On Multi-stream Multi-source Multicast Routing

Yuh-Rong Chen, Sridhar Radhakrishnan, Sudarshan K. Dhall
School of Computer Science
The University of Oklahoma
{yrchern, sridhar, sdhall}@ou.edu

Suleyman Karabuk
School of Industrial Engineering
The University of Oklahoma
karabuk@ou.edu

***Abstract*—Multicasting is an efficient way to deliver multimedia content (streaming, for instance) to different locations in the network. While end-to-end real-time constraints are important for interactive applications, sustained availability of bandwidth is more important to the destinations for multimedia streaming. In this research, we address the problem of multi-stream multi-source multicast routing problem (MMMRP) where each data stream could have multiple sources that will serve it and each source can serve multiple data streams in a sustained manner. The goal of MMMRP is to construct a routing *forest* for each of the data streams and the destinations while maximizing the residual bandwidth. The residual bandwidth is the available bandwidth after all destinations have been served with their desired streams. Our problem is shown to be $\mathcal{NP}$-hard and we provide an Integer Programming formulation together with an efficient heuristic algorithm (*MMForests*) based on widest-path algorithm. Our empirical evaluations show that our algorithm MMForests can construct the multicast routing trees both quickly and keeping the residual bandwidth close to the optimal.**

I. Introduction

Multicasting is an efficient way to deliver the multimedia contents or large files from a single source to multiple destinations. Multicasting can be performed at the network layer taking into account the Internet routers that support Internet Group Management Protocol (IGMP) in IPv4or Multicast Listener Discovery (MLD) in IPv6. Application layer multicasting [4], [6], [13], [15] is done by the end-hosts that form the nodes of the overlay network. The links of the overlay network are paths formed by Internet routers. Application layer multicasting is very flexible in the sense that newer protocols can be easily incorporated at the end-hosts, but are less efficient because the multicasting paths may sometimes involve overlapping Internet paths [8]. Multicast backbone (Mbone) [7] uses IP tunneling to connect multicast islands and allow end users to access it.

There has been a plethora of research activity dealing with the construction of multicasting trees that satisfies various constraints. For example, the problem of constructing a single source serving a single multimedia stream wherein minimum delay is desired can be solved efficiently by constructing a single source shortest path tree and pruning subtrees that do not have a destination node. In cases where the delay bound, delay variation bound, node degree bound, and others are desired the multicasting tree constructions problems have been shown to be $\mathcal{NP}$-hard [1], [15].

There has been a growing interest in building multiple multicast trees. Castro et al. developed SplitStream [4] where they split the source stream into k stripes and multicast them using disjoint multicast trees, i.e, the trees do not share common interior nodes. The participants then obtain each stripe from different trees. S. Birrer et al. [2] address the issue of bandwidth being the bottleneck as we move closer to the root. They do this by building *fat-trees* for multicasting, wherein the outgoing links near the root have higher bandwidth compare to links that are further away from the root.

While it is possible to solve the problem by combining the multicast trees by solving each subproblems individually, the result might not be optimal. For example, say we have a source s and a destination t and there are two video streams needed to be sent from s to t and each consume 1 unit of bandwidth. If we solve the problem for each of the streams individually, we may get two edge-joint paths which consume 2 units of the bandwidth on the common edges. The optimal solution could be two edge-disjoint paths from s to t which results in 1 unit of the bandwidth usage.

Ravindran et al. [14] developed a model and provided heuristics for the problem of merging two or more trees to save operational cost, in terms of network usage or bandwidth consumption. Kar et al. [11] presented an online algorithm to address the problem of finding minimum interference routing between pairs of sources and destinations. Figueiredo et al. later developed an algorithm [9] with improved computation time. Note that the case of multiple destinations for the same source was not considered in [9], [11]. Chen et al. [5] and Lee and Cho [12] developed IP models together with heuristics to merge multicasting trees such that the minimum residual bandwidth (which is the bandwidth of link that is unused) is maximized. The main difference in the problem addressed in this paper and the work in [5], [12] is that our work also considers the case in which a multimedia stream could be served by more than one source. Additionally, we have provided a polynomial-time heuristic which is able to obtain maximum minimum residual bandwidth that is close to the optimal. Algorithms in [5], [12] have to construct a Steiner tree at each iteration, thereby incurring an expensive computation and in comparison our algorithm employs a simply widest path computation.

