5-2009

Building Alternate Multicasting Trees in MPLS Networks

Limin Tang, *et al.*

Building Alternate Multicasting Trees in MPLS Networks

Technical Report UTD/EE/3/2009

May 2009

Limin Tang, Shreejith Billenahalli, Wanjun Huang, Miguel Razo,
Arularasi Sivasankaran, Hars Vardhan, Marco Tacca, and Andrea Fumagalli
Open Networking Advanced Research (OpNeAR) Lab
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas, Richardson, TX, USA
andreaf@utdallas.edu
Paolo Monti
Next Generation Optical Network (NeGONet) Group
School of Information and Communication Technology, ICT-FMI
The Royal Institute of Technology, Kista, Sweden
pmonti@kth.se

Abstract

An algorithm for computing alternate multicast trees in packet transport networks is proposed in this paper. The algorithm efficiently computes multiple sub-optimal tree candidates for a given multicast service request. The algorithm builds on the widely used computation of K ordered loopless shortest paths and can be applied to any connected network topology. Simulation experiments obtained for a multiprotocol label switching (MPLS) network are presented to evaluate the effectiveness and performance of the algorithm.

Keywords: MPLS network, multicast, alternate tree

the network equipment required in the network to support a given set of multicast LSPs. Section 4. contains some observations about future work on the subject.

2. Algorithm

This section contains the description of the algorithm to compute multiple alternate trees for a given multicast LSP request. The algorithm first computes multiple ranking loopless shortest paths from source to each of destination in the multicast tree. Multiple alternate trees are then computed based on these paths. The following notations are used:

- N: number of vertices in the network;
- M: number of edges in the network;
- s: source of the request;
- D: set of destinations of the request;
- d_i: the i^{th} destination of the request;
- n: number of destinations;
- P: set of ordered loopless paths from s to all $d_i \in D$;
- P_i: set of ordered loopless paths from s to d_i;
- p_{ij}: the jth shortest path from s to d_i;
- T: set of alternate multicast trees;
- K: minimum number of alternate trees to be computed;
- m: number of maximum hops from s to all $d_i \in D$; if no such constraint, $m = \infty$.

The algorithm is split into two steps or procedures.

Procedure 1 creates set P_i, i.e., it computes a set of K loopless shortest paths for every pair (s,d_i), $d_i \in D$.

Then, move to the next destination node $d_{i'}$ The pseudocode for Procedure 2 is given next.

Procedure 2:

$k \leftarrow 0$

$T \leftarrow \emptyset$

For ($P_i \in P$)

  For ($p_{ij} \in P_i$)

    create a graph $G(V, E)$, $V \leftarrow \emptyset$, $E \leftarrow \emptyset$

    For vertex $v \in p_{ij}$

      $V = V \cup v$

    EndFor

    For edge $e \in p_{ij}$

      $E = E \cup e$

    EndFor

    For $d_{i'} \in D$ and $i' \neq i$

      $v \leftarrow d_{i'}$

      $e \leftarrow$last edge of $p_{i'1}$

      While ($v \notin V$)

        $V = V \cup v$

        $E = E \cup e$

        $v = v$'s upstream vertex on $p_{i'1}$

```
      Break
    EndIf
  EndFor
```

2.1 Algorithm Complexity

This section evaluates the complexity of the proposed algorithm.

Procedure 1 computes K shortest paths from source to all destinations of the multicast traffic. Since computing K shortest paths for a pair of vertices has complexity $O(KN(M + N \log N))$ [6], the complexity of procedure 1 is $O(nKN(M + N \log N))$.

Procedure 2 has, at most, K iterations. Each iteration has three steps:

1. add vertices and edges of the j^{th} shortest path from s to d_i to the tree;
2. add vertices and edges of the shortest path from s to $d_{i'} (i' \neq i)$ to the tree;
3. if $m < \infty$, count number of hops from s to each d_i in t.

Since a shortest path can have at most M edges and at most $M + 1$ vertices, step 1 has complexity $O(M)$; similarly step 2 has complexity $O((n-1)M)$; complexity of step 3 is $O(nM)$ since number of hops between any pair of vertices is at most M; so the complexity of procedure 2 is $K(O((n-1)M) + O(M) + O(n)) = O(KnM)$.

Hence, the maximum complexity of the algorithm is $O(nKN(M+N \log N)) + O(KnM)$ $= O(nK(N+1)M+nKN \log N)) = O(nKN(M+N \log N))$, which means the proposed algorithm complexity is comparable to the complexity of computing K shortest paths from a given source node to each destination, i.e., procedure 1's complexity.

