Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Effective multicasting algorithm for dynamic membership with delay constraint

CHEN Lin (陈 琳)[1,2], XU Zheng-quan (徐正全)[2]

([1]*Computer College, Yangtze University, Jingzhou 434102, China*)

([2]*State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430072, China*)

E-mail: chan_@sohu.com; xuzq@firstlink.com.cn

Received May 21, 2005; revision accepted Sept. 10, 2005

Abstract: This paper proposes an effective heuristic algorithm for dynamic multicast routing with delay-constrained DDMR. The tree constructed by DDMR has the following characteristics: (1) multicast tree changes with the dynamic memberships; (2) the cost of the tree is as small as possible at each node addition/removal event; (3) all of the path delay meet a fixed delay constraint; (4) minimal perturbation to an existing tree. The proposed algorithm is based on "damage" and "usefulness" concepts proposed in previous work, and has a new parameter *bf* (Balancing Factor) for judging whether or not to rearrange a tree region when membership changes. Mutation operation in Genetic Algorithm (GA) is also employed to find an attached node for a new adding node. Simulation showed that our algorithm performs well and is better than static heuristic algorithms, in term of cost especially.

Key words: Multicast, Routing, Delay constraint, Quality of Service (QoS)
doi:10.1631/jzus.2006.A0156 **Document code:** A **CLC number:** TP391

INTRODUCTION

Some new emerging applications involve information transmission in a network. Different from traditional single-cast applications, these new arising applications are called multicast as they contain a single sender and multiple receivers. In general, it is needed to construct a multicast routing tree to deliver information along the tree branches. Because links in the tree are shared by paths to different receivers, just one information copy is needed in the middle node. This approach may reduce the number of information copy and save middle node run-time. The application of multicast routing tree maximizes network resource utilization.

During dynamic multicast session, membership changes randomly. Based on the nature of group membership, multicast groups are classified into two categories: static groups in which membership remains unchanged throughout session, and dynamic groups in which nodes can leave and/or join the group dynamically.

Generally, multicast applications have QoS (Quality of Service) requirements. One of the most common QoS requirements is an upper bound on the delay between the sender and receivers. It has been established that the problem to construct delay-constrained least-cost multicast trees for multicast is a Steiner tree problem (Wang and Richards, 1992; Winter, 1987), and is also an NP-hand problem.

In this paper, we address the problem of dynamic multicast routing with delay constraint (which is named as DDMR). Two previous concepts "damage" and "usefulness" are accommodated and a new adjusting parameter *bf* (Balancing Factor) is provided. Mutation operation in Genetic Algorithm (GA) is also employed to find an attached node for a new adding node. Simulation showed that our algorithm performs better than static heuristic algorithms, in term of cost especially.

EXISTING DYNAMIC MULTICAST ROUTING ALGORITHM

During multicast session, a member may add to/leave from multicast group randomly. The problem of updating a multicast group is known as the on-line Steiner problem in networks, which is also NP-complete (Fujinoki and Christensen, 2000). In the extreme case, the on-line multicast problem can be solved as a sequence of static multicast problems, by rebuilding the tree at each stage using static Steiner heuristics. However, this approach is too expensive and unsuitable for ongoing real-time multicast session that cannot tolerate the disturbances caused by excessive changes in the multicast tree after each additions or removal.

Some previous on-line dynamic multicast problems have been addressed, but only on the issue of cost-minimization.

Kheong *et al*.(2001) proposed an algorithm to speed up the creation of multicast routing trees in the case of dynamic changes. The idea is to maintain caches of pre-computed multicast trees from previous groups. The cache can be used to quickly find new paths connecting some of the members of the group. An algorithm for retrieving data from the path cache was proposed, which finds similarities between the previous and the current multicast groups. Then the algorithm constructs a connecting path using parts of the paths store in the cache.