The rest of this paper is organized as follows. In Section II, we introduce the notations and define MMMRP, show its $\mathcal{NP}$-hardness. An integer programming (IP) formulation is then provided in Section III and the heuristic algorithm based on widest path algorithm is presented inSection IV. Performance

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

evaluation and results are presented in Section V with conclusions drawn in Section VI.

II. Problem Definition

In this section, we introduce the notations for *Multi-stream Multi-source Multicast Routing Problem* (MMMRP) and then define and model the problem.

A. Notations

Let $G = (V, E)$ be an undirected graph representing the communication network, where $V = \{v_1, v_2, \cdots v_m\}$ is the set of nodes and $E = \{e_1, e_2, \cdots e_n\}$ is the set of the communication links. Let $C = \{c_1, c_2, \cdots c_n\}$ where c_j is the bandwidth of edge e_j. When necessary, we use $G = (V, E, C)$ to denote the network with bandwidth information. We use A_i to denote the set of neighbor(s) of node v_i in G and assume each v_i has the capability to multicast to its neighbors.

Assume there are data streams supplied and requested by some of the nodes in V and let $W = \{w_1, w_2, \cdots w_l\}$ be the set of distinct data streams. We use b_k to denote the bandwidth requirement of w_k. Let $S = \{s_1, s_2, \cdots s_p\} \subseteq V$ be the set of source nodes that supply the data streams and each s_i is the source of the data stream subset $\bar{W}_i \subseteq W$. We call these nodes *source nodes* or simply *sources* in the paper. We also assume that a data stream w_k can be provided by more than 1 nodes in S. Similarly, we use $D = \{d_1, d_2 \cdots d_q\} \subseteq V$ for the set of nodes that requests some data streams, and each d_j requests data stream subset $\hat{W}_j \subseteq W$. We call these nodes *destination nodes* or simply *destinations*.

For the convenience of description, we use S_k to denote the set of source nodes that can supply data stream w_k and D_k for the set of nodes that demands data stream w_k. $|S_k|, |D_k|$ are the cardinality (the size of the set) of S_k, D_k, respectively.

Without loss of generality, we assume that:

- The links have the same bandwidth.
- A source node does not demand the data stream it supplies, it can be pruned during the preprocessing in this case it does.
- Each stream consume the same unit bandwidth.

The goal of MMMRP is to find a way to deliver the data streams to their destinations using multicast while the minimum residual bandwidth among the links is maximized. When the links are homogenous, this is equivalent to minimizing the maximum bandwidth consumption among the links. Clearly, a loose upper bound for the maximum bandwidth consumption is $|W| = l$ and a loose lower bound is $\max\lceil\frac{|\bar{W}_i|+|\hat{W}_i|}{deg(v_i)}\rceil$, $\forall\, v_i \in V$ where $deg(v_i)$ is the degree of v_i in G.

B. Problem Variations

There are special cases of MMMRP which have been studied, here we summarize these problems with MMMRP and a generalized version of MMMRP.

1. A single stream with a single source, which is similar to previous researches modeled as Steiner tree problems in [3], [15]. However, we only concern the maximum bandwidth consumption among the links, it can be solved within polynomial time using any spanning tree algorithm with pruning.
2. Multiple streams with a single source: This is similar to the problem addressed by SplitStream [4] and can be considered as a special case of 3.
3. Multiple streams with a single sources for each stream: This is addressed in [2], [14] where the total cost of the tree is considered.
4. A single stream with multiple sources: In this case, a tree with one or more nodes will be constructed from each of the source nodes and each destination must be in one of the trees. We can transform this problem to 1 by add a dummy node connecting all the source nodes.
5. MMMRP: This is the problem we are addressing in this research, which cover all the previous cases.

C. Multi-stream Multi-source Multicast Routing Problem

We consider the case where there are multiple streams $\{w_1, w_2, \cdots w_l\}$. Each of the streams w_k can be supplied by a set of sources S_k and has its destination set D_k. The goal is to find the multicast trees for each of the data streams (and their sources) such that all the destinations are part of the multicast tree of the stream they request while the maximum bandwidth usage among the links is minimizes. A more generalized version of the problem GMMMRP is as follows.

Given a network $G = (V, E, C)$, set of data streams W with each w_k consumes b_k of bandwidth and supplied by $S_k \subset V$, requested by $D_k \subset V$. Assume the nodes have the ability to multicast data to its neighbors. The objective of the problem is to find a way to deliver the data streams from some of the sources to their destinations along the links using multicast such that the minimum residue bandwidth is minimized.

