Distributed Steiner Tree Construction in Wireless Sensor Networks with Buffer Limitations

516030910537
Yang Chenyu

June 2020

1 Introduction

The purpose of this project is to design a distributed algorithm to construct Steiner trees with consideration of buffer limitations. In a Internet of Things(IOT) scenario, the devices always need to be connected to one another through multi-cast. The decision of the topology of the connection is often described as an Steiner tree problem, That is, given an un-directed graph with non-negative edge weights and a subset of vertices, we wish to find a tree of minimum weight that contains all vertices (but may include additional vertices). The multi-cast problem is an NP-hard problem and has good heuristic algorithms to solve it with assumptions of reliable links [2], and unreliable links [1]. However, these algorithms has no consideration of the buffer limitation of the nodes, making them infeasible for some realistic scenarios. In this project, I introduce a novel algorithm that takes the buffer limitations into consideration, I tested this algorithm on random generated graphs and got good results.

2 Background

2.1 Problem setting of building multicast-tree

To design and verify the Steiner Tree construction algorithm, we have to first state the model and the assumptions of the network, with the following rules:

1. n nodes in total, each identified with a unique identifier to be distinguished from others
2. m nodes form a multicast group participating in message transmission($m < n$)
3. Density function $f(x)$ describing geographical distribution of nodes, where x is the position vector

4. Uniform transmission range r for all nodes [1]

2.2 Toward Source Tree Algorithm

To the best knowledge of the author, the best algorithm solving the problem defined in 2.1 is the Toward Source Tree (TST)[2] algorithm. The proposed algorithm in this project builds upon the TST. Thus, we have to briefly introduce TST algorithm in the first place.

As summarised in [1], the algorithm can be divided into three steps, and the details are shown as follows:

1. **Notifying Members.** The source transmits a notification message with the constant transmission range r, where $r = \Theta(\sqrt{\frac{\log n}{n}})$,and all nodes forward the message at the first time they receive it, At the end of this step, all multicast members will be notified.

2. **Connecting all Members.** After being notified, each multicast member, denoted as the sender, searches others within the coverage range r_c and if no members closer to the source are searched, r_c will be doubled and the sender will take an another search process. Similar to step 1 the sender transmits the search message with search range r, and all nodes within the coverage range relay the message at the first time they receive it. The searched multicast member may receive the same search messages from several relays, and it respond to the sender through the minimum-hop search path. After receiving those reponse messages, the sender selects the closest member among those having shorter distances to the source than the sender itself to connect to.

3. **Eliminating Cycles.** When the paths connecting different pairs of multicast members share a same relay, cycles may appear. Then a step for elimination the cycles becomes necessary. The step finds the minimum spanning tree of the graph with circle. To be specific, the relay nodes that find themselves involved in two paths choose one unnecessary path to cancel it.

3 Problem Setting of Network with Buffer Limitations

The problem setting is similar to that described in 2.1, except for one extra assumption:

- Each node have limited buffer for transition

[1] Transmission range means the maximum distance that allows a pair of node to build a direct connection

Now let's consider the way the buffer assumption might influence the way of Steiner tree construction. We model the buffer usage as the following to bridge the buffer and network topology:

3.1 Buffer usage assumptions

We assume that in the multi-cast process, each node send packages to its child nodes simultaneously. For the node i, the λ_i is the influx of the node, which is decided by the outflux of the source node, in a stable transmission, the influx is the same across the broadcast group

$$\lambda_i = \lambda_0. \tag{1}$$

The outflux of each node depends on the number of child of the node. Because when a node broadcast a message to its children, all the children have independent probability of failure. When any failure happens, the sender node should send the message again, thus slowing down the transmission. Thus, the outflux of node i can be modeled as:

$$\mu_i = \frac{\hat{\mu}}{deg(i) - 1} \tag{2}$$

According to Queueing theory, the probability for k packages to be in the queue is:

$$p_i\left(n_i = k\right) = \left(1 - \rho_i\right)\rho_i^k where \rho_i = \frac{\lambda_i}{\mu_i} \tag{3}$$

From these assumptions about transmission process, if we want the queue of each node less than the limit with probability P, we can finally derive the limitation on the number of children of each node.

3.2 problem setting with child number limitation

Thus, we can transfer the limitation of buffer to the limitation of number of children, and we can get the following rules for the network model

1. n nodes in total, each identified with a unique identifier to be distinguished from others
2. m nodes form a multicast group participating in message transmission ($m \leq n$)
3. Density function $f(x)$ describing geographical distribution of nodes, where x is the position vector
4. Uniform transmission range r for all nodes
5. **Each node have limited number of children**

4 Proposed Algorithm

In this project, we propose an algorithm based on TST. Specifically, we add an step after the TST to ensure the limitation of the number of children. The broad idea of the proposed algorithm is: Each node checks their number of children, and sends an *adoption request* if its number of children exceeds the limitation. The neighbors receiving the *adoption request* propose *solution*s if they have ability to take care of more children. Then the requested node find a proposed solution and shift the parent relationship to the node that proposed the solution. Fig1 provides a process of shifting parent-ship of the simplest case.