Virtual Trunk Dynamic Multicast (VTDM) is a dynamic multicast routing algorithm (Lin and Lai, 1998). It belongs to non-rearrangeable heuristics. A Virtual Trunk (VT) is a tree of the underlying graph and is used as a template for constructing multicast trees. The VTDM routing algorithm constructs multicast trees based on the VT. First the algorithm finds trunk nodes using shortest path strategy, then an underlying graph is established through these trunk nodes and does not change during the lifetime of connectivity.

The on-line multicast problem was first presented by Waxman (1988), and the on-line heuristics is divided into two categories: those that allow reconfiguration/rearrangement of trees and those that do not.

A simple non-rearrangeable heuristic algorithm was proposed by Waxman (1988) with the aim to minimize the perturbation to the existing multicast tree. A rearrangeable heuristic algorithm (Imase and Waxman, 1991) was proposed by enforcing an upper bound on the distance between nodes. A rearrangeable heuristic algorithm (Kadirire and Knight, 1995) was provided based on the notion of the geographic spread of a tree. The recently proposed ARIES heuristic algorithm (Bauer and Varma, 1997) aimed to combine the computational simplicity of GREEDY with better competitiveness, when damage to a fragment reaches a threshold, the ARIES algorithm will trigger a reconfiguration operation.

Methods mentioned above are unconstrained case. When delay constraint is taken into account, it will be constrained case.

Lagrangan-relaxation-based algorithm (LRA) (Hong *et al*., 1998) takes delay and cost simultaneously into consideration. It handles node removal by merely pruning unnecessary branches form the tree, and handles node addition by reducing the addition problem to determining a delay-bounded least-cost path between the new node and source node.

With node addition/removal, especially node removal, the performance of the multicast tree will degrade quickly if just marking the leaving node or pruning node branches including a deleted leaf node. LRA (Hong *et al*., 1998) algorithm only based on avoiding disturbance to the existing tree but not adjusting the useless part of the tree lead to poor performance.

Raghavan *et al*.(1999)'s algorithm based on a concept called Quality Factor (QF) representing the usefulness of a portion of the multicast tree to the overall multicast tree. When the QF of a region drops below a certain threshold, a rearrangement technique is used to modify the tree, where the path connecting the source node and regional nodes is delay-constrained and the cost is minimized.

The Path Length Control (PLC) (Fujinoki and Christensen, 2000) algorithm supports dynamic multicast tree with bounded bandwidth and path length, and aims to balance the tree performance achieved by SPT and GREEDY algorithm. PLC algorithm defines the trunk region and connection region, the nodes in the trunk are assigned a priority to connect adding node, and adding node selects one from multiple feasible paths based on a selection function.

Feng and Peter (1999) devised a heuristic algo-

rithm with the main goal of allowing easy insertion of new nodes in a multicast group. The algorithm is similar to Prim's algorithm for spanning trees in which it, at each step, takes a non-connected destination with minimum cost and tries to add this destination to the current solution. The algorithm uses a priority queue Q where the already connected elements are stored. The key in this priority queue is the total delay between the elements and the source node. The algorithm uses a parameter k to determine how to compute the path from a destination to the current tree. Given a value of k, the algorithm computes k minimum delay paths from the current destination d to each of the smallest k elements in the priority queue. Then, the best of the paths is chosen to be part of the routing tree. Different k values will change the amount of effort needed to connect to destination.

Algorithms (Waxman, 1988; Raghavan *et al*., 1999) were proposed to realize dynamic multicast using a threshold represented by "damage" and "usefulness" concepts. But the individual one cannot convincingly reconfigure a multicast tree when it reaches a threshold. So an integrated scheme will be considered in this paper. In addition, the concept of trunk node (Lin and Lai, 1998) is acceptable for finding an attached node for a receiver. In this paper, mutation operation in GA (Wang *et al*., 2001; Kadtie, 1994; Xiang, 1999) will be used to search multiple constrained paths and find attached node.

