

PAPER

A Multipoint-to-Multipoint Routing Method for Load Balanced Communications in Large Scale Networks

Hideaki TANIOKA[†], *Student Member*, Kazuhiko KINOSHITA^{††}, and Koso MURAKAMI^{††}, *Regular Members*

SUMMARY Recently, diverse multimedia applications with stringent multiple Quality of Service (QoS) requirements have been increasing. In particular, multicast communication has become more popular because of its availability and for efficient use of network resources. Most multicasts are point-to-multipoint, in which a source delivers data to multiple designated recipients, such as for video or audio distribution. In the near future, multipoint-to-multipoint communication services, including multimedia collaborations such as video conferencing and distant-learning, will be developed. However, when a conventional multicast routing algorithm is applied to a multipoint-to-multipoint communication service, it might result in excessive traffic concentration on some links. Therefore, we propose a new multipoint-to-multipoint routing method. It utilizes the Fallback+ algorithm to perform multicast routing for the purpose of satisfying multiple QoS requirements and alleviating traffic concentrations. Simulation experiments show that our method improves traffic load balance and achieves efficient use of network resources.

key words: *multimedia network, Fallback+, multipoint-to-multipoint communication, multicast, QoS guarantee*

1. Introduction

Diverse multimedia applications with stringent multiple Quality of Service (QoS) requirements have been increasing. To realize these services, QoS routing is one of the key elements. QoS routing is generally translated as the problem of determining a feasible route subject to multiple constraints. Note that it is NP-complete [1], so that many heuristic algorithms [1]–[9] have been proposed to satisfy all QoS requirements in real time.

On the other hand, multicast communications have been becoming popular because of their availability and efficient use of network resources. The majority of them are based on point-to-multipoint multicast communication, in which a source delivers the same data to multiple designated recipients, as in video or audio distribution. In the future, major multicast communications will shift to multipoint-to-multipoint communication services, which include multimedia collaborations such as video conferencing and distant learning.

For point-to-multipoint communications, several routing algorithms have been studied [10]–[20]. Most of them treat a point to multipoint routing as a multicast tree problem

to find a tree that connects the source node to all destination nodes. These algorithms aim at constructing a tree with respect to certain optimization objectives, such as the total tree cost, or the propagation delay to each destination.

When point-to-multipoint routing is utilized to execute multipoint-to-multipoint routing, two typical methods as follows are possible ones. One method constructs a single tree that is shared by all members of a multicast group. The other method constructs a tree for each member. In this paper, we define the former method as the Shared-Tree method, and the latter the Private-Tree method.

The Shared-Tree method is seemingly simple and tempting because we can construct just a single tree. However, multiple constraints on all members make the problem intractable and the increase in the number of sources makes it harder to construct the tree. Even if the objective tree could be constructed, sharing the identical set of links would contribute to excessive traffic concentrations on those links.

We classify the Private-Tree methods into two groups, depending on the process scheduling. In the first approach, each multicast tree is sequentially constructed on a certain node. In the other approach, the multicasts are concurrently constructed on each source node. The former makes for good traffic load balancing by feedback of constructed trees, but increasing the number of source nodes affects the processing time directly. In contrast, while the latter method has a processing time less subject to the number of source nodes, the constructed multicast trees might include the same links, contributing to excessive traffic concentrations just as the Shared-Tree method does. As a result, when conventional point-to-multipoint routing techniques are applied to multipoint-to-multipoint routing without additional ideas, they might cause excessive traffic concentration on some links or require long construction times.

Therefore, we propose a new multipoint-to-multipoint routing method to address these issues. The proposed method is based on the Private-Tree method from the point of view of scalability. In particular, we adopt one of the QoS routing algorithms, Fallback+ [22], to construct a multicast tree. This is because Fallback+ finds a shortest path tree to satisfy constraints on each member and has a feature to maintain tentative routes produced during the process of finding the objective route. Moreover, to alleviate traffic concentration, an additional process is introduced as follows. After individual tree construction, every source node exchanges tree data with all others to discover which links

Manuscript received September 19, 2002.

Manuscript revised January 22, 2003.

[†]The author is with the Department of Information Systems Engineering, Graduate School of Engineering, Osaka University, Suita-shi, 565-0871 Japan.

^{††}The authors are with the Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.

other source nodes are using. Subsequently, if necessary, each source node reconstructs its own multicast tree to alleviate traffic concentration autonomously. Note that the proposed method leverages tentative routes produced through Fallback+ to reconstruct a tree.

The remainder of this paper is organized as follows. Section 2 describes the proposed methods and briefly outlines some functions related to multicast routing. Section 3 presents the time complexity of the proposed method and shows the efficiency of the proposed method through some simulation experiments. Finally, we conclude the paper in Sect. 4.