$p_{1j'}$, there are at least two edges $e(v_1, v_2) \in p_{1j}$ and $e'(v_1', v_2') \in p_{1j'}$ in the network that satisfy $v_1 \neq v_1'$ and $v_2 = v_2'$. Obviously, $e \notin p_{1j'}$ and $e' \notin p_{1j}$; also, e and e' cannot both be in SPT, otherwise we will have two shortest paths from s to v_2, which is not possible; so either e or e' is not in SPT. Without loss of generality, we assume e is not in SPT, then e cannot appear in any path of p_{i1}, combined with $e \notin p_{1j'}$, we have $e \notin T_{j'}$. We know $e \in p_{1j}$ hence $e \in T_j$, so T_j and T_j' must be two different trees.

1. and 2. prove that a tree built based on p_{1j} is distinct from a tree built based on $p_{1j'}$ when $j \neq j'$. Since $j \in \{1, \ldots, K\}$, at least K distinct trees can be built by the algorithm.

The proof above is based on the assumption that the multicast request has no constraint on the number of hops from source to any destination. However, if the request has such constraint, then the algorithm will not guarantee that K distinct alternate trees can be found. This is quite obvious since if such constraint exists, even shortest path between source and a destination may not be found; in an extreme case when $m = 1$, no multicast tree can be built unless all destinations are 1 hop away from the source, which is highly unlikely in reality.

3. Experiments

We designed two experiments to examine the effectiveness of the algorithm. In Experiment I, we mainly concerned the value of K's effect on the whole network optimization; in Experiment II, performance of the algorithm is evaluated when LSPs have constraint which does not allow hop count from source to destination to exceed certain number.

is large, improvement of one more alternate tree for optimization is relatively minimal and can even cause a reverse effect in some cases.

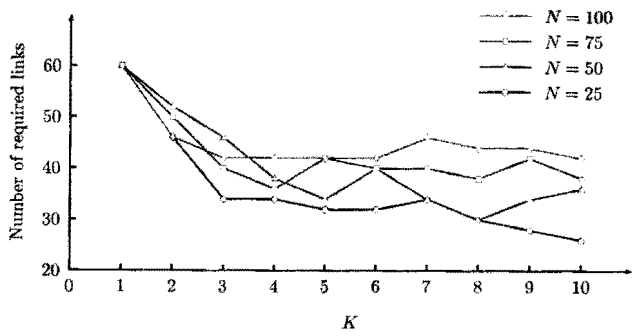

Figure 1

Effect of K on number of used edges for multiple multicast requests. The network has 10 vertices and 60 unidirectional edges, N is number of multicast requests.

3.2 Experiment II

Experiment II is similar to Experiment I, except that each request will have an extra constraint: hop count from source to each of the destinations cannot exceed certain value (m). Three cases are taken from Experiment I, which are network 1($|V| = 10, |E| = 60$) with 25 multicast requests, network 2($|V| = 20, |E| = 120$) with 50 multicast requests and network 3($|V| = 50, |E| = 300$) with 75 multicast requests. Results of optimization are shown in Fig. 4, 5 and 6 under different maximum hop count constraints. From Experiment I we can see that $K = 5$ usually provides fairly good optimization, so we set $K = 5$

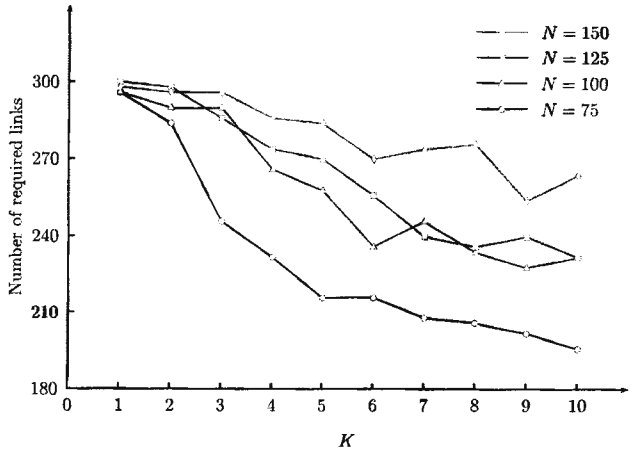

Figure 3

Effect of K on number of used edges for multiple multicast requests, the network has 50 vertices and 300 unidirectional edges, N is number of multicast requests.

and 160 unidirectional edges and network 3 has 50 vertices and 400 unidirectional edges. For each edge in the network, it has maximum transmission capacity C. A number of multicast requests, with average number of destinations ranging from 2 to 8 and bandwidth request $C/100$, are randomly generated for each network.

Simulated Annealing (SA) algorithm is used in the experiment to find the optimal usage of bandwidth, which is to minimize the number of required links in the network. SA keeps looking for a better solution during a certain amount of time by creating an alternate solution each time, which is generated by letting each request randomly choose one from all available trees for multicast. The purpose of these experiments is to find out degree of optimization of the whole network when alternate trees are available for multicast LSPs.

