A Distributed Dynamic Delay-Constrained Least-Cost Multicast Routing Heuristic*

WANG Zheng-ying, SHI Bing-xin, LIU Wei

(Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

E-mail: wangzhengying@263.net

http://wangzhy_appl.edu.chinaren.com

Received May 22, 2000; accepted September 15, 2000

Abstract: Many new distributed multimedia applications involve dynamic multiple participants, have stringent end-to-end delay requirement and consume large amount of network resources. In this paper, a new DDDCLCMR (distributed dynamic delay-constrained least-cost multicast routing algorithm) is proposed to support these applications. DDDCLCMR scales well because the source of the multicast tree needs only limited computation or may even not be involved in the route computation. When group membership changes, the existing multicast tree is perturbed as little as possible. Simulation results show that DDDCLCMR performs very well in terms of delay and cost for both static and dynamic multicast groups, compared with the best multicast algorithms known.

Key words: QoS; multicast; dynamic routing; distributed algorithm; heuristic

Many new distributed real-time applications of computer networks such as distance education and videoconferencing involve multiple participants, have stringent end-to-end delay requirement and consume a large amount of network resources. In order to support these new applications efficiently, multicast routing algorithms that compute least cost multicast trees under a given end-to-end delay constraint are desirable. Because the membership of the multicast group changes frequently in some applications (e.g. teleconferencing), the routing algorithms should also have the ability to alter an existing multicast tree to accommodate membership changes. Although a number of delay-constrained multicast routing algorithms[1~3] have been proposed in the past few years, but they are all designed for static multicast groups. References[4~7] have proposed some dynamic multicast routing heuristics, but they only consider one link metric (link cost or link delay). Some existing multicast routing protocols used in the

* This project is supported by the National 'Ninth-Five' Sci-Tech Key Project of China under Grant No. 96-743-01-04-02 ("九五"国家重点科技攻关项目). **WANG Zheng-ying** was born in 1974. He is a Ph.D. student at the Department of Electronics and Information Engineering of Huazhong University of Science and Technology. He received his B.E. and M.E. degrees from Harbin Institute of Technology in 1996 and 1998, respectively. His research interests include high-speed computer network, protocol engineering, and intelligent computer network. **SHI Bing-xin** was born in 1936. He is a professor and doctoral supervisor of the Department of Electronics and Information Engineering of Huazhong University of Science and Technology. His current research interests include computer network management, network simulation and performances analysis, mobile IP and network performances monitor. **LIU Wei** was born in 1976. He is a master student at the Department of Electronics and Information Engineering of Huazhong University of Science and Technology. He received his B.E. degree from the same department of Huazhong University of Science and Technology in 1998. His research interests include intelligent computer network management and wireless communication.

Internet (MBone) can very well support dynamic multicast groups. However, the underlying multicast routing algorithms have been designed only for best-effort delivery.

Though E. Biersack and J. Nonnenmacher[8] have presented a dynamic algorithm (WAVE) for the multi-constrained QoS routing problem, there are some drawbacks with this algorithm. (1) The number of responses could be very large; (2) The source node is always contacted whenever a new node is added, such a solution would not scale well; and (3) WAVE does not consider explicitly a given delay constraint.

In this paper, we extend the distributed unicast routing algorithm DCLC-DSF proposed by us earlier[9], and propose a distributed dynamic heuristic (DDDCLCMR) for the delay-constrained least-cost multicast routing problem. DDDCLCMR scales well because the source of the multicast tree needs only limited computation or may even not be involved in the route computation. When group membership changes, the existing multicast tree is perturbed as little as possible.

1 Problem Formulation

A network can be modeled as a graph $N(V,E)$, where V is the set of all nodes, representing routers or switches, E is the set of edges representing physical or logical connectivity between nodes. Each link is bi-directional. Let $s\in V$ be the multicast source, $M\subset V-\{s\}$ the multicast destinations, and R_+ the set of positive real numbers. We define two additive functions to each link $e\in E$: the delay function $\text{delay}(e):E\to R_+$ and the cost function $\text{cost}(e):E\to R_+$. Let t be any destination node of M, $p(s,t)$ the path from s to t. For a given source node $s\in V$ and a destination node set M, there exist the following relationships for the multicast tree constructed by s and M.