NETWORK MODEL AND PROBLEM DEFINITION

Generally, a network is modelled as an undirected connected graph $G=(V, E)$, where V is the set of nodes and E is the set of link. An edge $e\in E$ connecting nodes u and v will be denoted as (u, v). We associate each link $e\in E$ with two functions: delay function $Delay(e)$ and cost function $Cost(e)$.

A path $P=(v_1, v_2, \ldots, v_n)$ in the network, has two associated cost and delay characteristics:

$$Cost(P)=\sum_{i=1}^{n} C(v_i, v_{i+1}),\ Delay(P)=\sum_{i=1}^{n} Dl(v_i, v_{i+1}).$$

The delay and cost of a path P are additive metrics. Given a path P, its delay/cost is the sum of delay/cost of more links along with the P.

Similarly, a tree $T=(V_T, E_T)$, is a sub-graph of G, and has an associated cost function:

$$C(T)=\sum_{e\in E_T} C(e).$$

For every node x in tree T, a degree function $\deg_T(x)$ is assigned, which represents the number of branches at that node.

Successive updates of a multicast group are modelled as a request vector $R=(r_1, r_2, \ldots, r_m)$. Each request r_i is either of the form (v, add) (indicating the addition of v to the multicast group) or of the form $(v, remove)$ (indicating the removal of v from the multicast group).

A multicast call in a network is modelled as a 3-tuple: $C=(s, R, \Delta)$. Where $s\in V$ is sender/source, R is a request sequence of receivers mentioned above, and Δ is delay constraint.

Given such a call C on the network G, the objective is to determine a sequence of multicast trees T_1, $T_2, \ldots, T_m$ such that for each T_i $(1\le i\le m)$, the following are true:

(1) T_i spans all the current receivers.

(2) Any path P in the multicast T_i meets delay constraint Δ.

(3) The cost of tree T_i is as small as possible after each update.

PROPOSED ALGORITHM

Basic idea

The goals of the proposed algorithm are to ensure that the cost of the tree is as small as possible and that all paths in the tree satisfy delay constraint at each multicast event. On the other hand, real-time multicast tree cannot tolerate frequent large changes to the multicast tree as packets are constantly in flight within the tree. So the other goal is to find a moment (threshold) such that the tree is rearranged when the quality of the tree is downgraded and interruptions are reduced.

Bauer and Varma (1997) suggested the rearrangeable technique that allows the quality of a tree to degrade down to a fixed threshold. Any further degradation results in a rearrangement of a portion of the tree using a simple static Steiner heuristics. It adopts the idea of accumulating the "damage" to a tree and

triggering a rearrangement based on a fixed threshold.

Raghavan *et al*.(1999) took into account the "contribution" of a region to the rest of the tree and used a new concept "quality factor" to measure the "usefulness" of a region; and believed that even if a region of a tree has degraded to a certain extent, as long as it is serving a large number of multicast members, it should not be disturbed.

In order to decrease the multicast disturbance, a more exact moment must be defined. The algorithm proposed in this paper takes into consideration the advantage of the above two ideas simultaneously and provides a relatively more reasonable triggering parameter called Balancing Factor (*bf*). When the *bf* of a region reaches a specific threshold, reconfiguration will be triggered to reconstruct a region of tree or whole tree.

When a receiver requests to join multicast group, mutation operation in GA is adopted to achieve multiple constrained paths. A mechanism will be used to find one attached node from multiple intersecting nodes between the existing tree and multiple constrained paths for a receiver. The details are described in the following sections.

Definition and notation

Damage to a region R: R is a sub-tree rooted at one node u. During one multicast session, the number of node adding/leaving event is accumulated and referred to as the damage to a region R (denoted as $Dmg(R)$), This concept expresses the alterative level of a region because of node adding/leaving. Intuitively, a region with more $Dmg(R)$ should be reconfigured (Bauer and Varma, 1997).

