Efficient Approaches for Finding QoS Constrained Paths with Applications for Multicast Routing

Tzay-Farn Shih[1] and Chao-Cheng Shih[2]

[1]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 106.
shih@cobra.ee.ntu.edu.tw

[2]Department of Information Management, Shih Hsin University, Taipei, Taiwan 116.
ccshih@cc.shu.edu.tw

Abstract

Most of the proposed multicast routing algorithms require each node to maintain a global network state that is an approximation of the current network state due to the delay of propagating local state. As the network size grows, the imprecision of global state, storage and communication overhead will increase and result in poor scalability. In this paper, we proposed a scalable and loop-free distributed multicast routing protocol, which requires every node to maintain only its local state. It uses a distributed computation to collectively utilize the most up-to-date local state information to find multicast tree in a hop-by-hop basis. The performance of our algorithm was studied by simulation, and the results show that the performance is excellent.

Index Terms—Steiner tree, Source routing, Distributed algorithm, Multicasting

1. Introduction

Many applications of computer network such as videoconferencing, remote collaboration and distant education will involve multiple users that will rely on the ability of the network to provide multicast services. Thus, multicasting will likely be an essential part of future networks. In multicasting, messages are concurrently sent to multiple destinations. Routing is one of the issues for multicasting. The typical approaches to multicast routing require the transmission of packets along the branches of a tree spanning the source and destination nodes. The problem of computing multicast trees has received considerable attentions, and several algorithms have been proposed based on a number of optimization goals. One frequently considered optimization objective is to minimize the known as the Steiner tree, and finding such a tree is a well-known NP-complete problem [1, 2].

According to the optimization objective, multicast routing algorithms can be classified into one of two categories [3]. The first category is the minimum Steiner tree algorithms which minimize the total cost of the multicast tree. There are some minimum Steiner trees heuristics have been proposed in [4-7]. The other category is the shortest path tree algorithms. Their objective is to minimize the cost of each path from the source node to a multicast group member node. Bellman-Ford's algorithm and Dijkstra's algorithm are two well-known shortest path algorithms.

In this paper, we proposed a scalable and loop-free distributed multicast routing protocol that requires every node to maintain only its local state. It uses a distributed computation to collectively utilize the most up-to-date local state information to find multicast tree in a hop-by-hop basis. The performance of our algorithm was studied through extensive simulation. The simulation results reveal that its performance is better than other algorithms.

The rest of the paper is organized as follows. Section 2 details the motivation of our protocol. Our protocol is described in Section 3. Then, Section 4 presents the simulation results. Finally, Section 5 concludes this paper.

2. Motivation

In this section, we detail the problems of the distributed QoS routing algorithm (DRA) proposed by Shigang Chen and Klara Nahrstedt in [8-10].

In Shigang Chen and Klara Nahrstedt's algorithm, in order to make their algorithm work well for additive metric, the non-zero probing waiting-time Δt must be specified for each DRA that involves additive metric. If $\Delta t = 0$, the algorithm might fail to find a feasible path. Considering the topology depicted in Figure 1 for an explanation, where the number on each link is cost, *s* is source, and *t* is destination. Assume that the cost constrain of route is 6, there exists a path P = $s \rightarrow i \rightarrow k \rightarrow l \rightarrow t$ that satisfies the connection cost constrain. For simplicity, PROBE n($s \rightarrow i \rightarrow k$) denotes that PROBE n flows along the path $s \rightarrow i \rightarrow k$. If PROBE 2($s \rightarrow j \rightarrow k$) arrives at node *k* earlier than PROBE 1($s \rightarrow i \rightarrow k$), PROBE 1 will be dropped by node *k* and PROBE 2 will be dropped by node *m*. Hence, there is no PROBE can reach destination node *t* and fails to find a path. The problem mentioned above occurs while the DRA adopting additive metric. Hence, the non-zero probe waiting-time Δt must be specified for DRA variations that involves additive metric. In order to find the path P = *s* → *i* → *k* → *l* → *t*, PROBE 1 must to arrive node *k* earlier than PROBE 2. To this end, they let Δt = node-delay(*i*, *j*), where node-delay(*i*, *j*) is the protocol-processing time and the queuing delay at node *i* for link (*i*, *j*). But we found that this approach does not guarantee to prevent the above problem. Assume that node-delay(*s*, *i*) < node-delay(*s*, *j*) and link-delay(*s*, *i*) + link-delay(*i*, *k*) >> link-delay(*s*, *j*) + link-delay(*j*, *k*), then PROBE 1 will be send earlier than PROBE 2 at node *s* but arrive node *k* later than PROBE 2. It also fails to find a path. It increases the connection set up time but cannot guarantee to obtain an optimal solution. In next section we propose two algorithms to overcome the above problems.