2. Proposed Routing Method

2.1 Definitions

Throughout this paper, a communication network is represented by a directed weighted graph $G = (V, E)$, where V is the set of nodes and E is the set of directed links. A link $e = (u, v) \{u, v \in V\}$ is a direct connection from node u to node v . Any link $e \in E$ has several metrics, such as the hop count $h(e)$, and the cost $c(e)$. A multicast group $G_{MC} = \{g_1, \dots, g_M\}$, where $M = |G_{MC}| \leq |V|$, is a set of source nodes. When node g_m is treated as a source node, $D_m = G_{MC} - \{g_m\}$ is the set of destination nodes and $T_{g_m} \{g_m \in G_{MC}\}$ denotes the tree constructed by the source node g_m . $T_{G_{MC}}$ denotes the set of T_{g_m} .

2.2 Existing Methods

Before beginning on an explanation of the proposed method, to clarify differences between the Shared-Tree method and the Private-Tree method, we refer to them using the illustrative examples in Fig. 1. In particular, with respect to the Private-Tree method, we classified it into two groups as follows. One constructs a multicast tree for every source node

in G_{MC} in parallel. The other constructs a multicast tree for each source node in G_{MC} sequentially. We designate the former the Parallel-method and the later the Sequential-method. In the figure, gray circles indicate source nodes, white circles are other nodes, and black circles are active nodes that are constructing a tree at the time, respectively. Dashed lines indicate bidirectional link, normal lines are the links used for a tree, and arrow lines show the direction of flow.

The Shared-method finishes a tree construction in the first step and then starts to communicate using a single tree in the second step. The number of used links is the smallest but multicast traffic converges on the identical set of links. To the contrary, the Sequential-method requires as many steps as the number of sources to construct multicast trees (e.g. three steps in Fig. 1). It can distribute multicast traffic effectively.

The Parallel-method also finishes the tree construction in the first step and starts in the second step because all source each construct their own tree autonomously. However, multicast traffic might converge on the identical set of links same as the Shared-method.

2.3 Fallback+

The algorithmic flow of Fallback+ is similar to the conventional Fallback algorithm [5] with Dijkstra's algorithm, except that it uses the available data produced through Dijkstra's algorithm more effectively. Dijkstra's algorithm is likely to produce some tentative routes from the source node to every other node in the disposal process. Note that Dijkstra's algorithm discards such route data when it finds a shorter route, because Dijkstra's algorithm aims at finding the optimal routes from the source node with respect to just one metric. Fallback+ never discards data but preserves them in a table. If the shortest route selected by Dijkstra's algorithm does not satisfy all the QoS constraints, Fallback+ generates other routes using the preserved tabular data and checks them for feasibility in turn. Thus, Fallback+ can find a feasible route with a higher probability, and the computational complexity of Fallback+ is the same as conventional Fallback routing. Thus, the proposed method adopts Fallback+ not only to construct a multicast tree with QoS constraints but also to reconstruct it effectively taking advantage of the preserved data.

2.4 Procedure

2.4.1 Outline

We demonstrate the procedure of the proposed method with Fig. 2. In the figure, we consider that each source node in G_{MC} ($M = 3$) runs a multipoint-to-multipoint communication. Note here that *multicast broker* denotes a dedicated node/server for multicast service, which manages the multicast state, such as allowing members join or leave, and records the multicast trees for each group. Thus, source

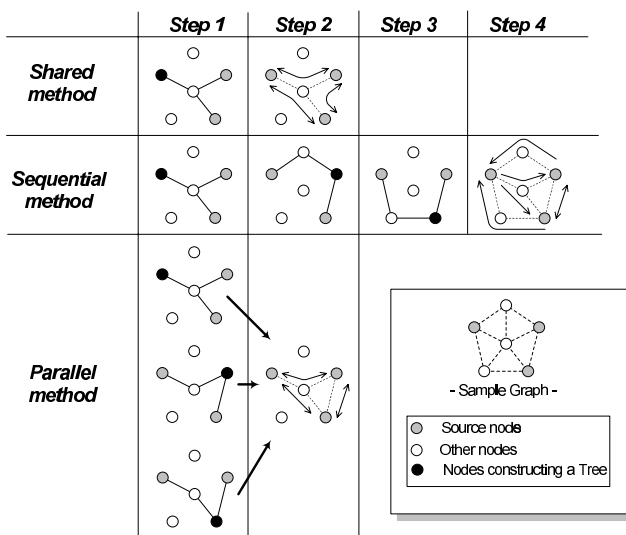


Fig. 1 Behavior of existing methods.

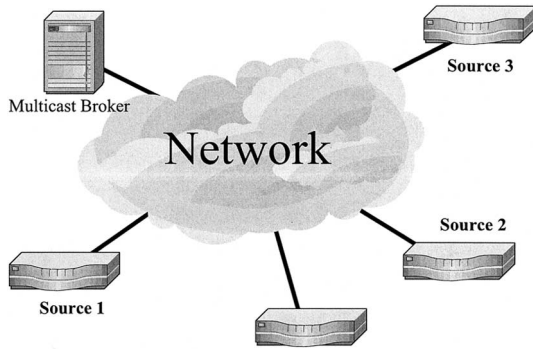


Fig. 2 Outline of the proposed method (Step.1).