Usefulness of a region R: a region provides service for multiple receivers. Usefulness of a region R is defined as the number of receivers in the region and is referred to as $Ufn(R)$. Even if the region is damaged seriously, because it services a number of members, it should not be reconfigured. This is the reason why parameter *bf is* designed.

Balancing factor (*bf*): This parameter takes the "damage" and "usefulness" of a region R into consideration simultaneously. It is defined as:

$$BF(R) = Ufn(R) / Dmg(R).$$

The smaller the $Dmg(R)$ value, the larger is the $Ufn(R)$ value, and the larger value of $BF(R)$ will be. $BF(R)$ keeps two ideas ("damage" and "usefulness") in balance intuitively (Raghavan *et al*., 1999; Bauer and Varma, 1997).

Multiple delay-bounded paths method based on mutation operation

In a network, there are multiple paths that satisfy delay-bounded low-cost. RA (Korkmaz and Krunz, 2001) is a random algorithm for searching delay-bounded least-cost path. RA algorithm views sender node as current node first, and searches one node from its neighbor nodes of the current node such that the path being built is delay-bounded and has least cost. Then one of its neighbor nodes is viewed as the current node and the search is going on, until the current node is a receiver node.

The RA algorithm may find the exact path that meets the delay bound and has least cost. There are some other paths that also satisfy the delay bound, but the cost of these paths is certainly not lowest. These paths have similarity with the least cost path found by RA algorithm. In other word, these paths that meet the delay constraint may share some links, this implies that it is easy to achieve another path by modifying one path. This is the principle of path similarity. Here the mutation operation in GA is employed to obtain multiple paths meeting delay constraint.

Fig.1 shows the path mutation operation, s is the sender and t is one receiver. The original path is obtained by RA algorithm. Any node m in the original path, it will be mutated as node m'. By using RA algorithm, two partial paths are obtained, corresponding named as r_1 and r_2, r_1 is the path from s to m' and r_2 is the path from m' to t. The path r_1+r_2 is path obtained by mutation operation and include node m'. The mutate path is delay bounded and has low cost but not least cost. One mutation operation on a node generates one mutation path. If there are k' nodes in the original path, k ($k \le k'$) mutate paths will be obtained (if there are multiple same paths, only one path is counted), and these k' paths have similarity. This

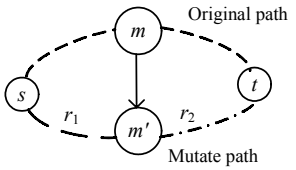

Fig.1 Path mutation operation

procedure is named as find_*k*_paths.

The original path achieved by RA and its similar paths obtained by find_*k*_paths algorithm will be saved in the receiver as cached paths.

Node addition

When an adding request arrives, find_*k*_paths algorithm is used to find *k* paths. The multiple intersect nodes between *k* paths and the current tree are all candidate attached nodes. A selection function *SF* then selects an intersect node as attached node. The adding node connects to multicast tree through partial path between attached node and adding node (Fig2).

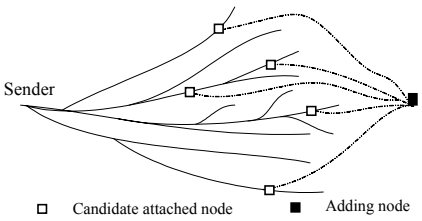

Fig.2 Add a receiver

Solid line in Fig.2 represents an existing multicast tree *T* and the dot line represents *k* partial paths achieved by find_*k*_paths algorithm mentioned above. And there are multiple intersect nodes that may be selected as attached node. In a special case, there is merely one intersect node between an existing tree and *k* paths, the intersect node is exactly the source node and will be attached node.

If a new receiver requests that it be added to the multicast group at the first time, *k* paths are calculated using find_*k*_paths algorithm and saved in the new receiver. If the adding receiver has been a member of the group, *k* paths cached in adding node are popped. Because there are multiple intersect candidate nodes between *k* paths and multicast tree *T*, a selection function to select one intersect node as attached node is defined as follows:

$$SF(y)=\frac{C(P)}{(\Delta-D(s,y)-D(P))},$$

where *P* is partial path between adding node and these intersect nodes. Intuitively, *SF* function selects the intersect node which has low link cost and long residual delay as attached node. Generally, the candidate node with least *SF* value will be selected as attached node.