1) $\text{delay}(p(s,t))=\sum_{e\in p(s,t)}\text{delay}(e)$

2) $\text{cost}(T)=\sum_{e\in T}\text{cost}(e)$

Definition 1. Delay-constrained multicast tree $T(s,M)$: A delay-constrained multicast tree $T(s,M)$ is the multicast tree which is constituted by source s and destination set M, and satisfies the delay constraint, i.e. $\{\text{delay}(p(s,t))\leqslant D_t,t\in M\}$, where D_t is the delay constraint of node t.

Definition 2. Delay-constrained least-cost multicast routing problem: Given a network $N(V,E)$, a source node $s\in V$, a destination node set $M\subseteq V-\{s\}$, a positive link delay function $\text{delay}(\cdot)\in R_+$ and a positive link cost function $\text{cost}(\cdot)\in R_+$, the delay-constrained least-cost multicast routing problem is to find a multicast tree that satisfies:

$$\min\{\text{cost}(T(s,M)),T(s,M)\in T_f(s,M)\}$$

where $T_f(s,M)$ is the set of all delay-constrained multicast trees constituted by s and M.

It has been demonstrated that the delay-constrained least-cost multicast routing problem is NP-complete[10].

2 The Proposed Algorithm DDDCLCMR

2.1 Routing information

We first discuss the routing information, which needs to be available for the proposed algorithm to do the route computation.

Each node knows the delay and cost of all its outgoing links. Each node has a unicast routing table that has a delay vector and a cost vector. The delay vector consists of $|V|$ entries, one entry for each node in the network. Each entry gives the delay and the next hop for a node in the network. Likewise, the cost vector gives the cost related information. The information provided by each vector is computed and maintained by an existing unicast rout-

ing protocol. Either a link-state routing algorithm or a distance-vector routing algorithm can compute the above vectors. This issue is beyond the scope of this paper and we will not discuss these procedures in this paper.

Furthermore, each node also has a multicast routing table that contains the routing information about each multicast tree.

2.2 DCLC-DSF algorithm

DCLC-DSF algorithm is a distributed delay-constrained least-cost unicast routing algorithm recently proposed by us. Here, we only give a simple description of DCLC-DSF. The reader can refer to Ref. [9] for the detailed discussion.

In DCLC-DSF, we define two functions, link judgement function, $\text{judge}(\cdot): E \to R_+$, and node selection function, $\text{select}(\cdot): V \to E$. The link judgement function is used to calculate the judgement value under a special rule. The node selection function is used to select the outgoing link according to the minimum judgement value. From the source node, each selected node first computes the two functions, and then selects a feasible outgoing link by the result of the node selection function. This process is repeated until the destination node is selected.

The link judgement function is crucial. It decides the performance of DCLC-DSF. We tried hundreds of link judgement functions during our study. The following one is a good example.

Without losing generality, we discuss any node v, any neighbor node $w \in A_v$, where A_v is the set of all neighbor nodes. In the following, we set $p_{d\min}(s,v)$ to be the least delay path from s to v, $D_{v,w} = D - (\text{delay}(p_{dsf}(s,v)) + \text{delay}(v,w))$, where s is the source, D the delay constraint, and $p_{dsf}(s,v)$ the path from s to v obtained by DCLC-DSF.

1. If $t \notin A_v$

$$\text{judge}(v,w) = \begin{cases} \dfrac{\text{cost}(v,w)}{D_{v,w}} & \text{delay}(p_{d\min}(w,t)) \leqslant D_{v,w} \wedge D_{v,w} > 0 \wedge w \notin p_{dsf}(s,v) \\ +\infty & \text{otherwise} \end{cases}$$

$$\text{select}(v) = (v,w) \quad \text{While judge}(v,w) = \min\{\text{judge}(v,i), i \in A_v\}$$

2. If $t \in A_v$

If $D_{v,t} \geqslant 0$ and $\{\text{cost}(v,t) \leqslant \text{cost}(v,w) + \text{cost}(p_{d\min}(w,t)), w \in A_v \wedge w \neq t\}$, OR $A_v = \{t\}$