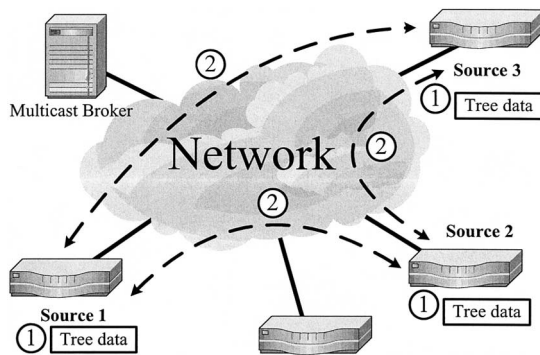


Fig. 3 Outline of the proposed method (Step.2).

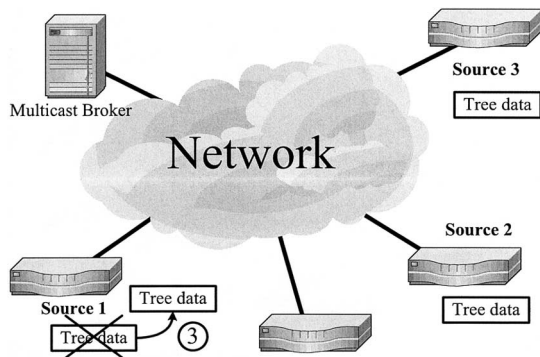


Fig. 4 Outline of the proposed method (Step.3).

nodes can maintain only their own tree data.

1. Each source node in G_{MC} constructs an individual multicast tree locally, independently of other source nodes. Note here that the tree constructed with Fallback+ satisfies multiple QoS requirements and some tentative route data is preserved (Fig. 3(1)) for later tree reconstruction.

2. Every source node exchanges its own tree data with all others, discovers which links other source nodes are using (Fig. 3(2)), and determines the state of traffic concentration.

3. When there are any links used by plural trees, the trees with such links should be reconstructed to alleviate the overlap (Fig. 4(3)). In this process, the tentative route data preserved in advance is used.

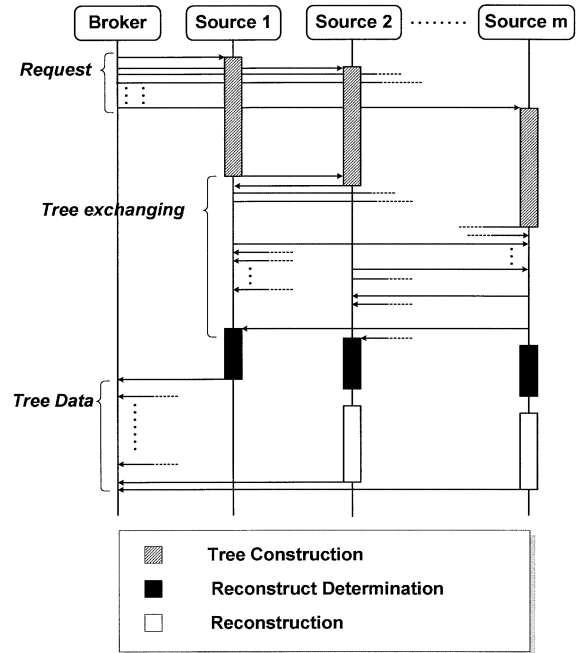


Fig. 5 Multipoint-to-multipoint routing sequence diagram.

To make sure when and where each message, such as multicast tree data and multicast request, is sent/received in the process of the proposed method, we show the time series transaction trace of the proposed method in Fig. 5.

In the figure, gray boxes correspond to “tree construction,” black boxes are “reconstruction determination,” and white boxes are “reconstruction,” respectively. The arrows represent the timing of, and show the direction in/from which each message is sent/received.

At first, when the multicast broker receives a multicast request, it sends a request message to the corresponding sources to construct multicast trees. Every requested source constructs a multicast tree from the current network state, and then exchanges the constructed tree data with the other multicast sources. Each source then reconstructs its own multicast tree as appropriate according to the traffic concentration and returns it to the multicast broker. This example shows the sequence in which “Source 1” returns its tree data to “Broker” without reconstruction, but “Source 2” returns its tree data after reconstructing it.

In the following subsections, we give a detailed explanation of each step.

2.4.2 Individual Multicast Tree Construction

This process uses a Prenode table that preserves available data, predecessor nodes of each node on routes from the source node, produced using Dijkstra’s algorithm. The proposed method, moreover, also preserves relevant predecessor nodes as candidates in finding the same hop route. After Dijkstra’s algorithm is finished, Fallback+ generates candidate routes by backward tracing from each destination node to the source node in this table. Correspondingly, the pro-

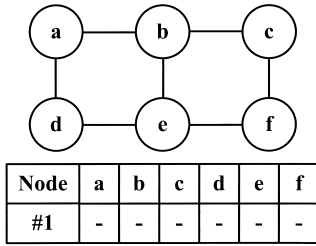


Fig. 6 Example graph and initial Prenode table.

