Delay Guaranteed Bandwidth-Efficient Multicast Routing in Wireless Multi-hop Networks

Hee Seok Sohn · Chae Young Lee

Department of Industrial and Systems Engineering, KAIST

Abstract

Recently, static wireless multi-hop networks such as wireless mesh networks (WMN), wireless sensor networks have been risen due to simplicity to deploy and cheap install cost. There are two key measures to evaluate performance of a multicast tree algorithm or protocol, end-to-end delay and the number of transmissions. End-to-end delay is the most important measure in QoS manner and it affects total throughput in wireless networks. Delay is same as hop count or path length from source to each destination and that is directly related to the packet success ratio. In wireless networks, each node use air medium for transmitting data, so bandwidth consumption problem is just same as the number of transmission nodes. If there are many transmitting nodes in a network, there would have a lot of collisions and queues caused by congestions. In this paper, we optimize two metrics by guaranteed delay scheme. We will provide IP formulation for minimization of the number of transmissions with guaranteed hop count and preprocessing to solve that problem. Also, we extend this scheme not just with guaranteed minimum hop count, but 1 or more guaranteed delay bound to compromise two key metrics. Then we will explain the proposed heuristic algorithm and show the performance and result in the later part of this paper.

1. Introduction and motivation

In a wireless multi-hop network, is a way to communicate data among a set of nodes which have ability to receive and send packets independently using wireless links. Each node is not able to transmit data directly to the others because of its limit of transmission range. In this paper, we focus on static wireless multi-hop network, like wireless sensor networks (WSN) or wireless mesh networks (WMN) [1].

Multicast routing means directed transmission tree from one source to many destinations and multicast routing protocols try to make a tree by minimum total edge cost. Multicast routing in communications has been considered same as Steiner tree problem (STP) because total edge cost needs to be minimized in both problems [2]. However, in wireless networks, multicast tree is not exactly same as STP due to wireless multicast advantage (WMA) [3]. Wireless multicast advantage means that a transmission of one node can transmit all neighbor nodes in wireless network due to omnidirectional characteristic of wireless broadcast. Therefore, in paper [3], [4], authors pointed that any problem in wireless networks should adapt this unique characteristic.

Wireless multicast routing tree problems are usually dealing with end-to-end delay [5], an average cost of each path from source to destination [6], [7]. Delay is the most important measures to evaluate multicast routing performance in quality of service manner. In this paper, we assume that each node has no ability to control power and fix the transmission range. Then, we will construct multicast tree with guaranteed delay, or hop count. Also, as we mentioned before, WMA should be considered to minimize average path length of tree.

One of the prominent multicast routing trees is the shortest path tree (SPT) which guarantees minimum path length of all source-destination pairs. This tree chooses the shortest path by Dijkstra's algorithm for each path and it is simply union of each shortest path. In [12], the example and shape of SPT is described. SPT is easy to exploit protocols and algorithms, and that makes hundreds of distributed protocols in wireless ad hoc networks use SPT by request and reply packet delivery. There are two prominent protocols of them and those are on demand multicast routing protocol (ODMRP) [8] and multicast ad hoc on-demand distance routing protocol (MAODV) [9].

Recently, [10], [11] suggested that reduction of bandwidth consumption in wireless networks is another key factor to affect the performance of multicast routing tree. Bandwidth consumption means the number of transmissions in networks because one transmission has its own bandwidth and consumes the bandwidth when it transmits data to neighbor nodes. The reason why the number of transmissions is serious measurement in wireless environment is that congestions and collisions originated from many transmissions in

heavy traffic condition decrease the total throughput of multicast routing. Also, that causes low success packet delivery ratio, and high energy consumption. Therefore, [11] proposed new multicast tree which is called minimum transmission node tree (MNT). In [12], the example and shape of MNT is described.

However, it has much longer path length than SPT because the algorithm does not consider delay of each path or hop count. Consequently, SPT is dealing with average path length and not with the number of transmission nodes and MNT is opposite and two metrics are not in role of cooperator, but in relation of trade-off.

So we have researched the trade-off and we will propose a novel algorithm for compromising two factors in later part of this paper.