Then $\text{select}(v) = (v,t)$

Else

$$\text{judge}(v,w) = \begin{cases} \dfrac{\text{cost}(v,w)}{D_{v,w}} & \text{delay}(p_{d\min}(w,t)) \leqslant D_{v,w} \wedge D_{v,w} > 0 \wedge w \notin p_{dsf}(s,v) \wedge w \neq t \\ +\infty & \text{otherwise} \end{cases}$$

$$\text{select}(v) = (v,w) \quad \text{While judge}(v,w) = \min\{\text{judge}(v,i), i \in A_v\}$$

DCLC-DSF is a simple, accurate and robust heuristic. The average path cost inefficiency is always within 8% from CBF[9], which is an optimal algorithm for the delay-constrained least-cost unicast routing problem. By the analysis of DCLC-DSF, we also have:

Theorem 1. A path constructed by DCLC-DSF from source s to destination t does not contain loops.

Theorem 2. DCLC-DSF always terminates in a finite time.

Theorem 3. DCLC-DSF will construct a delay-constrained path from source s to destination t if and only if such a path exists.

2.3 DDDCLCMR algorithm

DDDCLCMR is based on DCLC-DSF. It first selects the source as an initial multicast sub-tree, then adds/removes a destination node to form a new multicast sub-tree by the corresponding add/remove request. This destination node adding/removing operation exists during the whole multicast session. The process of establishing a mul-

ticast tree in DDDCLCMR is the process that dynamically adds and removes the destination nodes.

2.3.1 Destination node adding operation

Without losing generality, we discuss any destination node t. Let T_s denote the multicast sub-tree while adding t (T_s is composed of the initial multicast sub-tree, the added destination nodes and the added paths of the added destinations), $p_T(s,v)$ the path from s to v on T_s, $p_{dsf}(v,t)$ the path from v to t obtained by DCLC-DSF.

In DDDCLCMR, when t joins an existing tree, t contacts node v on the existing tree T_s with an add request $JoinReq(id,t,D_t)$ where id is the unique identifier of this multicast session. In order to reach a good statistical performance, node v is randomly selected on T_s.

The destination node adding operation is described as follows.

Step 1. Set $n \leftarrow v$.

Step 2. If $n=s$ and $\text{delay}(p_{dmin}(s,t)) > D_t$, there exists no delay-constrained path from s to t and this operation terminates (s may negotiate with t to relax the delay bound); otherwise go to the next step.

Step 3. If $\text{delay}(p_T(s,n)) + \text{delay}(p_{dmin}(n,t)) \leqslant D_t$, node n computes a path from itself to t satisfying the delay constraint of $D_t - \text{delay}(p_T(s,n))$ using DCLC-DSF; otherwise go to the next step.

Step 4. n sends the received add request message $JoinReq(id,t,D_t)$ to n's parent node w on T_s (according to Theorem 4, n should not send this request to its child node(s) on T_s). Set $n \leftarrow w$, go to Step 2.

According to Theorem 5, if the network satisfies the delay constraint of t, there exists a node n on T_s satisfying the condition of Step 3. After destination node t is added, a new multicast sub-tree composed of the original multicast sub-trees (T_s) and the added path ($p_{dsf}(n,t)$) is formed. The new multicast sub-tree will be used for the next destination node adding operation. This adding operation repeats until all the destination nodes are added.

It is easy to prove that DDDCLCMR will possibly form loops only when $\text{judge}(v,w) = \text{judge}(v,w') = \min\{\text{judge}(v,k), k \in A_v\}$ for link (v,w) and (v,w') in $N(V,E)$. This is because that when this condition occurs, DCLC-DSF will randomly select an outgoing link from (v,w) and (v,w'). In order to remove loops, many loop-avoiding algorithms can be adopted. Here we only give a simple one: adding additional information T_s (the topology of the sub-tree T_s) to $JoinReq(id,t,\text{DDCF}(t))$ message. When $\text{judge}(v,w) = \text{judge}(v,w') = \min\{\text{judge}(v,k), k \in A_v\}$ for (v,w) and (v,w') exists in DDDCLCMR, DDDCLCMR first looks up T_s, if one of them already has existed in T_s, then selects the outgoing link existing in T_s as v's outgoing link. Otherwise, DDDCLCMR randomly selects one from (v,w) and (v,w') as v's outgoing link. In this way, the loops can be easily avoided.