Given a request $r=(x, add)$ on the multicast tree *T*, the following steps are taken to service this adding request:

(1) If nod *x* exists in the tree, return.

(2) If there are no paths cached in node *x*, then *k* paths are calculated using find_*k*_paths algorithm and then saved in node *x*; otherwise cached paths in *x* are popped out.

Some cached paths could include failed nodes or links, in this worst case, if all of the cached paths cannot connect to this node *x* to the tree, new *k* paths will be calculated and cached.

(3) Executing *SF* function to select one from multiple candidate attached nodes.

(4) Adding node *x* to multicast tree *T* using partial path along attached node to node *x*.

The special case is that the sender will be selected as attached node. In this case the added path will be the delay-bounded least-cost path.

Node removal

When a leaving request $r=(x, remove)$ arrives, pruning technique is employed, and the following two steps are executed:

(1) If *x* is a leaf node, then the partial path servicing only *x* is pruned from the tree.

(2) If *x* is not a leaf node, it will become a relay node. Whether or not the rearrangement is triggered will be judged by the *bf* value.

Rearrangement algorithm

Because of node adding/leaving, the tree quality will be degraded, whether or not to trigger a rearrangement rest on the *bf* value.

(1) If $bf>\rho$, rearrangement is not triggered;

(2) If $bf\leq\rho$, rearrangement is triggered,

where ρ is a parameter. When rearranging a region, the paths cached in the node of this region will be used to find attached nodes for correlated nodes. Use of cached paths saves run-time.

EVALUATION OT THE ALGORITHM

Experimental network

In the experiment, network model (Kadtie, 1994) was used to generate a realistic connected graph. As each node pair (v, u), it is decided by the probability

that depends on the distance between these two nodes to judge whether there exists an edge. The distance is chosen randomly in $\{1,\ldots,L\}$ by a uniform random distribution. The probability formula is:

$$P((v,u)) = \beta \exp(-d(v,u)/(L\alpha)),$$

where $d(v, u)$ is the distance between node v and node u, L is the biggest distance between any two nodes, $\alpha,\beta\in(0,1)$ are real number. The link density can be increased by increasing β, the density of short links can be increased to over that of longer ones by increasing α. Setting parameters α and β to 0.25 and 0.2 respectively can generate a graph that roughly looks like the geographical maps of the major nodes in the Internet.

Sequences of request for adding/removing nodes are generated according to a simple probability model that may also be used to determine whether a request is to add or remove a node. The probability function $P_C(m)$ (Waxman, 1988; Kadirire and Knight, 1995) is defined as follow:

$$P_C(m) = \frac{\gamma(N-m)}{\gamma(N-m)+(1-\gamma)m},$$

where m is the current number of nodes in the multicast group, N is the number of nodes in the network, and γ is a real number parameter in the range (0, 1) and determines the size of the multicast group in equilibrium.

If the request is a node addition, a node is randomly chosen from those that are not in the multicast group and then added to the multicast group. If the request is a node removal, a node excluding the source node is randomly chosen to be removed from the multicast group.

Experimental results

Results of DDMR simulation are discussed in this subsection. For comparison, we also include simulation results for CSPT (Waxman, 1988) and RA (Korkmaz and Krunz, 2001) algorithms modified to meet delay constraint and have low cost. The metrics we use is cost of tree, maximal path delay, relation between cost and delay constraints, relation between cost and *bf* value, and relation between number of reconfigurations and *bf*.

All the experiments used a network with 120 nodes (N=120). The number of multicast events (addition/removal) was 150. In the DDMR algorithm, when an adding node is an intermediate node, cost of tree will not be computed and compared with other algorithms.