2. Related works

Latest simulation research [12] compares the performance of the two multicast routing tree algorithms, SPT and MNT. In simulation results, MNT has less packet success ratio than SPT and the gap is bigger when the node density or traffic load is getting higher and higher in a single multicast tree scenario. It verifies that hop count is much more important than any other metrics in wireless networks because each hop has lower success probability to transmit data and total packet success ratio of a path is expressed by multiplying each ratio of hops in the path. Unlike a single multicast tree scenario, the opposite result is generated in multiple multicast trees scenario. The remarkable result is originated from the number of transmissions. Whenever new multicast group is added to network, more and more transmissions are produced by SPT compared with MNT. Thus, those over transmissions cause serious problems like congestions and collisions and drop the packet success ratio, total throughput. So, an ultimate intuition from those simulation results is that both average path length and the number of transmissions are very important metrics to multicast routing tree in wireless networks, but the delay is little bit more considerable because general networks have moderate density and traffic load

In [11], they proved NP-Completeness of MNT and proposed node minimized algorithm. They showed similar to minimum Steiner tree problems, MNT is also NP-Complete problem because this problem can be represented the well-known vertex cover problem [13], which is proven to be a NP-Complete problem. In wireless networks, simple sum of edge cost is useless due to wireless multicast advantage, thus they proposed an algorithm to construct a node minimized multicast routing tree. Main idea of the algorithm is trying to cover destinations as many as possible without considering path length concern. If one node covers many destinations, one transmission from the node will satisfy all destinations in its transmission range and consequently, that process finally constructs MNT. More sophisticated algorithm is existed and one is geographical multicast routing (GMR) [10]. It uses the information of geographical location by GPS in each sensor node. Both MNT algorithms do not consider average path length at all. Also, we previously saw that delay does the major role in wireless multi-hop networks. Thus, we can say that MNT has a problem in delay minimization.

There is a multi-channel multicast algorithm (MCM) [14] which compromises with two major metrics, the number of transmissions and the average path length. However, many ties occur when runs the algorithm but there is not any tie break. That points this algorithm is not sophisticated due to tendency to be an arbitrary SPT. Also, it guarantees only the minimum hop count for each path, so it can't provide the optimal compromise between the delay and the number of transmissions. Therefore, in this paper, we will propose IP formulation and algorithm which provide tie-break schemes and a multicast routing tree with loose hop count bound by α value.

3. Problem formulation

3-1. IP formulation

We formulate integer programming by multi-commodity flow variables usually used in wireless multicast routing problems. The model uses flow variables for each destination and can be easily applied to many multicast routing problems by a little modification of the objective function. In [15], most recent and sophisticated version of the model is used in our IP formulation and we will modify objective function and add some variables and constraints. Denote general multi-commodity flow variables as follows.

$$x_{ij}^{d} = \text{Flow from source to destination } d \text{ on link}(i,j).$$

$$z_{ij} = \begin{cases} 1 & \text{if link}(i,j) \text{transmits data,} \\ 0 & \text{otherwise.} \end{cases}$$

Denote variables and constants specialized for our algorithm, including the level of nodes,

binary variable tells a node transmit or not and defined additional hop count to loosen delay bound.

$z_i = \begin{cases} 1 & \text{if node } i \text{ transmits data,} \\ 0 & \text{otherwise.} \end{cases}$

$L(i) =$ the level of node i, *constants*.

α = the allowed number of additional hop count, *constants*.

Then, we formulate IP in network graph *G(V, E)* and source node *s* destination set *D* as follows.

[IP multicast formulation]

Objective function : *minimize* $\sum_{i \in V} z_i$ (1)

Subject to :

Flow conservation property

$$\sum_{j:(i,j)\in E} x_{ij}^d - \sum_{j:(i,j)\in E} x_{ji}^d = \begin{cases} 1 & \text{if node } i \text{ is source } s \\ -1 & \text{if node } i \in D \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Link selection

$$x_{ij}^d \le z_{ij} \qquad \forall d \in D,\ \forall (i,j) \in E \quad (3)$$

$$z_{ij} \le \sum_{d \in D} x_{ij}^d \qquad \forall (i,j) \in E \quad (4)$$

Node selection

$$z_{ij} \le z_i \qquad \forall (i,j) \in E \quad (5)$$

Hop count bound

$$\sum_{(i,j)\in E} x_{ij}^d \le L(d) + \alpha \qquad \forall d \in D \quad (6)$$

Binary variable constraint

$$z_{ij}, z_i, x_{ij}^d \in \{0,1\} \qquad \forall (i,j) \in E \quad (7)$$

(1) is the objective function which minimizes the total number of transmission nodes. The meaning of the objective in graph theory is the number of nodes in multicast routing tree except leaf nodes in graph *G(V, E)*. Constraint (2) means flow conservation in multi-commodity flow model. Constraint (3) converts flow variables to link cost. To remove those unnecessary links, constraint (4) is added. Links without any flow are not selected by the constraint. Constraint (5) selects the transmitting nodes. When any link from a node is selected to support multicast service, the node is also selected because it should send data and it is the definition of a transmitting node. The main idea of this paper is in the constraint (6). That bounds the total length of the path from source to each destination. The bound can be shortest path length of the source-destination pair or more. When we allow longer path length, the value of α is more than 0 and if we make α large enough, the optimal tree will be MNT. By this constraint, we can find the node minimum multicast tree with guaranteed path length. Last constraint (7) is just a binary constraint of variables.