2.3.2 Destination node removing operation

The same as the discussion about destination node adding operation. We discuss any destination node t.

Step 1. Set $n \leftarrow t$.

Step 2. If n connects to more than two edges in the multicast tree, make the node as the relay node, and this operation completes. Otherwise, go to the next step.

Step 3. n releases the resource reserved for this routing, and sends a remove message $RmReq(id,n)$ to its father node v on the multicast tree T_s.

Step 4. Remove link (v,n). Set $n \leftarrow v$, go to Step 2.

When removing a leaf destination node, the path $p_T(n,t)$ is removed from T_s and a new multicast tree $T'_s = T_s - p_T(n,t)$ is formed, where n is the disconnection node of t on T_s. However, when removing a non-leaf destination node, the multicast topology will not be changed.

2.3.3 Example of destination node adding and removing operation

Figure 1 shows an example of adding and removing destination nodes under the delay constraint $D=25$ to all destinations.

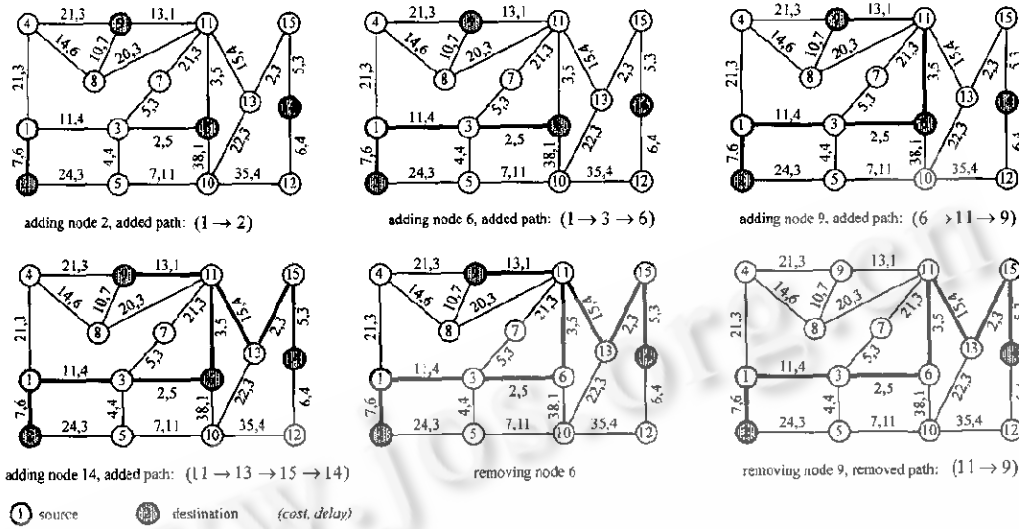


Fig. 1 An example of adding and removing operation with $D=25$ to all the destinations

3 Analysis of DDDCLCMR

3.1 Complexity of DDDCLCMR

For a network $N(V,E)$ with e links and n nodes, we respectively discuss the situation of adding or removing a destination node.

1. Adding a destination node

The computation complexity of adding a destination node at any node is $O(1)$. This is because each time a node receives a JoinReq message, it performs a fixed amount of computations, irrespective of the size of the network.

We now consider the worst case message complexity of adding a destination, i.e. the number of messages needed in the worst case that the link state changes with each probe of DCLC-DSF. According to Ref. [9], the worst case message complexity of DCLC-DSF is $O(e^2)$. For other message it is not more than $O(n)$. So the worst case message complexity of adding a destination is $O(e^2)$. In fact, the network is too unstable to use in practice in the above case. Generally, the period of the routing session establishment is very short; the network state unlikely changes drastically. In this situation, the message complexity of DCLC-DSF is $O(e)$, thus for a stable network the message complexity of adding a destination node in DDDCLCMR is $O(e)$.

2. Removing a destination node

According to the destination node removing operation, it is obvious that the worst case message complexity of removing a destination node is $O(n)$, and the computation complexity of removing a destination node is $O(1)$.

3.2 Correctness of DDDCLCMR

Theorem 4. Let s denote the source of the existing multicast tree T, w any non-leaf node of T, v any one child node of w on T and t the new node to be added. If $\text{delay}(p_T(s,w))+\text{delay}(p_{dmin}(w,t))>D_v$, then $\text{delay}(p_T(s,v))+\text{delay}(p_{dmin}(v,t))>D_v$.