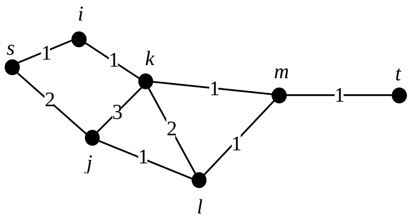

Figure 1. Network topology.

3. The Proposed Protocol

In this section, we propose two algorithms to solve these problems of DRA that we mentioned in Section 2. Our algorithms can be applied to solve both unicast routing and multicast routing problem.

The problems mentioned in the previous section can be solved by setting Δt = delay(*i*, *j*) = node-delay(*i*, *j*) + link-delay(*i*, *j*), where link-delay (*i*, *j*) is the propagation delay on link (*i*, *j*). It guarantees that PROBE packet flow along the least delay path will arriving destination earliest.

Although this approach guarantees to find a path if there exist a path, but the connection setup time is longer than the original version. To reduce the setup time, we propose the following replacement rule:

Best Predecessor Replacement: When a node receives a PROBE packet, it will compare the accumulated metric (e.g., accumulated delay, cost) of this PROBE with the previous PROBEs'. If the accumulated metric of the new PROBE is better than the former, the node changes its predecessor to the node where the PROBE be relayed and forwards this PROBE if no PROBE has been forwarded.

Since every node selected the best predecessor, the path find by this algorithm is better. Assume that the number on each edge represents the cost of link and PROBE 1(*s* → *j* → *k*) arrived at node *k* earlier than PROBE 2(*s* → *i* → *k*) in Figure 1. When PROBE 1 arrives at node *k*, *k* sets its predecessor as *j* and records its accumulated cost 5. After PROBE 2 arrives at node *k*, it compares PROBE 2's accumulated cost with PROBE 1's. Because PROBE 2's accumulated cost 2 is less than PROBE 1's, it changes the predecessor of *k* to *i* and updates *k*'s accumulated cost to 2. By this replacement strategy, the path *s* → *i* → *k* is used to replace the older path *s* → *j* → *k*. The PROBE is contended hop-by-hop to find a better path.

Algorithm I (Shih I): First, let Δt = node-delay(*i*, *j*) + link-delay(*i*, *j*). Then, do the **Best Predecessor Replacement**.

The algorithm running at each node c is as follows:

```
Suspend until receives a control packet CP from node p;
switch (packet type) {
case PROBE:
  if (c is not the destination node and the accumulated metric of
      CP is better than previous PROBE packets' accumulated
      metric) {
      let node p to be the predecessor of node c;
      update accumulated metric;
      for (every neighbor node j of c except node p)
          if(accumulated metric satisfy the QoS constraint )
            send a PROBE packet to c's neighbor j after Δt;}
  else if (c is the destination node and the accumulated metric of
      CP is better than previous PROBE packets' accumulated
      metric) {
          send a BACKWARD packet to c's predecessor p;
  }
  else    discard PROBE packet;
case BACKWARD:
  if (node c has enough resource for this connection) {
     reserve demanded resources;
     let node n to be the successor of node c;
     if (c is source node)
         connection setup success;
     else    send a BACKWARD packet to c's predecessor p;}
  else    send a FAILURE packet to c's successor n;
case FAILURE:
  release reserved sources for this connection;
  if (c is not the destination node)
       send a FAILURE packet to c's successor n;
} // end of switch
```

In Shih I, every node can relay PROBE packet only once that might be failed to find the optimal path sometimes. In Figure 1, we assume PROBE 1(*s* → *j* → *k* → *m*) arrived node *m* earlier than PROBE 2(*s* → *j* → *l* → *m*), and PROBE 1 arrived node *k* earlier than PROBE 3(*s* → *i* → *k*). When PROBE 1 arrived at node *m*, the predecessor and the accumulated cost of *m* were set to *k* and 6 respectively. While PROBE 2 arrived at *m*, *m* changes its predecessor to *l* because PROBE 2 has a lower accumulated cost 4. Node *k* set its predecessor to *j* and its accumulated cost to 5 while PROBE 1 arrived. PROBE 3's accumulated cost 2 is less than PROBE 1 carried. After PROBE 3 arrived at node *k*,

k's predecessor is changed to i. Because k has forwarded a PROBE, PROBE 3 can't be forward to m. Thus, node m can not get this information to update its predecessor to k. The destination node t will send an ACK packet that will flow along the path $t \rightarrow m \rightarrow l \rightarrow j \rightarrow s$ to source node s. Instead of choosing a good path $s \rightarrow i \rightarrow k \rightarrow m \rightarrow t$, a non-optimal path $s \rightarrow j \rightarrow l \rightarrow m \rightarrow t$ was selected.