3-2. Preprocessing

If the problem in graph is in case of NP-Complete or NP-Hard, preprocessing is used before optimization of LP or IP formulations. Preprocessing reduces the nodes and links which cannot be used in the optimal trees. Therefore, reduced nodes and links have an obvious reason to be deleted. Reduced topology provided by doing preprocessing promises us to be solved in short time due to low complexity of graph. This useful technique is well adapted to minimum Steiner tree problem in [16]. There are several major preprocessing schemes, but we can hardly use those things due to change of tree problem. Hop count bound in this paper is powerful role in preprocessing with the level of nodes and an intuition from [16]. We will discuss about it later, and this preprocessing is divided into three cases which are $\alpha = 0$, $\alpha = 1$, $\alpha \ge 2$. The reason why the process is changed by allowed additional hop count value is showed in follow propositions.

Proposition 1. In graph G(*V, E*), let all nodes are assigned level by BFS method. A link which directs from higher level to low level, in other words *uplink*, means at least one destination has longer path length than shortest path. Also, the longer one has at least two more edges than the shortest one.

Proof. Assigned level to a node by BFS means that the length of path is at least the value of level in any path from source to the node. So, let's assume two nodes *u*, *v* and *L(u)* = *L(d)* + 1. The shortest path length from source to node *u* is *L(u),* and the link *(u,d)* adds one hop count to the path from source to node *d*. Finally, path to node *d* is resulted by adding the path from source to node *u* and link (*u,d*). Therefore, *L(u)* + 1 is the path length of it, we have proved proposition.

Proposition 2. In graph G(V,E), let all nodes are assigned level by BFS method. A link between same level nodes, in other words *sibling link*, means at least one destination has longer path length than shortest path. Also, the longer one has at least one or more edges than the shortest one.

Proof. Exactly same as the proof of proposition 1. When link (*u,d*) is connected, length of the path from source to *d* is *L(u)* + 1 and *L(u)* = *L(d)*, so the path has at least one or more edges that the shortest path.

According to the above propositions, when $\alpha = 0$, in other words all paths from source to destinations are guaranteed to the shortest path, we can delete all *uplinks* and *sibling links* . One of the basic preprocessing in the Steiner tree problem is deletion of leaf nodes which are not destinations [16]. If we select those nodes or links to the nodes, unnecessary costs are added to the result tree. Combination of the preprocess scheme and our propositions provides highly reduced network topology. First, we delete all *uplinks* and *sibling links*, and then we have many leaf nodes. Each leaf node is easily deleted if it is not destination, also links connected to the node are can be deleted. New leaf nodes are generated by that procedure and repeat again and again from maximum level, which is the set of the nodes farthest form source, to the source. Total procedure is written below.

[Preprocessing, $\alpha = 0$]

0. Denote the destination set *D*, L*i* is the level of node *i* and each level of node is defined by BFS.
1. Delete all links between same level nodes.
2. Delete all links from higher level node to low level node.
3. Find maximum level among the destinations and we denote the level M.
4. Delete all nodes *i* and links connected to the nodes, when L*i* > M is satisfied.
5. Delete all nodes *i* and links connected to the nodes, when L*i* = M and *i* is not in *D* is satisfied.
6. Do until M is 0
   Find node set *N* whose level is M.
   Delete all nodes *i* and links connected to the nodes, which have no child when node *i* in *N* and not in *D*.
   M=M-1.

When $\alpha = 1$, *sibling links* are available due to proposition 2, so we need to modify 1, 5 process in preprocessing procedure, $\alpha = 0$. We can't delete nodes in range of destinations because one more path length is acceptable in this case. If a path arrives in shortest hop count at the node which is a sibling of destination *d*, then it is possible to reach *d* with the shortest path length + 1 hop count. Sibling means nodes that have same level and are neighbors each other. Consequently, process 1 is deleted and to modify process 5, we redefine the destination set *D* to original destinations and their sibling nodes. Modified preprocessing is as follows.