Theorem 5. DDDCLCMR will construct a delay-constrained multicast tree constituted by source s and destination set M if and only if such a multicast tree exists.

The proofs of Theorems 4 and 5 are given in the appendix.

Compared to WAVE, DDDCLCMR has the following merits. (1) The number of responses sent to the new node to be added could be smaller; (2) DDDCLCMR scales better because the source in DDDCLCMR only needs limited computation or may even not be involved in the route computation; (3) DDDCLCMR specifically considers a given delay constraint.

4 Experiments

We have used simulation experiments to evaluate the accuracy of DDDCLCMR, and have run our simulations on the distributed routing simulator, DRS[29], developed by ourselves. We have done the following experiments: (1) routing request success ratio, (2) average message complexity, (3) multicast tree cost inefficiency.

For each run of the experiment we generated a random network with an average node degree of 4. The source, the destination set and the delay requirement (D) of each connection request are randomly generated. D is uniformly distributed in the range of [30ms,160ms]. The cost of each link is uniformly distributed in [0,50ms]. The delay of each link is uniformly distributed in [0,200ms].

For simplicity, we set the delay constraints of all destination nodes equal in our experiments.

4.1 Routing request success ratio

The routing request success ratio θ_{req} is defined as follows:

$$\theta_{req}=N_{ack}/N_{req}$$

where N_{ack} is the number of multicast routing requests accepted, N_{req} is the total number of multicast routing requests.

The least delay multicast tree algorithm (LDT) has the highest routing request success ratio of all delay-constrained multicast routing algorithms because it connects the destination to the source using the least delay path between them. Therefore, we compare DDDCLCMR with LDT in this experiment. Note that we only consider the destination node adding operation and omit the destination removing operation in this experiment.

Figure 2 compares the success ratios of DDDCLCMR and LDT ($N_{req}=100$). The success ratio is a function of the delay requirement. The two algorithms have the same success ratio in satisfying the imposed delay constraint. This is because both of them are always capable of constructing a delay-constrained multicast tree, if one exists. This also proves Theorem 5.

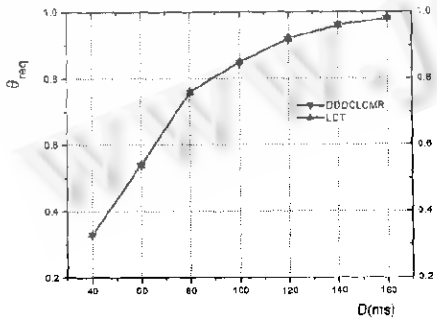


Fig. 2 Comparison of average success ratios of DDDCLCMR and LDT ($N_{req}=100$)

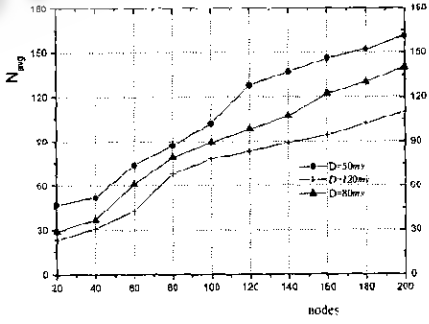


Fig. 3 Average number of messages of DDDCLCMR ($N_{req}=100$)

4.2 Average message complexity

The average message overhead N_{avg} is defined as follows.

$$N_{avg}=N_{msg}/N_{req}$$

where N_{msg} is the total number of messages sent, N_{req} is the total number of connection requests.

Figure 3 shows the average number of messages of 15 destination nodes versus the size of the network with number of nodes varying in [20,200], for three different values of the delay constraint: 50ms, 80ms and 120ms. All the three curves indicate clearly that the average number of messages grows very slowly with the size of the network. This means that DDDCLCMR algorithm can be applied in large-scale networks without causing message flood, which also proves that DDDCLCMR can scale well.

4.3 Multicast tree cost inefficiency

Because the bounded shortest multicast algorithm (BSMA[3]) has the best cost performance among all the proposed delay-constrained static multicast heuristics[11], thus we select BSMA for comparison. Besides we also compare DDDCLCMR with NAIVE[7] which is a known dynamic multicast algorithm that can compute multicast trees satisfying a given delay constraint.