In order to find an optimal path, we propose the following forward rule and Algorithm:

Best Forward Rule: When a node receives a PROBE packet, it will compare the accumulated metric (e.g., accumulated delay, cost) of this PROBE with the previous PROBEs'. If a node receives a PROBE packet with a better accumulated metric, it will do the **Best Predecessor Replacement** and forward this packet to its neighbors.

Algorithm II (Shih II): First, let Δt = node-delay(i, j) + link-delay(i, j). Then, do the **Best Forward Rule**.

Shih II can solve the problem discussed as above. When PROBE 3 arrived at k, k changes its predecessor to i and relays this PROBE to m. Because m get a packet that has smaller accumulated cost than PROBE 2's, it sets its predecessor to k. The ACK packet is replied along the path $t \rightarrow m \rightarrow k \rightarrow i \rightarrow s$. Thus, an optimal path $s \rightarrow i \rightarrow k \rightarrow m \rightarrow t$ will be established. The algorithm can be obtained by modify the PROBE phase of Shih I as follows:

```
Suspend until receives a control packet CP from node p;
switch (packet type) {
case PROBE:
   if (c is not the destination node and the accumulated metric of
        CP is better than previous PROBE packets' accumulated
        metric) {
         let node p to be the predecessor of node c;
         update accumulated metric;
         for (every neighbor node j of c except node p)
              if(accumulated metric satisfy the QoS constraint )
                 send a PROBE packet to c's neighbor j after Δt;}
   else if (c is the destination node and the accumulated metric of
        CP is better than previous PROBE packets' accumulated
        metric) {
              send a BACKWARD packet to c's predecessor p;}
   else   discard PROBE packet;
case BACKWARD:
      similar to Shih I
case FAILURE:
      similar to Shih I
} // end of switch
```

Theorem 1 If the path of a connection is existed, it must be loop-free.

Proof: If the path of the connection has a loop, there must be a node k on the path received and forwarded the same PROBE packet twice. If a PROBE packet passed node k twice, the accumulated metric will greater or equal to the previous PROBE's and it will be discard. That contradicts to the above assumption. #

Theorem 2 The path P = $s \rightarrow n_1 \rightarrow n_2 \rightarrow \cdots \rightarrow n_k \rightarrow t$ established by the algorithm Shih II is optimal.

Proof: If the path P = $s \rightarrow n_1 \rightarrow n_2 \rightarrow \cdots \rightarrow n_k \rightarrow t$ is not optimal, there must exist an optimal path P′ = $s \rightarrow n_1 \rightarrow n_2 \rightarrow \cdots \rightarrow n'_k \rightarrow t$ such that $Am(s \rightarrow n'_k) + Am(n'_k \rightarrow t)$ is minimal, where $Am(x \rightarrow y)$ represents the accumulated metric from node x to node y. According to the algorithm Shih II, node n'_k will be select to as the new predecessor of node t. That contradicts to the fact that t's predecessor is node n_k. #

4. Simulation Model and Results

We study the average delay, connection setup time, average probe overhead for different network size and group size by simulation. The simulator was developed in C++ language.

The network size was ranged from 20, 30, 40, 50 and 60 nodes that were generated randomly. In each size, it generates different group size in the range of 5, 10, 15, and 20 nodes and adds source node randomly. The simulator allows the use of two types of traffic sources: voice source and video source. An on-off model is used for voice sources, more detailed description of the model can be found in [11]. A 10-state model is used for video source that is described in [12]. Both of them are variable bit rate sources suitable for multimedia applications. Background traffic on each link is modeled either as multiplexed video sources traversing that link or as a single aggregate source, called a batch source.

Figure 2 shows the results of running Dra, Shih I, and Shih II on a randomly generated network, network size is 20, for different group size. The average delay for there algorithms are increased with the number of group size. It matches the condition, the more connection established, the network is busier and the delay is increased. The average delay of the multicast tree established by Shih II is the least. The average transmission delay of Dra is greater than Shih I and Shih II. The simulation results reveal that the path found by Shih II is better than Shih I or Dra.