[Preprocessing, $\alpha = 1$]

0. Denote the destination set *D*, L*i* is the level of node *i* and each level of node is defined by BFS.
1. $D = D \cup$ sibling nodes of *d*, $\forall d \in D$
2. Do same as $\alpha = 0$ case from process 2

If $\alpha \geq 2$, both *uplinks* and *sibling links* are available. Therefore, we should delete process 1, 2, 4 and modify process 5 in preprocessing, $\alpha = 0$ case. Moreover, we can't delete the nodes which are in α hop range of destinations in process 5 like extension of $\alpha = 1$ case. We conclude that this modification of preprocessing will hardly reduce the topology size due to doubled number of links. We assume the efficiency of preprocessing will be very low, and do not suggest preprocessing in case of $\alpha \geq 2$.

Next figures are the result of preprocessing in random graph. Figures represent all nodes and links in graph G which has 200 nodes and 10 destinations. The source node is located in center of topology. Figure 1 is original graph and $\alpha = 0$ case is Figure 2, $\alpha = 1$ case is Figure 3. We can see $\alpha = 1$ case preprocessing didn't reduce the topology well due to bunch of *sibling links*. However, remarkable reduction in $\alpha = 0$ case makes always solve the optimal solution even in high density topology. $\alpha = 0$ case is easily optimized by IP formulation and preprocessing and we propose an algorithm to solve $\alpha \geq 1$ case.

4. Proposed algorithm

Given the difficulty of the problem in $\alpha \geq 1$ cases, we describe a heuristic algorithm which makes path length guaranteed and node minimized multicast tree by adapting α value to path length and covering many nodes once. Also, we try to find a routing tree which has lower average path length with the same number of transmissions. Then we first propose some ideas to make this kind of tree.

First, select the node which covers the destinations as many as possible. If one node can cover many destinations or nodes in pathway to other destinations and we select the node in tree, only one transmission cost makes many destinations connected. It directly exploits node minimized multicast tree. Therefore, we try to find the most covering node and select the node in our algorithm.

Second, take the branching point of tree very far away from the source node. It could be easily

seen in Figure 4, left tree uses 5 transmissions and has average path length 3, on the other hand, right one uses 4 transmissions and has average path length 4. If a tree branches earlier, each branch has to progress its objective destination and it causes more additional nodes to transmit data to same destinations. By this reason, the proposed algorithm goes from maximum level of graph to source node and try to make branch point earlier step of algorithm.

Third, prohibit uplinks in multicast tree. Previous two ideas are related to node minimized tree, but the last one works for reducing average path length. As we show in section 3, uplinks make high (two or more) cost tree in average path length manner. We can convert one *uplink* to two *sibling links* without loss of the transmission numbers and it will cover same nodes with less average path length. Best case example is in Figure 5, both trees use same number of transmissions but right one has better average path length.

We put those factors into our algorithm and newly adapted α value to loosen path length limitation. Assign current additional hop count to each node and we will make loose path length tree by check and change the value of a node. Since our algorithm runs based on BFS level assigning and adapts α to make trees, initially we assign the level by BFS and the current additional hop count with 0 to each node in graph. BFS process is as follows.

[BFS Level Assignment]

1. Assign level 0 to source node, set a node set of current level, C = {source node} and set $i = 0$.
2. Repeat
   Assign level i +1 to level-unassigned child of node n, for $\forall n \subset C$
   C consists of new assigned nodes.
   $i = i + 1$.
   Until C is empty

[α-looseness Algorithm]

Denote that

α: limit of the number of additional hop count to the shortest path length

A_n: current additional hop count value of node n

D : set of all destinations in graph

D(c) : set of all destinations with level c.

Up(n) : set of lower level nodes than n still not connected in transmission range of node n

$Dn_\alpha(n)$: set of same level node m with $A_m < \alpha$ or higher level node m with $A_m = \alpha$ in transmission range of node n including node n

0. Li is the level of node i and each level of node is defined by BFS
1. Find the maximum level among the destinations and we denote the level M.
2. Set Ai = 0 for all nodes i in graph
3. For c : from M to 1

   While $D(c) \neq \emptyset$

   Select node I, $\arg\min_{i \in D(c)} |Up(i)|$

   If tie occurs, select arbitrary one.

   Select node j, $\arg\max_{j \in U(i)} |Dn_\alpha(j) \cap D|$

   If tie occurs, select node j, $\arg\min_j L_j$

   If tie occurs again, select node j, $\arg\max_j |Up(j)|$

   Connect link(j, k), $\forall k \in Dn_\alpha(j) \cap D$.