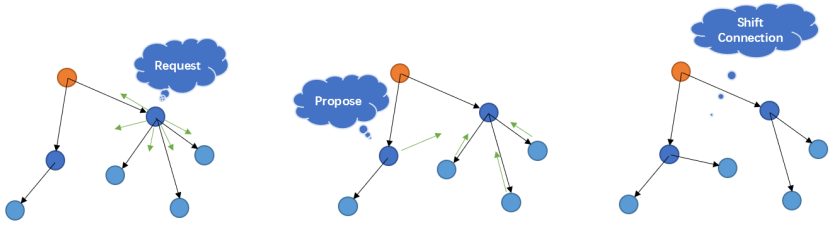

Figure 1: The idea of the three phase of proposed algorithm. The child number limitation is 2

I will introduce the algorithm in the following manner. First, I define the key concepts in this algorithm, and then introduce the procedure of the algorithm. Last, give a proof of the correctness of this algorithm.

4.1 Key Concepts

Solution Tree When receiving an *adoption request*, a node that already reaches the child number limitation can also choose to adopt this child, however, the new parent has to send another adoption request(noted as sub-request) to find another parent that can take one of its child. Having received the solution of the sub-request, it can proposed the solution to take over the child of the original requested node. Thus, the request-solution relationship between the parents forms a solution tree, as the fig2 shows.

Added Length By the definition of Steiner tree, the topology we want is a minimum-length tree with degree limitations. Thus, we have to try to minimize the extra cost of our "adoption" operations, and we introduce the *Added Length* to represent it. The Added length of a solution is the difference of the total length of the tree after and before the solution operation, shown as the fig3

New Relay When a node not belonging to the Steiner tree receives an adoption request, it can choose to respond to this request to become a *new relay*. Specifically, it an broadcast a sub-request that looks for a parent for itself. If

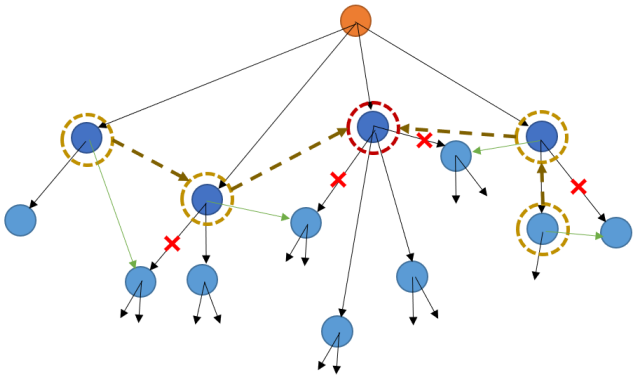

Figure 2: An example of the solution tree. The arrows with red crosses are the edges to be removed, and the green dashed arrows represents the edges to be connected. The nodes with dashed circles are nodes in the solution tree, where the red one indicates the root of the solution tree, which is also the node initiated the request

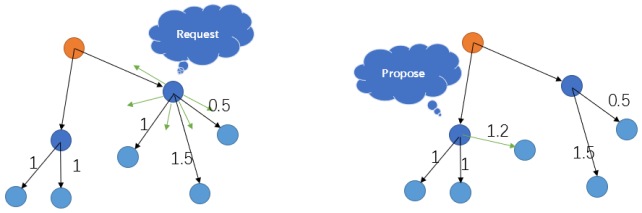

Figure 3: An example of the notion of **added length**. In this example, the added length of the solution is $1.2 - 1 = 0.2$

this sub-request is fulfilled, then the node can propose a solution to the original requested node. The idea of new relay is shown in fig4

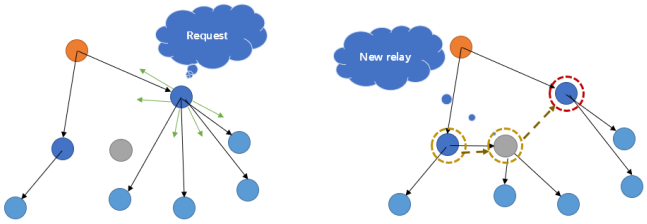

Figure 4: An example of the new relay. In this example, grey node is not originally involved in the tree, and becomes a new relay for fulling the request

4.2 Algorithm procedures

request and forward request The action of initiating a request and forward a sub-request follows the following set of rules.

1. A node checks its number of children. If that exceeds the limit, then it initiates a request.
2. Each request has the information of all the nodes to be adopted, and the current added length.
3. Upon receiving a request (either original request or sub-request), a node calculates the added length for it to fulfill the request. It maintains the requested node which has the minimum added length. It can choose to forward a sub-request.
4. When forwards, each node broadcasts all of its children in the message
5. When a node meets its limitation while cannot meet its limitation after taking a child, then it forwards a sub-request, providing the list of its children, and current added length.
6. When the added length exceeds a limit r_c, forwards terminates.