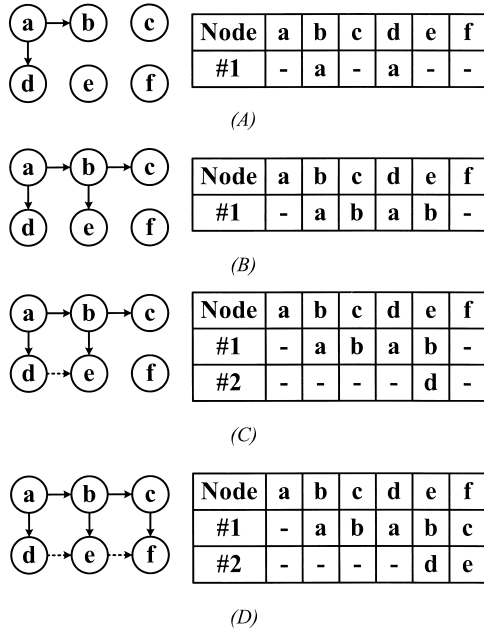


Fig. 7 Tree construction.

posed method generates a multicast tree from the source node *s* to the destination nodes by the same procedure. Note that the proposed method stores unused data for later tree reconstruction.

We demonstrate the construction of a multicast tree using Fig. 6. Here let us consider a multicast tree from the source node 'a' to destination nodes {b, f} generated using the objective function $h(e)$.

In the Prenode table, "#n" indicates predecessor node data and increasing n shows newer data.

Initially, one-hop reachable routes from the source node 'a' are selected, so that the routes from node 'a' to nodes {b, d} are selected in a specific order and the Prenode table is then updated as shown in Fig. 7(A). Note that the graphs in Fig. 7 show the Prenode table graphic. In these graphs, the solid lines indicate the routes found first and the dashed lines indicate those found later.

Next, one of the one-hop reachable routes from the minimum hop route among the existing routes is selected. In this example, the route from node 'a' to node 'b' is selected as the minimum hop route, so that the routes from node 'b' to node {c, e} are selected and then Prenode table is updated as shown in Fig. 7(B). With respect to another minimum hop

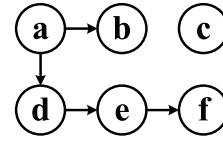


Fig. 8 Tree construction-complete.

route from 'a' to node 'd,' the route from node 'd' to node 'e' is found but it is discarded according to the normal Dijkstra's algorithm because the same hop route from 'a' to node 'e' has already been found. The proposed method, however, maintains it and then the Prenode table is updated as shown in Fig. 7(C). The same procedure is repeated until all routes are found.

Finally, the Prenode table is as shown in Fig. 7(D) and then a multicast tree as shown in Fig. 8 is constructed with the latest predecessor node of each node in the table.

Note that unused data, such as the predecessor node 'b' of node 'e' and the predecessor node 'c' of node 'f,' are preserved for later reconstruction.

2.4.3 Reconstruction Determination

After the individual multicast tree constructions and when all source nodes in G_{MC} have discovered their own trees, each node must decide whether reconstruction is necessary. Here, we introduce an evaluation function $f(m_l)$ and a threshold ($=th$). The function $f(m_l)$ denotes the number of overlapped trees on the link l . It is formulated as

$$f(m_l) = \frac{m_l}{M}, \quad (1)$$

where m_l is the number of trees that uses the link l . When $f(m_l) \leq th$, the reconstruction process is skipped. Otherwise, the reconstruction process reconstructs trees as many as $(m_l - M \times th)$. Here, the proposed method assigns a unique number to each source node beforehand, that enables each applicable node to determine the necessity of reconstruction autonomously without negotiation. In concrete terms, the reconstruction process runs in the pre-determined order of the assigned number.

2.4.4 Reconstruction

Some source nodes reconstruct their own multicast tree using the predecessor node data preserved in the course of the first tree construction process. We illustrate an execution example.

Here let us consider $G_{MC} = \{a, d, f\}$ and a multicast from node 'a' to the destination nodes {d, f}, which must be reconstructed. The node 'a' computes the Prenode table as shown in Fig. 7(D) during the individual multicast tree construction.

At first, node 'a' constructs the tree 'a' → 'd' → 'e' → 'f' as shown in Fig. 9(A) with the latest predecessor of each node in Prenode table.

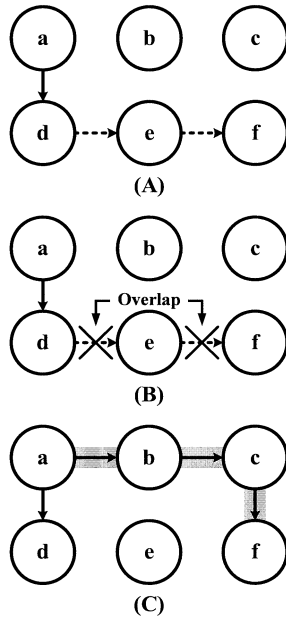


Fig. 9 Multicast trees.