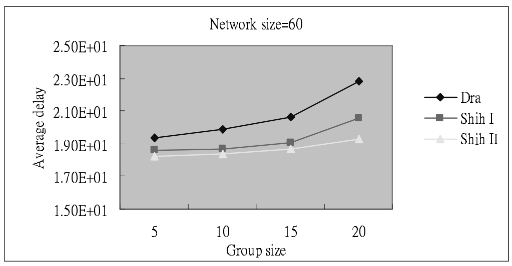

Figure 2. Average delay.

The connection setup time of transmission request for Dra, Shih I, and Shih II are presented in Figure 3.

The network is randomly generated and the group size is set to 20. In this figure, we can find that the connection setup time is growing with the network size for each algorithm. While the network nodes increased, the route search space is increased and induced a longer setup time. Take an insight into the result, we found that the connection setup time of Shih II for the same connection request is smallest. It means that Shih II construct a path faster than others. This is a very important requirement in high-speed network routing for real time applications.

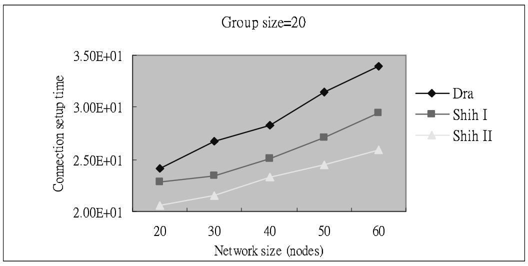

Figure 3. Connection setup time.

A node might forward probe packet several times in Shih II while a best predecessor replacement is occurred. The re-probing phase will forward extra probe packets. Although the cost of faster connection setup is the extra message overhead, it is never greater than the double of Dra's in our simulation.

5. Conclusion

In this paper, we proposed two loop-free distributed multicast routing algorithm, Shih I and Shih II. In our algorithms, each node requires to maintain only its local state that saves the storage and communication overhead significantly. Our protocols use a distributed computation to collectively utilize the most up-to-date local state information to find multicast tree in a hop-by-hop basis. In Shih I, each node can forward probe packet only once so the probe message is lower than Shih II. Shih I sometimes cannot guarantee to find an optimal path by the restriction mentioned above. In order to construct an optimal path, we revise Shih I to Shih II that guarantee to find an optimal path if it exists. The performance of our algorithm was studied through extensive simulation. The simulation results reveal that the performance of our protocols is better than DRA.

REFERENCES

[1] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations* (R. Miller and J. Thatcher, eds.), pp. 85-103, Plenum Press, 1972.

[2] P. Winter, "Steiner Problem in networks: A Survey," *Networks*, Vol. 17, no. 2, pp. 129-167, 1987.

[3] H. F. Salama, D. S. Reeves and Y. Viniotis, "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-speed Networks," *IEEE JSAC*, pp. 332-345, April 1997.

[4] S. Ramanathan, "An Algorithm for Multicast Tree Generation in net- works with Asymmetric Links," in *Proceedings of IEEE INFOCOM*, pp. 337-344, 1996.

[5] L. Kou, G. Markowsky, and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, vol. 15, no. 2, pp. 141-145, 1981.

[6] V. Rayward-Smith, "The Computation of Nearly Minimal Steiner Trees in Graphs," *International Journal of mathematical Education in Science and Technology*, vol. 14, no. 1, pp. 15-23, January/February 1983.

[7] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Mathematical Japonica*, vol. 24, no. 6, pp. 573-577, February 1980.

[8] Shigang Chen and Klara Nahrstedt, Distributed QoS Routing, Technical Report No. UIUCDCS-R-97-2017, Department of Computer Science, University of Illinois at Urbana-Champaign, July 1997.

[9] Shigang Chen, Klara Nahrstedt, Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing, in *Proceedings of Annual Conference on Local Area Networks* (LCN'98), Boston, MA, October 1998, pp. 80-89.

[10] Shigang Chen, Klara Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network Magazine*, Special Issue on Transmission and Distribution of Digital Video, Vol. 12, No. 6, November-December 1998, pp. 64-79.

[11] Guerin et. al., "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks," *IEEE JSAC*, Vol. 9, no. 7, September 1991.

[12] Maglaris et. al., "Performance Models of Statistical Multiplexing in Packet Video Communications," *IEEE Trans. on Comm.*, Vol. 36, no. 7, July 1988.

[1] T. F. Shih is with the Department of Computer Science and Information Engineering, China College of Marine Technology and Commerce, Taipei, Taiwan 212.