Propose Solutions When a node receives a request and find its solution to fulfill its sub-request, it proposes solutions to the requested node, following the rules:

1. Each node cannot propose a solution unless it degree limitation can be fulfilled (by its sub-request)
2. Each node finds each of its children a best solution.
3. A solution node in the subtree of children i cannot take a children j where $i.(i, j$ is a rank of nodes, ordered by distance to parent)
4. Each node sends back solutions only if its request can be fulfilled
5. Each node sends an order of preference and maximum number of children it can take, and all its ancestors A node cannot take its ancestors

Shift Parent-ship

1. Each node connect its accepted solutions and unfulfilled requests
2. Each node uses up all its degrees first before given child to other

4.3 Proof of tree

In this subsection, we formally prove that the result of this algorithm is a tree by induction over the steps of reconnect.

Theorem 1. *The result of the proposed algorithm yields a tree*

Proof. It is easy to check that there can be a total order of the nodes of the solution tree in the Breadth-first manner. We denote the solution tree of step k, as the nodes of the solution tree whose number is less than k. It is easy to check that when $k = 0$, the tree after carrying out the solutions in the solution tree of step k is a tree(as no transition of parent-ship happens). Then, we claim that: given a connected solution tree of step k, carrying out the solution tree of step $k + 1$ still forms a tree.

To show that carrying out the solution tree forms a tree, we first show that the number of edges is still number of vertexes minus one. It is because that the operations either delete an edge and then add one, or add an edge and a vertex(new relay).

Then we show that the tree is still connected by contradiction. We assume that after taking a step, the subtrees of a set of children becomes disconnected with the source. Then the new parents of the children must comes from these subtrees. This contradicts with the rule(item 3 in the *Propose Solutions* step) that a node cannot be parent of its ancestor or siblings of ancestors with higher rank. □

5 Experiments and results

5.1 length and message complexity

We tested the length and message complexity in a network model with Uniform distribution, and total number of nodes $N = 30000$, and degree limit is 2. In the following figures, the curve with label "TST" means the result of TST, and "Relay" and "No Relay" are all the result algorithm with or without *New Relay* allowed. The TST tree violates the degree limitation and our proposed algorithm can meet the degree limitations.

From the length figure, we can see that the proposed algorithm will not drastically increase the tree length. Rather, it keeps the tree length to some stable value. The tree length is larger than baseline TST because the tree is kind of "regularized" in terms of node degrees. The second figure shows the message complexity of running the algorithms. From the figure we can see that the extra messages for running these algorithms is negligible compared with the TST's message complexity.

5.2 Effect of degree limitation

The effects of degree limitation to delivery Ratio is shown in fig6, which shows the importance of our algorithm (our algorithm makes no bad receivers). With-

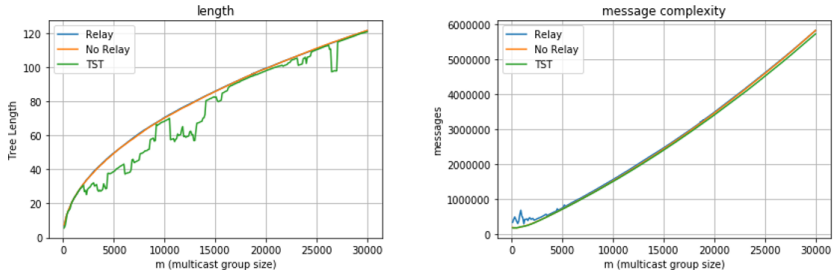

Figure 5: The experiment result of tree length and message complexity. Tested in a network model with Uniform distribution, and total number of nodes $N = 30000$, degree limit is 2

out our algorithm, the messages received by the bad receivers will be incomplete because the messages lost from limited buffer, the proposed algorithm can remedy that with negligible cost.

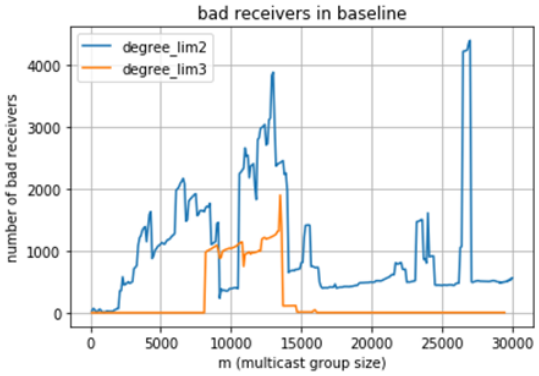

Figure 6: The number of bad receivers vs multi-cast group size

6 Conclusion

In this project, I studied the effect of buffer limitation of the multi-cast tree construction in a mobile network. I build a model which bridges the buffer limitation and the topology. Then I proposed an algorithm based on TST to ensure the limitation of the number of children of each nodes. Experiments show that the proposed algorithm can remedy buffer limitation caused package loss with negligible cost.

References

[1] Efficient Distributed Steiner Tree Construction in Wireless Sensor Networks with Unreliable Links.

[2] Hongyu Gong, Luoyi Fu, Xinzhe Fu, Lutian Zhao, Kainan Wang, and Xinbing Wang. Distributed Multicast Tree Construction in Wireless Sensor Networks. pages 1–17.