Suppose that node 'a' assesses that there are two overlapped links ('d' → 'e', 'e' → 'f') between its own tree in Fig. 9(B) and the tree of node 'd' after exchanging tree data and $f(m_{\{d\} \rightarrow \{e\}}), f(m_{\{e\} \rightarrow \{f\}}) > th$, Node 'a' must then reconstruct its own tree. In this case, node 'a' can select a route 'a' → 'b' → 'c' → 'f' for node 'f' in Fig. 9(C) instead of the previous one by referring to the Prenode table. As a result, node 'a' reconstructs and derives the new tree 'a' → 'b' → 'c' → 'f', 'a' → 'd'. This tree can bypass the overlapped links.

Note that if there is no route that causes decline in overlapping of links, this process is finished and no further reconstructions are executed.

2.5 Other Multicast Functions

We show some functions necessary for implementing our proposal simply on real networks.

2.5.1 Member Join/Leave

On joining a multicast group, a new source node receives the tree data for the multicast group from the multicast broker (MB) by sending a "join" message. With the tree data of the other multicast source nodes and the current network state, a new source node constructs its own multicast tree and then by sending the result of tree construction lets MB modify the multicast state.

On the other hand, when a participating source leaves the multicast group, it sends a "leave" message to let MB delete its tree and prune unnecessary branches from the trees of the other multicast source nodes. Note that, if the source transmits data to other participating sources, branches for it are not pruned.

2.5.2 Connection Management

Generally, connection management uses the ACK/NAK approach to keep a connection stable and ensure reliability. In multicast communication, however, that approach can lead to the ACK/NAK implosion problem.

In this paper, we assume that multipoint-to-multipoint communication services, including video conferencing and distance learning, provide live streaming data, in which the ACK/NAK approach is not used. If connection management is necessary, it is up to the application or an ACK less approach such as RTP control protocol could be used.

3. Performance Evaluation

3.1 Time Complexity Analysis

In this subsection, we evaluate the time complexity of the proposed method. Let L denote the number of links connected to a node.

Tree Construction

This process requires $O(PV \log V)$ time complexity in the worst case, where P denotes the number of metrics. This is because Fallback+ [22] is adopted to construct a multicast tree.

Reconstruction Determination

The number of multicast trees used by a multicast group is M and each multicast tree includes $V - 1$ links in the worst case, so that this process requires $O(M(V - 1))$ time complexity to know which links are overlapped and determine whether reconstruction is necessary.

Reconstruction

The maximum length of each route is $(V - 1)$ and the maximum number of node data entries for each node is L . Therefore this process requires $O((M - 1)L(V - 1))$ time complexity.

Exchanging tree data between all source nodes requires $O(M)$, as well as the processes mentioned above.

Thus, the total time complexity of the proposed method is

$$O(PV \log V + M(V - 1) + (M - 1)L(V - 1) + M). \quad (2)$$

Although P has little effect on the time complexity because it is far smaller than V , M generally increases in proportion to V and L is equal to V in the worst case, so that the time complexity might come close to $O(V^3)$.

When it is assumed that L is independent of V but several dozens at most, the proposed method has a reasonable solution time in a deterministic polynomial time complexity $O(V^2)$ and is suitable for practical use.

3.2 Simulation Experiments

To examine the effectiveness of the proposed method, we

have evaluated its performance through simulation experiments.

3.2.1 Simulation Models

We generated a random graph using Waxman's method [23] and selected multicast members (nodes) randomly. Each graph is composed of 100 nodes and the average bi-directional node degree is four. Each multicast group has 10 source nodes.

3.2.2 Comparison Algorithms

We evaluated the Parallel-method and the Sequential-method for comparison.

As mentioned above, the Parallel-method is faster than the Sequential-method, but it might cause traffic concentrations because each multicast tree is constructed separately without consideration of which links other multicast trees are using. With the Sequential-method, ex-ante results help to construct a better multicast tree from the point of view of traffic load balance. The computation time, however, is sensitive to the number of the source nodes and much more network resources are consumed.

In the following simulations, we assume that the proposed method and comparison methods adopt Fallback+ in tree construction so that they can construct a multicast tree with multiple QoS requirements.

3.2.3 Performance Measures

We introduce two performance measures: Overlap Level (OL) and Used Links (UL). OL denotes the number of trees that use a link and is defined as

$$OL(l) = \sum_{g_m \in G_{MC}} \Phi_{g_m}(l)$$

where

$$\Phi_{g_m}(l) = \begin{cases} 1 & l \in T_{g_m} \\ 0 & \text{otherwise.} \end{cases}$$

We illustrate this example as follows. In Fig. 10, there are three multicast trees $T_B = \{l_{B \rightarrow A}, l_{A \rightarrow F}, l_{B \rightarrow C}\}$, $T_F = \{l_{F \rightarrow E}, l_{E \rightarrow B}, l_{B \rightarrow C}\}$, $T_C = \{l_{C \rightarrow D}, l_{D \rightarrow E}, l_{E \rightarrow F}, l_{E \rightarrow B}\}$. In this example, the OL value for $l_{B \rightarrow C}$ is two and that for $l_{A \rightarrow F}$ is one.

