INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2003; **13**: 321–335 (DOI: 10.1002/nem.485)

A dynamic multicast routing satisfying multiple QoS constraints

By Debasish Chakraborty, Goutam Chakraborty and Norio Shiratori*

In this paper we propose a QoS-based routing algorithm for dynamic multicasting. The complexity of the problem can be reduced to a simple shortest path problem by applying a Weighted Fair Queuing (WFQ) service discipline. Using a modified Bellman–Ford algorithm, the proposed routing builds a multicast tree, where a node is added to the existing multicast tree without re-routing and satisfying QoS constraints. With user defined life-time of connection this heuristic algorithm builds multicast tree which is near optimum over the whole duration of session. Simulation results show that tree costs are nearly as good as other dynamic multicast routings that does not consider QoS. Copyright © 2003 John Wiley & Sons, Ltd.

Introduction

With recent developments in transmission and computing technologies, distributed multimedia applications, such as video conferencing and video on demand have now become possible and will be widely used in the near future. Such applications transmit information which usually have different traffic characteristics and various QoS performance requirements, similar to real-time communication. The received data is meaningless, if some of the QoS constraints are not satisfied. To achieve performance guarantees it has to achieve resource reservation and excise network control.[4,32]

Usually a connection-oriented route is preferred because of its simplicity for reserving resources. The route will be fixed before connecting the call and will be continued until the end of the session. Thus, while finding the route, not only must all of the required QoS constraints be satisfied, but also it is important to allocate the resources effectively, so that the success of subsequent calls is high. The QoS routing problem has been proved to be NP-complete.[27]

Routing algorithms which are currently employed in networks are either Dijkstra or Bellman–Ford shortest path (SP) algorithms. A routing metric for the SP problem is only a single QoS parameter, usually the number of hops or the average path delay. Though Dijkstra or Bellman–Ford algorithms are optimal and run in polynomial time, they could not find the routes that satisfy multiple QoS constraints such as bandwidth, delay, delay jitter and loss probability, which are required by multimedia applications.

In a packet switch network *multicast* refers to the delivery of packets from a single source (*sender*) to multiple destinations (*receivers*). The receivers are said to be members of a multicast (MC) group and they may be collectively addressed.[26] The group membership is said to be *dynamic* when a receiver may join or leave the group at any instant of time during the session.[15] A multicast routing scheme is

Debasish Chakraborty is employed on JGN R&D Project, Telecommunication Advancement Organization (TAO), Tohoku University Office, Goutam Chakraborty works in the Department of Software and Information Science, Iwate Prefectural University, Japan, and Norio Shiratori works at the Research Institute of Electrical Communication (RIEC), Graduate School of Information Science, Tohoku University, Japan.

**Correspondence to: D. Chakraborty, TAO Tohoku University Office, Research Institute of Electrical Communication (RIEC), Graduate School of Information Science, Tohoku University, 2-1-1, Katahira, Aobaku, Sendai, Japan 980-8577.*
deba@shiratori.riec.tohoko.ac.jp

responsible for determining the packet delivery path from the sender to all receivers, typically a sub-tree of the network topology (the *multicast tree*). The routing scheme must dynamically reconstruct the delivery tree when membership changes. Heuristics to construct trees of low overall cost have been developed in references 1, 2, 19 and 28.

The minimization of tree cost has traditionally been formulated as the Steiner minimal tree (SMT)[9,29] problem. The SMT problem has been well studied from a purely graph theoretical point of view and a good survey can be found in references 29 and 16. The existing optimal solutions and many of the heuristics are impractical for use on Internet. We have proposed an algorithm, efficient and suitable for practical applications, for forming an optimum MC tree from a selected parent node joining every newly requested participant nodes in a MC session. While joining the existing MC tree, our algorithm will try to find a suitable route that can satisfy the various QoS requirements of that application, with a least-cost path, if such a path exists.

Different QoS based routing problems are discussed in references 11, 12 and 30. In Parekh's paper[22] it is shown that when the Weighted Fair Queuing (WFQ) service discipline is used, we can formulate an upper bound for end-to-end delay, bandwidth, delay jitter and buffer space requirements for loss-free transmission. For the point-to-point QoS-based routing problem it has been used in references 5 and 20 to reduce complexity. In the proposed algorithm we used a similar approach to take care of all QoS requirements. In addition, depending on the duration of connection for different participants, we modify the cost of the links constituting the multicast tree. This is to encourage new connection paths to use existing links of the multicast tree prudently and thereby optimize resource utilization.

We abbreviate our algorithm for the **M**ultiple **Q**oS constrained **D**ynamic **M**ulticast **R**outing algorithm as *MQ-DMR*. *MQ-DMR* can successfully find the route that can satisfy required QoS constraints, if such a path exists. At the same time, it could effectively distribute traffic throughout the network while adding a new node to the existing multicast tree. This is because we search for a path with low cost, where cost is considered as the inverse of available bandwidth. Our previous work on this issue has been published in reference 7.

The rest of the paper is organized as follows. In the next section we define the dynamic multicast routing and QoS requirements. In the third section we present how the complexity of QoS routing can be reduced by using expressions derived from service discipline. In the fourth section we describe the proposed algorithm. In the fifth section the simulation setup is described and comparisons of results with other algorithms are reported and discussed. Concluding remarks are given in the sixth section. In the Appendix, the pseudocode for our algorithm is given.

In many practical applications participants in the multicast (MC) group change.