D. $\mathcal{NP}$-Hardness

Given a set of source-destination pairs, the minimum interference paths (MIP) problems seeks to find the paths for each pair such that the minimum available capacity among the links is a maximum. Finding such paths has been shown to be $\mathcal{NP}$-hard by Kar et al. [11]. Now, consider a special case of MMMRP with each source serving a unique stream to a single destination, which is equivalent to the MIP problem. Hence MMMRP is also $\mathcal{NP}$-hard. Furthermore, the network flow techniques that used to find heuristic solutions for MIP problems in Kar et al. [11] cannot be applied to MMMRP as MMMRP requires multicasting as explained in the coming section.

III. Integer Programming Formulation

One of the important properties of the classic transshipment problems or network flow problems is the total supply equals the total demand for the nodes.However, a data packet can be duplicated at any intermediate nodes using multicasting. Hence the LP models for solving classic network problems can not be used directly to solve MMMRP. Here we will treat the problem as a network flow problem, but add some additional decision variables and constraints to incorporate multicasting

in this problem. The following decision variables are defined to be used in the model.

- X_{ijk}: non-negative integer variables that represent the total number of w_k's flow from the edge (v_i, v_j) when treated as a network flow problem.
- F_{ijk}: binary variables that take the value 1 if X_{ijk} is $positive$, 0 when X_{ijk} is 0. This also represents if w_k flows through the edge v_i to v_j in MMMRP.
- Z: A non-negative integer variable for measuring the maximum number of distinct data streams (F_{ijk}) flow through any of the links, which is also the objective.

We define a constant c which is an integer greater than or equal to $\max|D_k| \ \ \forall \ \ w_k \in W$. The model is shown below.

Model MMMRP

Minimize Z
Subject to:

$$\sum_{v_i \in S_k} \sum_{v_j \in A_i} X_{ijk} = |D_k| \quad \forall \ w_k \in W \tag{1}$$

$$\sum_{v_j \in A_i} X_{jik} = \sum_{v_j \in A_i} X_{ijk} + 1 \quad \forall \ w_k \in W, v_i \in D_k \tag{2}$$

$$\sum_{v_j \in A_i} X_{ijk} = \sum_{v_j \in A_i} X_{jik} \quad \forall \ w_k \in W, v_i \in V, \ v_i \notin S_k, v_i \notin D_k \tag{3}$$

$$X_{ijk} \leq cF_{ijk} \quad \forall \ w_k \in W, \ v_i \in V, v_j \in A_i \tag{4}$$

$$\sum_{v_j \in A_i} F_{jik} = 0 \quad \forall \ w_k \in W, v_i \in S_k \tag{5}$$

$$\sum_{v_j \in A_i} F_{jik} \leq 1 \quad \forall \ w_k \in W \ v_i \in V, v_i \notin S_k \tag{6}$$

$$\sum_{w_k \in W} (F_{ijk} + F_{jik}) \leq Z \quad \forall \ (v_i, v_j) \in E \tag{7}$$

The objective function Z measures the maximum bandwidth usage among the links. Constraints (1) to (3) are as used in the classic network flow problems. Constraints (1) ensure the copies a data stream d_k sent out by its source nodes equal the number of requests. On the other hand, Constraints (2) enforce the copies of incoming data stream d_k is exactly 1 more than of outgoing at a node that demands d_k. Constraints (3) assures for each intermediate node v_i, the number of outgoing and incoming copies of d_k are the same. Constraints (4) is used to determine if a data stream d_k flows from v_i to v_j. If there is at least one copy of d_k flows from v_i to v_j, F_{ijk} is set to 1 by this constraint, it could be 0 or 1 otherwise. We use constraints (5) and (6) to remove the cycles based on the following two observations: (i) there should not be any incoming data stream w_k from any neighbor of v_i if $v_i \in S_k$ (Constraints (5))and (ii) there should be at most 1 neighbor of v_i supplying data stream w_k to v_i if $v_i \notin S$ (Constraints (6)). Constraints (7) to measure the maximum bandwidth usage on each link using Z.

IV. Algorithm MMForests

We present a heuristic algorithm *MMforests* based on *widest-path algorithm* for MMMRP in Algorithm 1. The idea of MMforests is as follows.