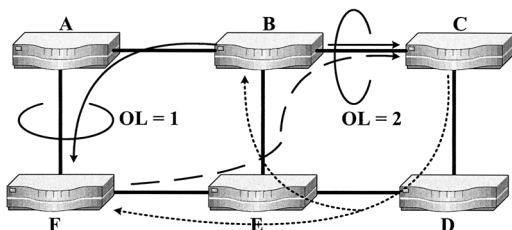


Fig. 10 OL examples.

A high OL value means increased traffic concentration. In other words, a link with a high OL value might be a bottleneck for further connections.

On the other hand, UL denotes the number of cumulative links used by a multicast group and is defined as $UL = |T_{G_{MC}}|$. A high UL indicates wasteful use of network resources

3.2.4 Traffic Load Balance

We first have studied the load balance of multicast traffic for each method. Figure 11 shows the average maximum OL with 95% confidence interval for each multicast group with each method. Figure 12 shows the number of links used by a multicast group that have the indicated values of OL. The number of links is normalized by the results of the Parallel-method. As for the proposed method in these graphs, the results are plotted with respect to different thresholds ($th = 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$). In these two graphs, it can be seen that the result of the Sequential-method is almost the ideal value with regard to the traffic load balance.

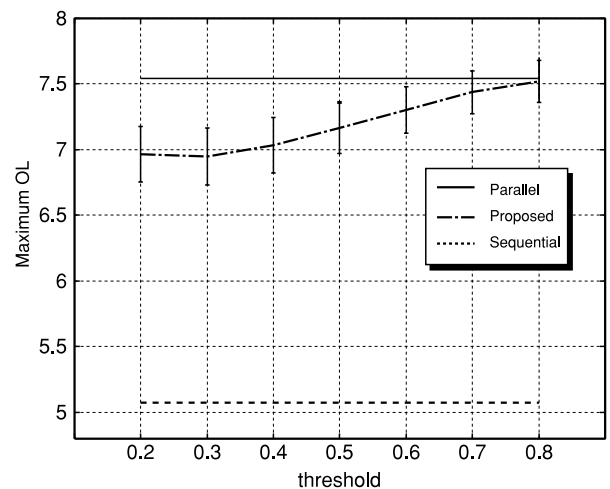


Fig. 11 Maximum overlap level per multicast group.

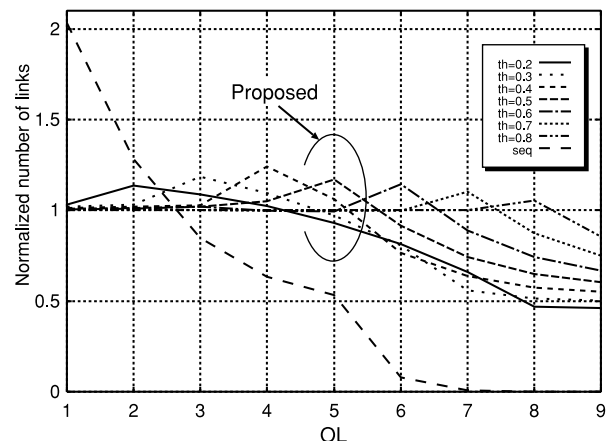


Fig. 12 Normalized number of links by OL.

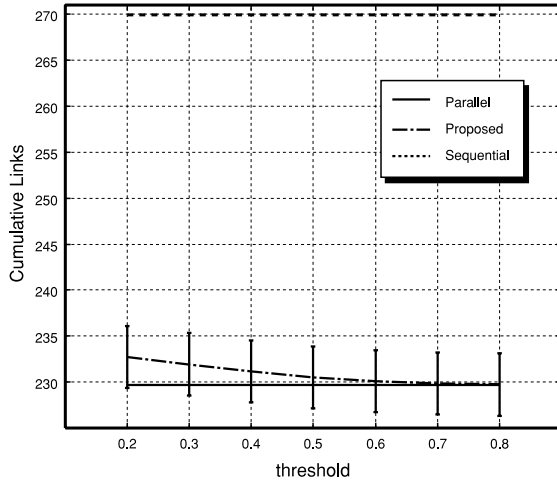


Fig. 13 Cumulative used links.

It is apparent from Fig. 11 that the result of the proposed method for high threshold is almost same as that of the Parallel-method, but the results for lower thresholds ($th \leq 0.4$) show a reduced maximum OL. This is because the proposed method steadily reduces the number of heavily congested ($OL = 7 \cdots 9$) links, as indicated by Fig. 12.