   Set $A_j = \max_{k \in Dn_\alpha(j) \cap D} (A_k + L_j - L_k + 1)$

   $D = D \backslash \{k\}$, $k \in Dn_\alpha(j) \cap D$

   $D = D \cup \{j\}$

Brief procedure of proposed algorithm in case of $\alpha = 1$ is described in Figure 6-9.

We gave two criterions to break ties. One is node i selection criterion and the other is node j selection criterion. The former selects a node which has least upper nodes in transmission range, because the fact that a node has less upper nodes in its range means there is less chance to be selected due to less parent candidates when other nodes have more priority. The latter selects a node which has low level and most upper nodes in its range. The reason why choose low level is that if low level node is selected to the parent node of some destinations, its average path length is less than high level node because level difference of nodes is critical to path length.

5. Simulation results

In our simulation, we assume that all nodes are static and each node has same fixed power to communicate each other. This assumption is practical in wireless sensor networks or wireless mesh networks. Due to fixed power assumption, The transmission range of each node is 90m generally used in wireless mesh networks [17]. Topology size is 500m ×500m and use grid model. Note that we have maximum 5-hop shortest path to reach any node from the source because we assume 500m side and 90m transmission range in grid model. For high confidence level, we simulate 100times for all cases and evaluate average values

In Figure 10, we compare two algorithms

which guarantee shortest hop count to each destination in 100 nodes topology. Normalized number of transmissions means each evaluated value divided by optimal number of transmissions.

In Figure 11, we simulate in 300 nodes topology and we can find two things in dense networks. First, the difference between two algorithms is slightly bigger than 100 nodes. It will grow more and more in extremely dense networks due to arbitrary selection of the MCM algorithm. Second, the results of two algorithms are far from the optimal solution because dense network makes more and more candidate paths to destinations and heuristics can't find efficient multicast routing tree properly.

Although $\alpha = 0$ case is not a strong of our proposed algorithm, we showed it performs good result that the existed algorithm. Now, we will show our simulation result in $\alpha \geq 1$ cases which are the strong of our algorithm and proposed first.

In Figure 12 results of our algorithms in $\alpha = 0,1,2,3,4$ cases are shown in 300 nodes topology. Also, we control the proportion of receivers from 10% to 50%. As the result, when we allow more additional hop counts to each source-destination pair, the number of transmissions converges to some values in each case.

Figure 13 shows the average path length in $\alpha = 0,1,2,3,4$ cases are almost same each other due to same size of topology. Note that the increase of the additional hop count by 1 does not lead to increase exactly by 1 the average path length because we try to reduce the average path length in the procedure of our α looseness algorithm.

6. Conclusion and discussion

We have pointed out the problem of SPT and MNT in constructing multicast tree in wireless multi-hop networks. Each tree considers only one characteristic of wireless networks. So we have suggested delay guaranteed node minimization problem and solved by appropriate IP formulation. Due to NP-Completeness of the problem we have proposed effective preprocessing procedure and by that processing, we could have efficiently reduced in $\alpha = 0$ case. We can find all optimal multicast routing trees of harsh environments like extremely high density networks in short time. As well as $\alpha = 0$ case, we have dealt with $\alpha \geq 1$ cases, but our preprocessing is not useful anymore because of *uplinks* and *sibling links*.

In $\alpha \geq 1$ cases, we couldn't suggest adequate solutions, so we have proposed new algorithm to make multicast tree with bounded number of hop counts and minimized number of transmissions. To run this algorithm, BFS level allocation scheme is needed and the level of nodes roles in whole process. We denoted something complicated sets of nodes and constant α which means allowed additional path length to the multicast tree. By utilizing α, the algorithm makes hop count bounded tree successfully.

In simulation results, we have shown that our algorithm performs better in $\alpha = 0$ case than the existed multicast tree algorithm MCM and firstly provides a delay guaranteed multicast routing tree in any α value and topology circumstances. The main strength of proposed algorithm is the flexibility to any kind of networks. In any node density, any receiver density and any α case, our algorithm performs well and makes an adequate tree to the given network topology. Generally, it is better than other algorithm in dense networks and the results shows that $\alpha = 1$ is a proper compromise between the number of transmissions and the average path length, especially good in some graphs which have high proportion of receivers.

Furthermore, we have assigned $\alpha = 0$ to all nodes initially in our algorithm but other assigning schemes are easily exploited by changing initial α value of each node. For example, if we want to guarantee fairness among the paths, we set a proper threshold to do that. Then, destinations which have level over the threshold are assigned maximum α not to allow any additional hop count and other destinations in threshold are assigned (α — one's level) because we want to fully utilize additional hop count with guaranteed path length.