Input: $G = (V, E)$, data stream set W, sources and destinations of each stream w_k: $\{S_k\}$, $\{D_k\}$.
Output: Set of multicast forests F

```
1  F = φ;
2  Set the capacity {c_j} of each e_j ∈ E to l (|W| = l ) ;
3  C = {c_j};
4  G' = (V, E, C);
5  Forest f_1 = DijkstraForest(G', S_1, D_1);
6  foreach e_j ∈ f_1 do
7  |   c_j = c_j − 1;
8  end
9  foreach w_i ∈ W − {w_1} do
10 |   Forest f_i = WPForest(G', S_i, D_i);
11 |   foreach e_j ∈ f_i do
12 |   |   c_j = c_j − 1;
13 |   end
14 end
15 F = {f_i};
16 return F
```

Algorithm 1: MMForests Algorithm

First, we set the capacity c_j of each communication link e_j to $|W|$ (the number of data streams), which is the loosen upper bound (line 2). Then for each of the data streams w_i, we construct a *multicast forest* f_i that spans the destination set D_i (line 5, 10) and each tree is rooted at one of the source node in S_i. The trees in f_i do not have nodes or edges in common. Then we update the residue bandwidth by subtracting 1 from c_j if e_j is in f_i, repeat this until all data streams are processed (line 6 - 8, 11 - 13).

Each *multicast forests* is constructed using Widest-Path Forest Algorithm (Algorithm 2) which is based on widest-path algorithm (a modified version of Dijkstra's algorithm) except the original Dijkstra's algorithm is used for the first data stream. We omitted the algorithm based on the original Dijkstra's algorithm because it is similar to Algorithm 2, which works as follows.

We first construct the single source widest paths for each of the sources (line 2 - 4). Then we find the path from each of the destinations to one of the sources (line 5 - 22) as follows. For each of the destinations d_j, we set d_j as the current node. There will be a "widest-path" from the current node to each of the sources. We choose the widest one among the "widest paths" and find the next node on the path (line 8 - 15). Then we add the edge from current node to next node to the f and set the next node as the current node (line 16, 17). We will repeat this procedure until one of the sources is reached then continue for next d_j. The resulting graph will be a forest where each tree is rooted at one of the source node and there's no overlapping of nodes or edges among the trees.

The reason that we do not directly construct a path from each d_j to one of the sources but instead of building the path step by step can be explained using Figure 1. Suppose we are constructing the widest path from d to one of the source nodes (S, T). We will choose the one that is wider (say to S)

Input: $G = (V, E, C)$, $S = \{s_1, s_2 \cdots\}$, $D = \{d_1, d_2 \cdots\}$.
Output: Multicast forest f

```
1  f = φ;
2  foreach source s_i ∈ S do
3  |  Tree t_i = WidestPathTree(G, s_i);
4  end
5  foreach d_j ∈ D do
6  |  current_node = d_j;
7  |  while true do
8  |  |  next_node = next node on the widest path from
   |  |  current_node to s_1 in t_1 ;
9  |  |  next_width = the bottleneck bandwidth from
   |  |  current_node to s_1 in t_1;
10 |  |  foreach s_i ∈ S − {s_1} do
11 |  |  |  if the bottleneck bandwidth from d_j to s_i in
   |  |  |  t_1 > next_width then
12 |  |  |  |  next_node = next node on the widest
   |  |  |  |  path from d_j to s_i in t_i ;
13 |  |  |  |  next_width = the bottleneck bandwidth
   |  |  |  |  from d_j to s_i in t_i;
14 |  |  |  end
15 |  |  end
16 |  |  Add (current_node, next_node) to f;
17 |  |  current_node = next_node;
18 |  |  if next_node ∈ S then
19 |  |  |  break;
20 |  |  end
21 |  end
22 end
23 return f
```

Algorithm 2: Widest-Path Forest (WPForest) Algorithm

and determine the next node on the path which is v in the example. If we keep going from v to S all the way, we may miss some "wider" paths if (d, v) is the bottleneck. In this case, two paths $d \to v \to \cdots \to S$ and $d \to v \to \cdots \to T$ have the same residue capacities. But our goal is try to use the link with higher capacities, and hence we need to make decision again at each node and so on.