3.2.5 Network Utilization

We have also studied the consumption of network resources. Figure 13 shows the mean UL for each method. From this graph, we see that the mean UL of the proposed method seems to increase with reducing *threshold*, but the value for the Parallel-method is within the confidence intervals of that for the proposed method. As a result, the proposed method consumes about the same number of links as the Parallel-method. On the other hand, the Sequential-method consumes many more links to achieve traffic load balance.

From these two results, the *threshold* should be set to a relatively small value to take full advantage of the proposed method, although we should do more simulations using large numbers of sources.

3.2.6 Blocking Probability

We have compared the blocking probability of the proposed method to that of the Parallel-method to investigate its practical effectiveness. Suppose that each link has 200 Mbps bandwidth. The connection requests arrive according to a Poisson arrival process with rate λ and each multicast group with 10 sources is selected randomly. Once a connection is established, each connection holds 10 Mbps bandwidth on each link along the multicast tree for an exponentially distributed time with fixed unit mean. Figure 14 shows the blocking probability as a function of arrival rate λ .

The results show that the proposed method outperforms the Parallel-method. When the blocking probability is around 1%, the proposed method can offer services for about 1.2 times more requests than the Parallel-method.

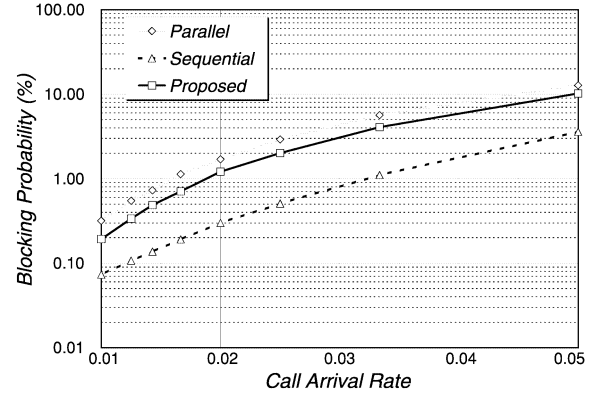


Fig. 14 Blocking probability.

Table 1 Computation time for each process.

process	threshold						
	0.2	0.3	0.4	0.5	0.6	0.7	0.8
TC	3.16						
RD	2.02	1.95	1.86	1.81	1.79	1.78	1.78
RC	0.115	0.067	0.060	0.069	0.048	0.036	—

(ms)

3.2.7 Computation Time

We have also worked out the computation time of each process. This simulation was done on a PC with 1GHz Pentium III and 1GB memory. Table 1 shows computation times for each process. In the table, “—” denotes that the result is nearly equal to zero because “reconstruction” is rarely executed for $th = 0.8$. “TC” stands for Tree Construction, “RD” is Reconstruction determination, “RC” is Reconstruction, respectively.

From the table, it is clear that the computation time of “reconstruction” is much less than that of “tree construction.” This is because tentative routes are used in reconstructing. Moreover, the sum of the computation time of “reconstruction determination” and that of “reconstruction” is still less than the computation time of tree construction.

The computation time of the parallel method is equal to that of “tree construction” and that of the sequential method is about ten times more than that of “tree construction.”

Note here that the results are in which the number of nodes is 100 and the number of sources a group is 10, moreover, we can use full CPU resource to execute. When the computation time for each process increases with increasing the number of nodes or sources, however, this relationship between the proposed method and the existing method is kept. Thus, in real networks, the absolute values of computation time and its difference may become larger.

In the simulation, the communication delay for the exchanging of the tree data is ignored. However, even if the assumed network has the same bandwidth as the current network, it takes just several dozens of milliseconds to communicate because the size of the tree data is at most several KB. Therefore, the total computation time of the proposed

method is practical.

4. Conclusion

In this paper, we have proposed a new routing method for multipoint-to-multipoint communications. It constructs a multicast tree for each source node using Fallback+ to satisfy multiple QoS requirements. Note that each tree is constructed in parallel. Subsequently, it allows all source nodes to exchange their tree data to detect any overlapping links, and reconstructs the multicast tree with the preserved tentative route data to alleviate traffic concentrations. Performance evaluation results show that the method has a practical time complexity and can achieve good traffic load balance and efficient use of network resources compared with existing methods.