If $P_C(m)$ is larger than 0.4, the simulation can guarantees that there are about 40 members in the multicast group; when $P_C(m)$ is larger than 0.35, the simulation can guarantee that there are about 90 group members.

Fig.3 shows that the cost of tree changes dynamically with the multicast event. All events are node addition requests before the fortieth one after which nodes may add to/leave from a multicast group. We calculate cost of tree at each event. Fig.3 shows that the cost of tree changes dynamically and that the cost that DDMR algorithm achieves is the smallest among the three algorithms. The cost that CSPT achieves is larger than that of the other two algorithms. Experimental results show that rearrangements are triggered 2 times.

Fig.4 gives the maximal path delay at each multicast event where delay constraint is 20. All of the practical paths are less than 20, in other words, all paths in trees meet the delay constraint.

Fig.5 shows how the delay constraint impacts the cost of tree. At different delay constrained point, the DDMR algorithm yielded smallest cost. When the delay constraint is bigger than a specified value, the problem will become unconstrained case, when the cost that DDMR algorithm achieves is also the smallest. Fig.8 shows three curves yielded by DDMR algorithm at different delay constraint. It is obvious that the larger the delay constraint, the smaller is the cost of the tree.

The *bf* in DDMR is a threshold that decides whether or not to rearrange the multicast tree. Different threshold has different effect on the cost of the tree and the number of tree reconfiguration. Fig.6 shows that DDMR also achieves smallest multicast cost at every *bf*, but that the cost of the tree changes regularly. Fig.7 gives the relation between reconfiguration count and *bf*. When *bf* is in (0,0.7), reconfiguration is not triggered. If *bf* is 0.8, reconfiguration is triggered 2 times; if *bf* is 4, reconfiguration is triggered 5 times. Fig.10 shows that large *bf* leads to larger reconfiguration count. Reconfiguration rate

was 0.67%~3.3% in the experiment.

The above experiments were conducted on a sparse multicast group with only 40 members, so another experiment was conducted on a dense multicast group with about 90 group members.

Fig.8 shows that the cost of multicast tree changes dynamically with the multicast events as in Fig.3. Before the multicast event is equal in number to 90, all events are adding requests, after that, nodes may add to/leave from the multicast group. The simulation result showed that DDMR algorithm is the best approach before the number of events is 60, when the multicast event goes on, the cost of DDMR is better than that of CSPT but worse than that of RA, because the RA algorithm is an accurate approach to achieve least path cost.

Fig.9 shows that the maximal path delays of the above three algorithms are all less than those of the delay constraint algorithms, in other words, all paths at each multicast event meet the delay constraint.

Fig.10 shows the relation between *bf* parameter and reconfiguration count. Compared with Fig.7, this figure shows that rearrangement is triggered when *bf* is 1.8, but that the rearrangement is triggered when *bf* is 0.8 in Fig.7. This means that the *bf* value should be larger when the multicast group is dense. What *bf* should be depends on network environment and application requirement, for example, if the multicast group is dense, the *bf* value should be smaller, otherwise the *bf* value should be larger.

CONCLUSION

An effective algorithm for delay-constrained dynamic multicast routing (DDMR) is proposed in this paper aimed to minimize the cost of the tree and decrease disturbance to multicast session. Based on a previous algorithm, a parameter bf is designed to evaluate branch of tree and a multiple similar paths algorithm based on mutation operation is proposed to find the attached node. Simulation studies showed that our proposed algorithm performs well.