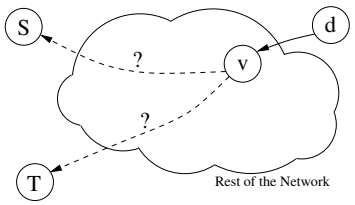


Fig. 1: Widest Path Selection.

Complexity of MMForests

First we consider the complexity of *WPForest*. The complexity of Dijkstra's algorithm is $O(n + m \log m)$, where n is the number of edges and m is the number of nodes. The loop from line 5 to 22 is $O(n \cdot |D_k||S_k|)$ for data stream w_k, hence the overall complexity is $\max(O(n + m \log m), O(n \cdot |D_k||S_k|))$. *WPForest* is called l (total number of data streams) times in *MMForests* and hence the overall complexity of *MMForests* is $O(n \cdot l \cdot |D^*| \cdot |S^*|)$ when $n > m$ and $|D^*| = \max |D_k|, |S^*| = \max |S_k|, \ \forall\ 1 \le k \le l)$,

V. Performance Evaluation

We generate 30 random instances for each different configuration using the parameters in TABLE I with the average node degree of 3. We assume a homogenous network, in which each source node supplies the same number of data streams, each destination node requests the same number of data streams and each data stream is supplied by the same number of source nodes. For simplicity, we assume that a node can be either a source node or a destination node but not both.

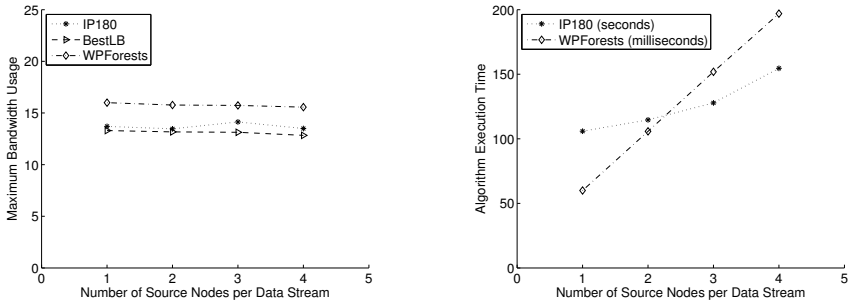
TABLE I: Parameter Used in Instance Generating

Name	Description
N	Number of nodes
N_S	Number of source nodes
N_W^S	Number of data streams supplied by a source node
N_D	Number of destination nodes
N_W^D	Number of data streams requested by a destination node
N_W	Number of data streams
N_S^W	Number of source nodes that supply a data stream ($N_S^W = N_S \cdot N_W^S / N$)

We evaluate the impact of different parameters and compare the performance of *Model MMMRP* with our heuristic *MMForests*. MMForests and Model MMMRP are implemented in C++ with Gurobi Optimizer [10] (version 4.6) C++ Library for the IP model. The workstation used in the experiments is an Intel Xeon (E5520 at 2.27 GHz) machine with 12 GB of RAM running Linux kernel 3.0.0-16. Multithreading (16) is used when possible for parallel barrier in Gurobi optimizer, *MMForests* algorithm only uses a single thread. Note that finding an optimal solution using the IP model may take a long period of time (hours to days) for large instances, Hence we limit the running time of the solver to 180 seconds and obtain *best known solution* (*IP180*) and *best available lower bound on solution* (*BestLB*). BestLB is obtained from the solver during solving the IP model. If the gap between IP180 and BestLB is small, IP180 is close to optimal. This gives us an idea about the optimal solution and allow us to compare it with our heuristic. We compare (a) *maximum bandwidth usage* (in terms of number of data streams) among the links and (b) *execution time* for both approaches in the experiments. The result figures show the average of 30 instances of each network configuration. Note that in the results of execution time, the unit for IP180 is in *seconds* and MMForests is in *milliseconds*.