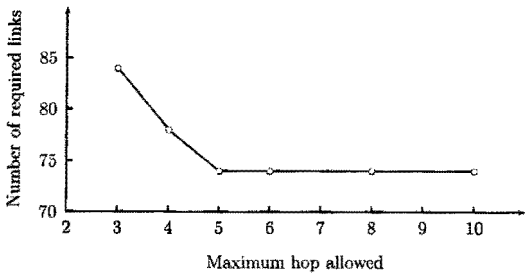

Figure 5

Effect of maximum hop count constraint on number of used edges for multiple multicast requests. The network has 20 vertices and 120 unidirectional edges, number of multicast requests is 50 and $K = 5$.

4. Conclusion

This paper presents an algorithm for computing alternate multicast tree candidates in a connected network topology. Both the algorithm and its complexity are based on the computation of K shortest paths — a widely used algorithm in networking. The purpose of computing multiple tree candidates for every individual multicast service request is to provide the network control plane with multiple options to choose from for every service, while optimizing some global cost function, e.g., bandwidth utilization, percentage of blocking. The effectiveness of the algorithm in computing alternate tree candidates was tested using a MPLS network example.

The considered computation of both the K shortest paths and alternate tree candidates accounts for the maximum hop count constraint only. However, multicast in MPLS networks may be subject to other types of constraint, e.g., limited multicast functionality at

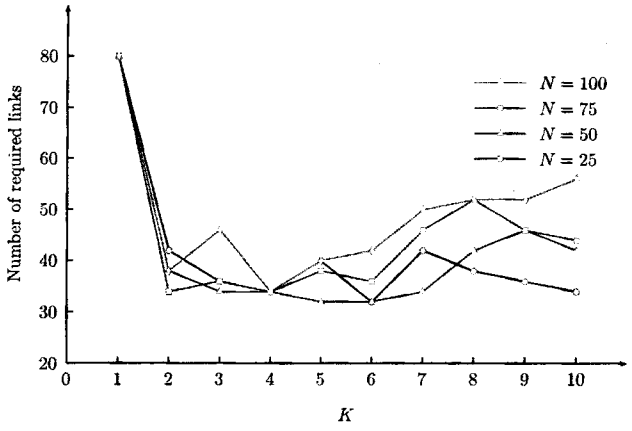

Figure 7

Effect of K on number of used edges for multiple multicast requests. The network has 10 vertices and 80 unidirectional edges, N is number of multicast requests.

[6] E. Q. Martins and M. M. Pascoal, "A new implementation of yen's ranking loopless paths algorithm," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, p. 121, 2003.

[7] J. Hershberger, M. Maxel, and S. Suri, "Finding the k shortest simple paths: A new algorithm and its implementation," *ACM Trans. Algorithms*, vol. 3, no. 4, p. 45, 2007.

[8] W. Wei and A. Zakhor, "Multiple tree video multicast over wireless ad hoc networks," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 1, pp. 2–15, Jan. 2007.

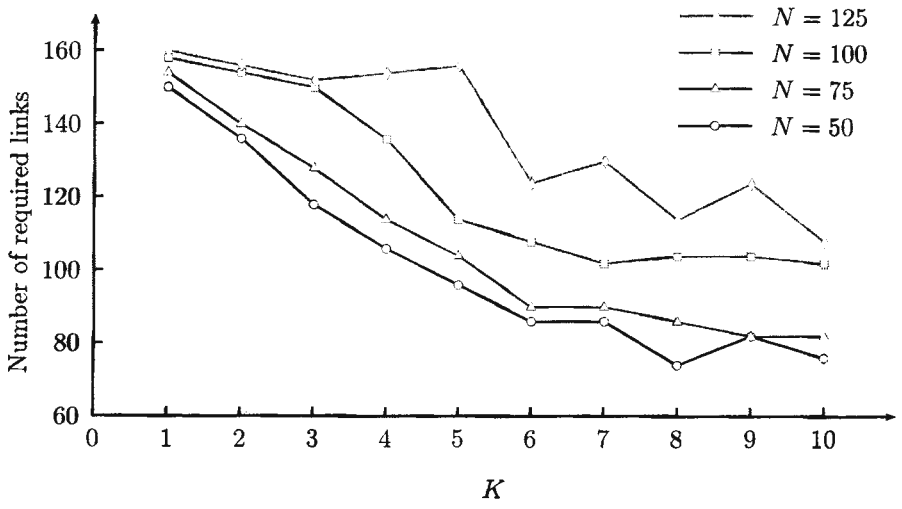

Figure 9

Effect of K on number of used edges for multiple multicast requests, the network has 50 vertices and 400 unidirectional edges, N is number of multicast requests.