We also give the relation between *bf* and reconfiguration number. Larger *bf* triggers more rearrangements, with the denser multicast group having smaller *bf* value. This study has practical instructional meaning in realizing multicast applications. Future

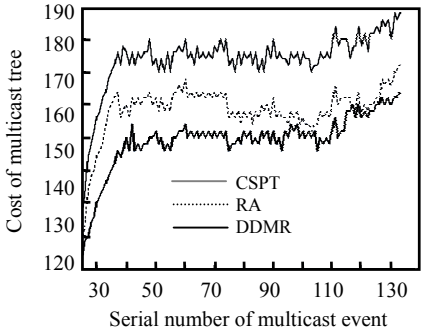

Fig.3 Cost of tree with multicast event (*N*=120, *bf*=0.8, *Δ*=30, 40 members)

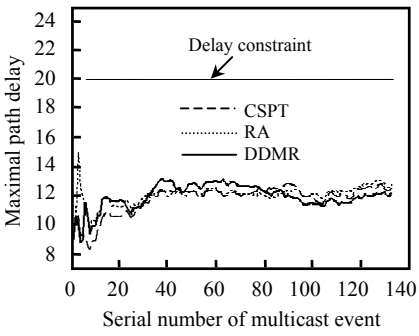

Fig.4 Maximal path delay of tree with multicast event (*N*=120, *bf*=0.8, *Δ*=20, 40 members)

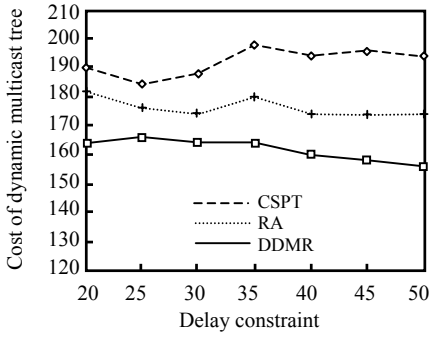

Fig.5 Relation between delay constrained and cost (*N*=120, *bf*=0.8, *Δ*=20)

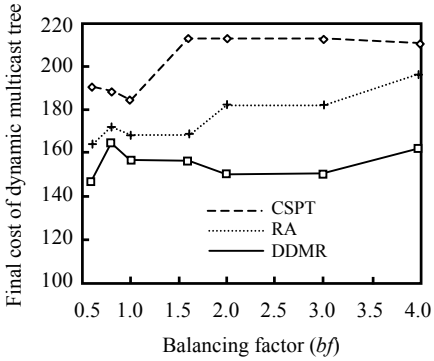

Fig.6 Relation of *bf* and cost (*N*=120, *Δ*=20)

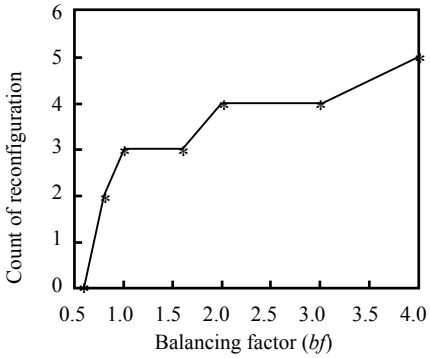

Fig.7 Relation between *bf* and reconfiguration (*N*=120, *Δ*=20)

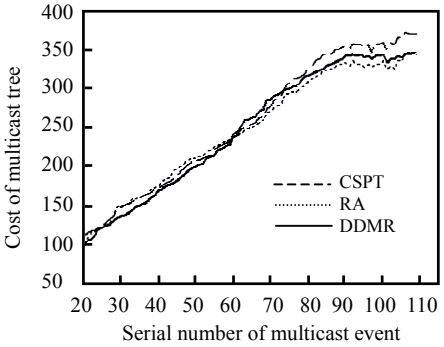

Fig.8 Cost of tree with multicast event (*N*=120, *bf*=4.5, *Δ*=20, 90 members)

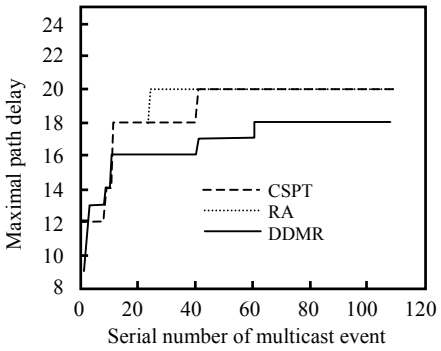

Fig.9 Cost of tree with multicast event (*N*=120, *bf*=0.45, *Δ*=20, 90 members)