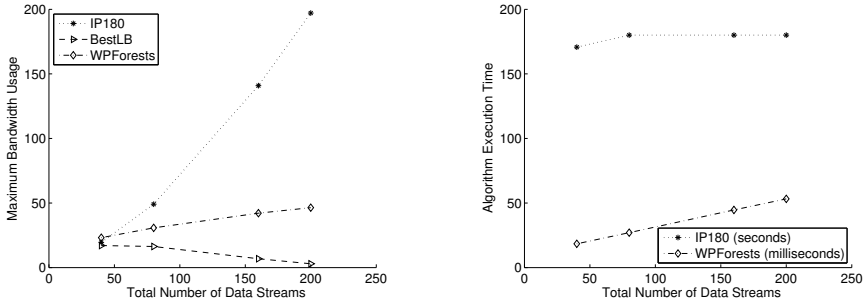
A. Size of the Network (N)

For network size (N) from 100 to 500 and keep other parameters the same ($N_S = 10, N_D = 40, N_W^S = 8, N_W^D = 20, N_S^W = 2$), We omitted the figure here due to space limitation. The results show that *IP180* and *BestLB* are very close which implies IP180 is close to optimal. The solution from *MMForest* is a little more than IP180 but with very short running time ($\le 150ms$) comparing to 100+ seconds



(a) (b)

Fig. 2: Impact of the number of sources per data stream. (a) Maximum bandwidth usage among links. (b) Execution time.



(a) (b)

Fig. 3: Impact of the total number of data streams. (a) Maximum bandwidth usage among links. (b) Execution time.

for IP180. One thing that suppress us is that BestLB does not decrease much as the network size increase, this may imply the bottleneck in the network is implicitly affected by the average degree of the nodes.

B. Number of Source Nodes for a Data Stream (N_S^W)

We manipulate the number of sources per data stream (N_S^W) while keeping other parameters the same ($N = 400, N_S = 10, N_D = 40, N_W = 40, N_W^D = 20$). Not that N_W^S is also affected by N_S^W. N_W^S are set to $4, 8, 12, 16$ with resulting N_S^W as $1, 2, 3, 4$ respectively. The results (Figure 2) shows the maximum bandwidth usage does not decrease much as N_S^W increases. Our heuristic can quickly find solutions that are close to optimal.

C. Number of Data Streams (N_W)

We experiment the effect of total number of data streams ($N_W = 40, 80, 160, 200$) and other parameters remain the same ($N = 100, N_S = 10, N_D = 80, N_W = 40, N_W^S = 8, N_W^D = 20$). We find that the solver starts failing to find a good feasible (and a good BestLB) as the number of data streams increases. The gaps between the best solution found and the BestLB value become large, especially for $N_W = 160, 200$, these information are less meaningful to us. From Figure 3, we can see that MMForests performs well when $N_W = 40$ and we also believe that the optimal value also increases when N_W increases. If our assumption holds, we have confident that MMForests finds good solutions because the slope Figure 3 is small. Note that in Fig. 3a BestLB decreases because the solver was not to improve it much (from 0) when problem sizes get larger.

D. Number of Data Streams Requested per Destination

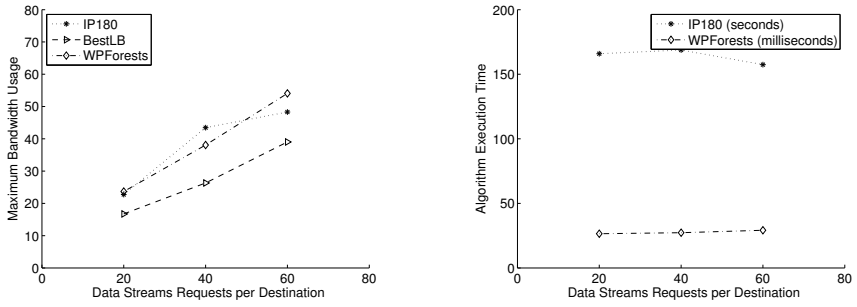
The results of manipulating the number of data streams request by each of the destinations ($N_W^D = 20, 40, 60$) are shown in Figure 4. Other parameters are $N = 100, N_S = 10, N_D = 40, N_W = 80, N_W^S = 16, N_S^W = 2$. Gurobi uses more than 150 seconds in average to find good solutions in all cases while the solutions found by MMForest is comparable with the solver (Figure 4a).

E. Total Number of Destination Nodes

In the last experiment, we change the number of destination nodes ($N_D = 40, 80, 120, 160, 200$) and other parameters are kept the same $N = 400, N_S = 10, N_W = 40, N_W^S = 8, N_W^D = 20$. Model MMMRP hits its limitation for $N_D = 120, 160, 200$ but still can provide some meaningful BestLB values. Our heuristic algorithm MMForests find good solutions (by comparing to BestLB's) by using a little more than 100 milliseconds. The result also shows N_D only slightly affects the execution time of MMForests.