The Multicast Tree Cost Inefficiency $\varphi(T)$ is defined as follows.

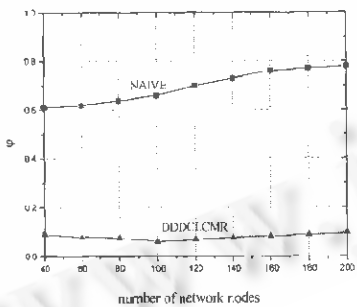
$$\varphi(T)=(\mathrm{cost}(T)-\mathrm{cost}(T_{BSMA}))/\mathrm{cost}(T_{BSMA})$$

where T_{BSMA} is the delay-constrained multicast tree obtained by BSMA. In this simulation, we first consider the situation where the request is only *JoinReq*, next we consider the situation where the request may be *JoinReq* or *RmReq*.

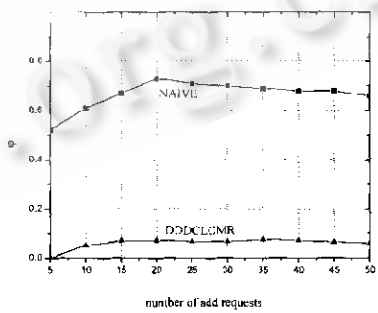
1. Only Add Request

In the simulation, the original request of the destination node was sent to a random node on the existing tree.

This experiment simulates the situation of some applications in which members rarely leave the added group until the end of the application. Figure 4(a) shows the percentage excess cost over BSMA versus number of network nodes for the number of add requests of 20 and the delay bound of 80ms. Figure 4(b) shows the percentage excess cost over BSMA versus number of add requests for the number of network nodes of 100 and the delay bound of 80ms.



(a) Percentage excess cost over BSMA versus number of network nodes for the number of add requests of 20 and delay bound $D=80$ms



(b) Percentage excess cost over BSMA versus number of add requests for the number of network nodes of 100 and delay bound $D=80$ms

Fig. 4

From these figures, we see that the DDDCLCMR has a cost performance that is always within 10% worse than BSMA, while the NAIVE is up to 70% worse than BSMA (which is similar with the simulation result in Ref. [7]). This proves that DDDCLCMR is accurate and scalable.

2. Add or Remove Request

This experiment simulates the situation where the request may be *JoinReq* or *RmReq*. To determine whether the next request is addition or remove, we compute the following probability function $P_a(k)$ which is similar with that introduced in Ref. [4]:

$$P_a(k)=\frac{\alpha(n-k)}{\alpha(n-k)+(1-\alpha)k}$$

where n is the number of network nodes, k is the current number of group nodes on the multicast tree and α is a parameter between (0,1). α represents the ratio of the number of group nodes to the number of the network nodes. To determine whether the next modification will be a join or leave, we compute a random number $r(0\leqslant r<1)$. If $P_a(k)<r$, the new request is *RmReq* and randomly one of the group members on the current tree is determined as the node to be deleted; if $P_a(k)\geqslant r$, the next request is *JoinReq* and a non-group member is randomly selected as the node to be added.

In the simulation, we set $\alpha=0.15$, $D=80$ms, the network node number $=100$. The simulation results are shown in Fig. 5. Figure 5(a) shows the results when the number of requests is small (<10) and Fig. 5(b) shows the case for large number of requests ($\leqslant 50$). From both figures it is easy to see that the cost inefficiency for DDDCLCMR increases slightly with the number of requests under both situations. However, NAIVE has a cost performance that is always much worse than DDDCLCMR. NAIVE is up to 60% worse than BSMA, while DDDCLCMR is always within 10% worse than BSMA from small number to several hundreds of requests in our simulation.