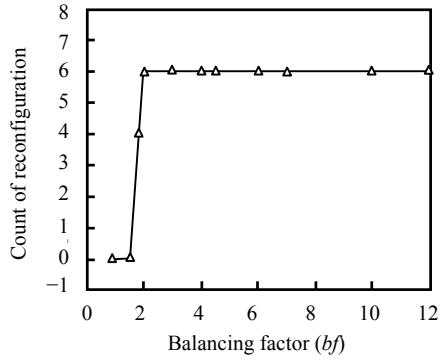

Fig.10 Relation between *bf* and reconfiguration (*N*=120, *Δ*=20, 90 members)

investigation will focus on a distributed design of our algorithm.

References

Bauer, F., Varma, A., 1997. ARIES: A rearrangeable inexpensive edge-based on-line Steiner algorithm. *IEEE JSAC*, **15**(3):382-397.

Feng, G., Peter, T., 1999. Efficient multicast routing with delay constraint. *International Journal of Communication Systems*, **12**(3):181-195. [doi:10.1002/(SICI)1099-1131(199905/ 06)12:3<181::AID-DAC394>3.0.CO;2-Y]

Fujinoki, H., Christensen, K.J., 2000. A routing algorithm for dynamic multicast trees with end-to-end path length control. *Computer Communications*, **23**(2):101-114. [doi:10.1016/S0140-3664(99)00167-X]

Hong, S., Lee, H., Park, B.H., 1998. An Efficient Multicast Routing Algorithm for Delay-sensitive Applications with Dynamic Membership. Proc. of IEEE INFOCOM, p.1433-1440.

Imase, M., Waxman, B., 1991. Dynamic Steiner tree problems. *SIAM J. Disc. Math.*, **4**(3):369-384. [doi:10.1137/0404033]

Kadirire, J., Knight, G., 1995. Comparison of Dynamic Multicast Routing Algorithms for Wide-Area Packet Switched (Asynchronous Transfer Mode). Networks, IEEE INFOCOM, p.212-219.

Kadtie, J., 1994. Minimizing packet copies in multicast routing by exploiting geographic spread. *ACM SIGCOMM Computer Communication Review*, **24**:47-63.

Kheong, C., Siew, D., Feng, G., 2001. Efficient Setup for Multicast Connections Using Tree Caching. Proc. IEEE INFOCOM, p.249-258.

Korkmaz, T., Krunz, M., 2001. A randomized algorithm for finding a path subject to multiple QoS constraints. *Computer Networks*, **36**(2-3):251-268. [doi:10.1016/S1389-1286(00)00209-7]

Lin, H.C., Lai, S.C., 1998. VTDM—A Dynamic Multicast Routing Algorithm. Proc. IEEE INFOCOM'98.

Raghavan, S., Manimaran, G., Siva Ram Murthy, C., 1999. A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. *IEEE/ACM Trans. Networking*, **7**(4):514-529. [doi:10.1109/90.793020]

Wang, F.H., Richards, D., 1992. Steiner tree problems. *Networks*, **22**:55-89.

Wang, Z., Shi, B., Zhao, E., 2001. Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm. *Computer Communications*, **24**(7-8):685-692. [doi:10.1016/S0140-3664(00)00273-5]

Waxman, B.M., 1988. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, **6**(9):1617-1622. [doi:10.1109/49.12889]

Winter, P., 1987. Steiner problem in networks: a survey. *Networks*, **17**(2):129-167.

Xiang, F., Luo, J.Z., Wu, J.Y., Gu, G.Q., 1999. QoS routing based on genetic algorithm. *Computer Communications*, **22**(15-16):1392-1399. [doi:10.1016/S0140-3664(99)00113-9]