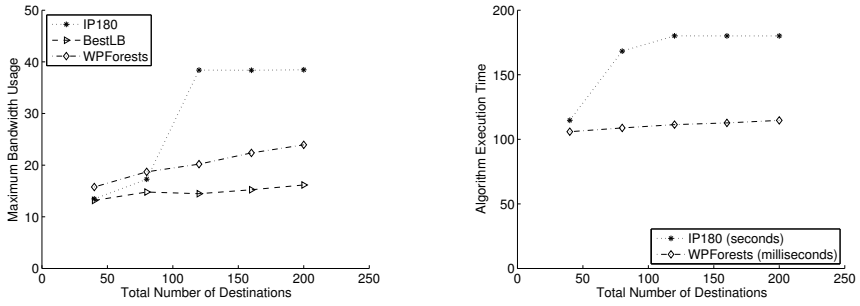
VI. Conclusions

MMMRP is a generalized version of the problem of constructing multiple multicasting trees with minimum interference. We have formulated the problem using integer programming and proposed a heuristic algorithm MMForests based on widest path algorithm for solving this problem. Our experimental results show that MMMForests finds good



(a) (b)

Fig. 4: Impact of the number of data streams requested per destination. (a) Maximum bandwidth usage among links. (b) Execution time.



(a) (b)

Fig. 5: Impact of total number destination nodes. (a) Maximum bandwidth usage among links. (b) Execution time.

multicast forests in terms of maximizing residual bandwidths while being efficient in execution time. The results also show that the execution time increases significantly in proportion to the size of the network, number of source nodes per data stream, and total number of data streams. It does not increase proportional to the number of data streams request per destination and total number of destinations. Experimental results also indicate that the structure of the network (degree of the nodes) has impact on the optimal objective value (maximum bandwidth usage among the links). The purpose of this research is to improve multimedia streaming service. In a real world situation, the streaming sessions occur in a timed manner with various bandwidth consumption and the residue bandwidth should be computed dynamically. An online version of WPForest needs to be developed to address these issues. Another issue to be addressed in the future is the overhead of gathering information necessary for the algorithm.

REFERENCES

[1] S. M. Banik, S. Radhakrishnan, and C. N. Sekharan, "Multicast routing with delay and delay variation constraints for collaborative applications on overlay networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 3, pp. 421 – 431, 2007.

[2] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda, "Fatnemo: Building a resilient multi-source multicast fat-tree," in *In Proc. of IWCW*, 2004, pp. 182–196.

[3] E. Brosh, A. Levin, and Y. Shavitt, "Approximation and heuristic algorithms for minimum-delay application-layer multicast trees," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 473–484, Apr. 2007.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 298 – 313, Oct. 2003.

[5] S. Chen, O. Günlük, and B. Yener, "The multicast packing problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 311 – 318, Jun. 2000.

[6] Y.-H. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1456 – 1471, Oct. 2002.

[7] H. Eriksson, "Mbone: the multicast backbone," *Commun. ACM*, vol. 37, pp. 54 – 60, Aug. 1994.

[8] S. Fahmy and M. Kwon, "Characterizing overlay multicast networks and their costs," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 373 – 386, Apr. 2007.

[9] G. Figueiredo, N. da Fonseca, and J. Monteiro, "A minimum interference routing algorithm," in *Communications, 2004 IEEE International Conference on*, vol. 4, Jun. 2004.

[10] Gurobi Optimizer. http://www.gurobi.com.

[11] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with mpls traffic engineering applications," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 12, pp. 2566 – 2579, Dec. 2000.

[12] C. Y. Lee and H. K. Cho, "Multiple multicast tree allocation in ip network," *Computer Operations Research*, vol. 31, no. 7, pp. 1115 – 1133, Jun. 2004.

[13] Project NICE at the University of Maryland. http://www.cs.umd.edu/projects/nice/.

[14] K. N. Ravindran, A. Sabbir, D. Loguinov, and G. S. Bloom, "Cost optimal multicast trees for multi-source data flows," in *INFOCOM*, 2001, pp. 966–975.

[15] K.-H. Vik, P. Halvorsen, and C. Griwodz, "Evaluating steiner tree heuristics and diameter variations for application layer multicast," *Elsevier Computer Networks*, vol. 52, no. 15, pp. 2872 – 2893, 2008, special issue on Complex Computer and Communication Networks, Elsevier.