References

- [1] J.M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol.14, pp.95–116, 1984.
- [2] Y.P. Aneja and K.P.K. Nair, "The constrained shortest path problem," *Networks*, vol.13, pp.549–555, 1983.
- [3] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol.17, pp.36–42, 1992.
- [4] H.C. Jochsch, "The shortest route problem with constraints," *Journal of Mathematical Analysis and Applications*, vol.14, pp.191–197, 1966.
- [5] W.C. Lee, M.G. Hluchyi, and P.A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Netw.*, vol.9, pp.46–55, July 1995.
- [6] C. Pornavalai, G. Chankraborty, and N. Shiratori, "QoS routing algorithm for multimedia communication," *IEICE Technical Report*, IN97-30, 1997.
- [7] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol.14, no.7, pp.1228–1233, 1996.
- [8] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," *INFOCOM2001*, vol.2, pp.834–843, Alaska, 2001.
- [9] G. Feng and C. Douligeris, "An efficient approximate algorithm for finding paths with two additive constraints," *IEICE Trans. Commun.*, vol.E85-B, no.6, pp.1143–1151, June 2002.
- [10] B. Wang and J.C. Hou, "Multicast routing and its QoS extension: Problems, algorithms, and protocols," *IEEE Netw.*, vol.14, pp.22–35, Jan./Feb. 2000.
- [11] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta Inform.*, vol.15, pp.141–145, 1981.
- [12] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "Multicast routing for multimedia communication," *IEEE J. Sel. Areas Commun.*, vol.1, no.3, pp.286–292, 1993.
- [13] Q. Sun and H. Langendörfer, "Efficient multicast routing for delay-sensitive applications," *Applications, in Second International Workshop on Protocols for Multimedia Systems (PROMS'95)*, pp.452–458, 1995.
- [14] E. Gelenbe, A. Ghanwai, and V. Srinivasan, "Improved neural heuristics for multicast routing," *IEEE J. Sel. Areas Commun.*, vol.15, no.9, pp.147–155, 1997.
- [15] H.F. Salama, D.S. Reeves, and Y. Viniotis, "Evaluation of multicast routing algorithm for real-time communication on high-speed networks," *IEEE J. Sel. Areas Commun.*, vol.15, no.3, pp.332–345, 1997.
- [16] G.N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE J. Sel. Areas Commun.*, vol.15, no.3, pp.346–355, 1997.
- [17] N.F. Maxemchuk, "Video distribution on multicast networks," *IEEE J. Sel. Areas Commun.*, vol.15, no.3, pp.357–372, 1997.
- [18] A. Shaikh and K. Shin, "Destination-driven routing for low-cost multicast," *IEEE J. Sel. Areas Commun.*, vol.15, no.3, pp.373–381, 1997.
- [19] M. Parsa, Q. Zhu, and J.J. Garcia-Luna-Aceves, "An iterative algorithm for delay-constrained minimum-cost multicasting," *IEEE/ACM Trans. Netw.*, vol.6, no.4, pp.461–474, 1998.
- [20] R. Sriram, G. Manimaran, and C.S.R. Murthy, "Algorithms for delay-constrained low-cost multicast tree construction," *Computer Communications*, vol.21, no.18, pp.1693–1706, Dec. 1998.
- [21] C.P. Low and N. Wang, "On finding feasible solutions for the group multicast routing problem," *IEICE Trans. Commun.*, vol.E85-B, no.1, pp.268–277, Jan. 2002.
- [22] H. Tanioka, K. Kinoshita, T. Takine, and K. Murakami, "Fallback+: A routing algorithm subject to multiple QoS constraints," *Proc. 1998 International Technical Conference on Circuits/Systems, Computers and Communications*, vol.1, pp.319–322, July 1998.
- [23] B.M. Waxman, "Routing of multipoint connection," *IEEE J. Sel. Areas Commun.*, vol.6, pp.1617–1621, 1988.



Hideaki Tanioka received his B.E. and M.E. from Osaka University in 1999 and 2001 respectively. He is currently a D.E. candidate in the Department of Information Systems Engineering, Graduate School of Engineering, Osaka University.



Kazuhiko Kinoshita was born in Osaka, Japan, on June 13, 1973. He received the B.E., M.E. and D.E. degrees in information systems engineering from Osaka University, Osaka, Japan, in 1996, 1997 and 2003, respectively. From April 1998 to March 2002, he was an Assistant Professor of the Department of Information Systems Engineering, Graduate School of Engineering, Osaka University. Since April 2002, he has been an Assistant Professor of the Department of Information Networking, Graduate School of Information Science and Technology, Osaka University. His research interests include personalized multimedia communication networks and agent communications. Dr. Kinoshita is a member of IEEE.



Koso Murakami received the B.E., M.E. and Ph.D. degrees from Osaka University, Osaka, Japan in 1971, 1973 and 1991, respectively. From 1973 to 1995, he was with Fujitsu Laboratories Ltd., Kawasaki, Japan, engaged in research and development of digital switching systems, Asynchronous Transfer Mode switching systems and photonic switching technologies. From 1990 to 1995, he was a Manager of the Communications Network Systems Laboratory of the same company and was responsible

for research and development of broadband ISDN, Intelligent network, Telecommunications Management Network and personal communication networks. In April 1995, he joined Osaka University and was a Professor of the Computation Center of that university from 1995 to 1997. From 1998 to 2001, he was a Professor in the department of Information Systems Engineering of Osaka University. Since April 2002, he has been a Professor in the department of Information Networking in Graduate School of Information Science and Technology, newly established in Osaka University. Since April 2001, he has concurrently served as Director of the Collaborative Research Center for Advanced Science and Technology of Osaka University. His research interests extends to ultra high-speed networks and multimedia information networking architecture. Prof. Murakami is a senior member of IEEE and a member of IPSJ.