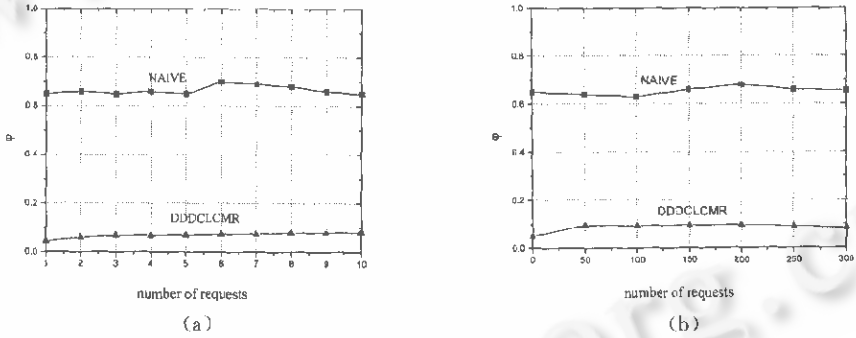


Fig. 5 Percentage excess cost over BSMA versus number of add/remove requests, for number of network nodes $=100$ and delay bound $=80$ms

5 Conclusions

In this paper, we studied the delay-constrained least-cost multicast routing problem which is known to be NP-complete, and proposed a heuristic algorithm, called distributed dynamic delay-constrained least-cost multicast routing heuristic (DDDCLCMR). DDDCLCMR is a distributed dynamic multicast routing algorithm based on the unicast heuristic DCLC-DSF. In DDDCLCMR, the source of the multicast tree needs only limited computation or may even not be involved in the route computation. When group membership changes, the existing multicast tree is perturbed as little as possible.

We proved the correctness of DDDCLCMR by showing that it is always capable of constructing a delay-constrained multicast tree, if such a multicast tree exists. Simulation results show that DDDCLCMR requires much fewer messages on the average. We compared the performance of DDDCLCMR with BSMA, which has the best cost performance among all the proposed delay-constrained static multicast heuristics. We also compared DDDCLCMR with NAIVE, a typical dynamic delay-constrained multicast routing algorithm. Our evaluation of the cost performance of the algorithms showed that DDDCLCMR is always within 10% from the BSMA, while NAIVE is

up to 70% worse than BSMA in some cases.

In summary, DDDCLCMR is a simple, efficient, distributed multicast routing algorithm that scales well to large network sizes. This encourages us to use it as a starting point for implementing a multicast routing protocol that is capable of providing QoS guarantees for real-time multicast applications. Future work should focus on mechanisms to cope with transient situations when the contents of the cost vectors and the delay vectors at different nodes are not consistent. In addition, we would like to extend DDDCLCMR to other NP-complete QoS multicast routing.

References:

[1] Kompella, V. P., Pasquale, J. C., Polyzos, G. C. Multicast routing for multimedia communication. IEEE/ACM Transactions on Networking, 1993,1(3):286~292.

[2] Guo, L., Matta, I. QDMR: an efficient QoS dependent multicast routing algorithm. Technical Report, NU-CCS-98-05, College of Computer Science, North-eastern University, 1998.

[3] Parsa, M., Zhu, Q., Garcia-Luna-Aceves, J. J. An iterative algorithm for delay-constrained minimum cost multicasting. IEEE/ACM Transactions on Networking, 1998,6(4):461~474.

[4] Waxman, B. M. Routing of multipoint connections. IEEE JSAC, 1988,6(9):1617~1622.

[5] Kadirire, J., Knight, G. Comparison of dynamic multicast routing algorithms for wide-area packet switched (Asynchronous Transfer Mode) networks. In: IEEE INFOCOM'95. Boston Massachusetts: IEEE Press, 1995. 212~219. http://www.ieee-infocom.org/1995

[6] Bauer, F., Varma, A. ARIES: a rearrangeable inexpensive edge-based on-line steiner algorithm. In: Proceedings of GLOBECOM'95. Singapore, 1995. 382~397. http://www.ieee.org/comsoc/GC-95.html

[7] Doar, M., Leslie, I. How bad is naive multicast routing. In: IEEE INFOCOM'93. San Francisco, California: IEEE Press, 1993. 82~89. http://www.ieee infocom.org/1993

[8] Biersack, E., Nonnenmacher, J. WAVE: a new multicast routing algorithm for static and dynamic multicast groups. In: Proceedings of the 5th Workshop on Network and Operation System Support for Digital Audio and Video. Durham, New Hampshire, 1995. 243~254. http://hulk.bu.edu/nossdav95/abstracts.html

[9] Wang, Z. Y., Shi, B. X., Zou, L. A distributed heuristic algorithm on delay-constrained least-cost unicast routing. In: Gong, K., Niu, Z. S. eds. Proceedings of the 2000 International Conference on Communication Technology. Beijing, China: IEEE Press, 2000. 853~859.

