) in increasing order. If load status of any node \( i \in A^*_R \) exceeds the cut-off value \( C^*_f \), we can assume that node \( i \) is loaded. We need to replace node \( i \) by best duplicate relay node \( v \) that can cover every node of \( C^*_s \). For each loaded node \( i \), if cover set \( C^*_s(i) \) is not empty we need to find duplicate relay node that can replace node \( i \) using a greedy heuristic Duplicate Relay Finder (DRF) shown in Algorithm 2.

For each node \( v \in N(s) \cap v \notin A^*_R \), if \( C^*_s(i) \) is subset of \( C^*)(v) \), we can pick that \( v \) as possible duplicate relay node for node \( i \). We store all such possible \( v \) in set \( R_i \). If \( R_i \neq \emptyset \), we can surely replace \( i \) by a node from set \( R_i \). From \( R_i \), we choose that node \( v \in R_i \) which is least loaded. We put that \( v \) in active relay set \( A^*_R \). Then we remove \( i \) from \( A^*_R \) and insert it in inactive relay set \( I^*_R \). But if \( R_i \) is empty, node \( i \) is irreplaceable as we need to keep the network connected.

IV. Correctness of Algorithm

**Theorem 4.1:** For any iteration \( i \geq 1 \), \( G^i_{LE} \) is a connected subgraph of \( G \).

**Proof:** Let, \( \pi =< u_0, u_1, \ldots, u_i, u_{i+1}, \ldots, u_k > \) be a path between node \( u \) and node \( v \) in the initial graph \( G = (V, E) \) where \( u = u_0 \) and \( v = u_k \). We will show that a link between \( u_i \) and \( u_{i+1} \) always exists in any random subgraph \( G^i_{LE} \) of \( G \). We will proof the theorem by inductive approach.

**Base case:** There exists a minimum energy aware path between any \( u_i \) and \( u_{i+1} \) in \( G^i_{LE} \). This is proved in LEATC [1].

**Induction Hypothesis:** Let us assume that a link between \( u_i \) and \( u_{i+1} \) exists in the graph \( G^i_{LE} \).

**Induction Step:** Let us assume that \( u_i \) used relay node \( x \) to reach \( u_{i+1} \) in \( G^{i-1}_{LE} \). If \( x \) becomes loaded, according to dLEATC, \( x \) will be replaced by new relay \( y \) if \( u_{i+1} \in C^i(y) \) in \( G^i_{LE} \). Otherwise, \( x \) is not replaced at all. So, \( u_i \) is still able to reach \( u_{i+1} \) in \( G^i_{LE} \). This proves the induction step and establishes the correctness of dLEATC.

V. Conclusion

This paper specifies load in terms of delay for solving load and energy aware topology control problem. Author’s proposed dLEATC algorithm generates single dynamic relay set of each node. Moreover, single relay set consumes less memory than multiple relay set. So, dLEATC clearly outperforms LEATC [1] in space requirement.

**References**