[10] Garey, M. R., Johnson, D. Computers and Intractability: a Guide to the Theory of NP Completeness. New York: W. H. Freeman and Co., 1979. 205~221.

[11] Salama, H. F., Reeves, D. S., Viniotis, Y. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. IEEE JSAC, 1997,15(3):332~345.

Appendix

The Proof of Theorem 4.

Obviously, there exists $\text{delay}(p_{d\min}(w,t)) \leqslant \text{delay}(w,v) + \text{delay}(p_{d\min}(v,t))$. Therefore,

$$\text{delay}(p_T(s,v)) + \text{delay}(p_{d\min}(v,t)) = \text{delay}(p_T(s,w)) + \text{delay}(w,v) + \text{delay}(p_{d\min}(v,t)) \leqslant$$
$$\text{delay}(p_T(s,w)) + \text{delay}(p_{d\min}(w,t)) > D_c$$

Theorem 4 holds.

The Proof of Theorem 5.

Theorem 5 is equivalent to:

(1) If a delay-constrained multicast tree constituted by source s and destination set M exists, DDDCLCMR will construct one without fail. And

(2) If DDDCLCMR constructs a multicast tree, the tree will be delay-constrained.

Proof 1. As to (1), we prove its converse-negative proposition, which is equivalent to the original proposition, i.e. if DDCLCMR fails to construct a delay-constrained multicast tree constituted by source s and destination set M, no delay-constrained multicast tree exists.

DDDCLCMR failing to construct a multicast tree means that at least one destination node $t \in M$ fails to join the multicast sub-tree. Let T_i be the multicast sub-tree while adding destination node t.

According to DDDCLCMR, DDDCLCMR fails to add t only if $\text{delay}(p_{d\min}(s,t)) > D_t$, so that $\text{delay}(p(s,t)) \geqslant \text{delay}(p_{d\min}(s,t)) > D_t$ for any path $p(s,t)$ from s to t. Therefore, no path from s to t satisfies the delay constraint, and no multicast tree constituted by s and M satisfies the delay bound.

(1) holds.

Proof 2. Without losing generality, we discuss any destination node t. Let T_i be the multicast sub-tree when adding t, and v the connection node of t and T_i.

According to the destination node adding operation, t succeeds in adding only if $\text{delay}(p_T(s,v)) + \text{delay}(p_{d\min}(v,t)) \leqslant D_t$, which means there exists at least one path from v to t that satisfies the delay constraint $D_t - \text{delay}(p_T(s,v))$.

According to Theorem 3, DCLC-DSF will construct a path $p_{dsf}(v,t)$ from v to t which satisfies:

$$\text{delay}(p_{dsf}(v,t)) \leqslant D_t - \text{delay}(p_T(s,v))$$

i.e.

$$\text{delay}(p_{dsf}(v,t)) + \text{delay}(p_T(s,v)) \leqslant D_t$$

Therefore, the routing path $p_T(s,v) + p_{dsf}(v,t)$ of t satisfies the delay constraint D_t.

(2) holds.

By combining (1) and (2), Theorem 5 holds.

一种延时约束费用最小分布式动态组播路由算法

王征应, 石冰心, 刘 伟

(华中科技大学 电子信息工程系,湖北 武汉 430074)

摘要: 多媒体应用一般包含多个组播成员,它消耗大量的网络资源且有严格的端端延时约束.针对这个问题,提出了一种延时约束费用最小的分布式动态组播路由启发算法 DDDDCLCMR(distributed dynamic delay-constrained least-cost multicast routing algorithm).在 DDDDCLCMR 算法中,组播源点很少或根本不参与路由计算;即使组播成员发生改变,组播树变化也很小,算法扩展性好.实验结果表明,无论组播成员改变与否,DDDCLCMR 算法都能获得满足延时约束且费用很低的组播树.

关键词: 服务质量;组播;动态路由;分布式算法;启发式

中图法分类号: TP393 **文献标识码**: A