Background

—Dynamic Multicast Groups—

In many practical applications participants in the multicast (MC) group change. During running of a session new receivers join or old ones leave. A MC algorithm should be able to allow for changes in the MC group without disrupting communications between the source and the existing members of the MC group.[8] Heuristic algorithms such as KMB (Kou, Markowsky, Berman)[19] do not meet this requirement. In this algorithm, any change in the MC group membership would require us to recompute the complete MC tree. Changes in the MC tree would cause disruption to communication for existing members. Though the other two familiar dynamic MC routing, Greedy[28] and Naive algorithms,[8] could successfully add new nodes without any rerouting, none of them can provide any direction to take care of the delay or hop-count or any other QoS requirements of an application.

In the Naive algorithm, a new node joins the multicast group with the shortest path from the source every time. This is the simplest but it produces a very high tree cost. In the Greedy algorithm, the new node joins with the shortest path to the nearest node that is in the multicast tree. One

of the basic problems with the Greedy algorithm is that it does not consider the duration for which a node is connected in the multicast group. So, the route which is optimal at a particular time may become non-optimal after some time. Many nodes that remain connected in the tree have in fact already left the session. They have to stay as Steiner nodes. (The nodes that stay between two active multicast nodes, but do not take active part are called Steiner nodes.) This makes the total tree cost high, though the number of multicast members may have reduced.

We have also implemented the Least-hop-path algorithm. This tries to find the path between a source to a new destination with the shortest hop length. So, it will always satisfy the hop-constraints if there are any, but the resultant tree cost for this algorithm is higher than for other single constrained algorithms, as it is not considering tree link-costs.

—QoS-based Multicast Routing Problem—

We simply assume that a communication network can be modeled as a direct connected graph $G = (V, E)$. Here V represents the set of nodes, which could be routers, servers or switches, and E represents the set of edges or links of the network. $h(e) = h(u, v) = 1$, $c(e) = c(u, v)$, $d(e) = d(u, v)$, $b(e) = b(u, v)$, $j(e) = j(u, v)$, $l(e) = l(u, v)$ are the hop, cost, delay, available bandwidth, jitter and loss functions of link e, connecting directly the nodes u and v. The link cost function is a measure of links load, as we consider $c(e) = \frac{1}{b(e)}$. $S \subseteq V$ as a subset of vertices of V. The QoS constraint dynamic multicast problem is to find a route for a newly joining node, such that the route satisfies multiple QoS requirements, while effectively using the existing links of the MC tree, and distributes the traffic uniformly. This leads to proper resource management and improved success rate for subsequent calls.

In the dynamic version of the multipoint problem, we start with a network and consider a sequence of requests R_i, where each R_i is a duple (d, t). Here $d \in V$ is the node to be added and t is the duration of time for which the connection is requested. The problem is to find a sequence of routes satisfying the QoS constraints connecting the newly joining nodes to the MC tree, such that the overall cost of the tree for the whole duration of the session is minimized.

Here, we have made an important assumption that every connection request is tagged with the required time duration. In fact, recently it was felt that advance resource reservation should come with the required time duration, because blocking a resource for an unknown period could lead to extremely difficult situations for resource management.[13] Moreover, if the network resources has to be paid for, it is very natural to think in this way. Finally, without this duration available at the time of joining it is impossible to design dynamic MC routing, which could optimize the tree cost for the whole duration of the session and also would not use rerouting. The algorithm for the near optimal solution given in reference 19 uses total re-routing every time there is a member change. If the new member is not among the existing Steiner nodes, the configuration of the optimum tree with the new MC member can be totally different from the previous one.

When a new destination node d joins the multicast tree, the problem is to find a route from the source node to the destination node d, $P(s, d) = (s, j, k, \ldots l, d)$, such that (i) it would use the existing multicast links wisely to minimize the increase in traffic in the network and distribute the traffic to balance the load throughout the network by choosing least cost (least loaded) links, and (ii) it would satisfy all the required QoS constraints, from source to destination. How we accomplished these two objectives will be clear later. Let us first discuss the QoS constraints and the optimization objectives.

For the path P we define the following:

$$H(P) = \sum_{e \in P} h(e) \tag{1}$$

$$C(P) = \sum_{e \in P} c(e) \tag{2}$$

$$B(P) = \underset{e \in P}{Minimum}(b(e)) \geq \Delta_{bandwidth} \tag{3}$$

$$D(P) = \sum_{e \in P} d(e) \leq \Delta_{delay} \tag{4}$$

$$J(P) = \sum_{e \in P} j(e) \leq \Delta_{jitter} \tag{5}$$

$$L'(P) = \prod_{e \in P} l'(e) \geq (1 - \Delta_{loss}) \tag{6}$$

Here, $H(P)$, $C(P)$, $B(P)$, $D(P)$, $J(P)$ and $L(P)$ are the number of hops, cost, bottleneck bandwidth, delay, jitter, and loss on the path $P(s, d)$ respectively, where s is the source and d is the destination node. The QoS constraints are symbolized by Δ with respective suffixes. These constraints as defined in equations (3), (4), (5) and (6) are to be satisfied. We don't need to optimize them, i.e. we don't need to find the widest bandwidth path or the path with least delay or delay jitter. The constraints only specify the bounds.

Equations (1) and (2) are usually the optimization functions of QoS-based routing problem. We need to find the path (satisfying QoS constraints), such that total cost along the path is minimized. Hop and cost may sometimes be contradictory as the minimum-hop route will not necessarily minimize the cost. For point-to-point connection, in general, the minimum-hop route should be the primary criterion of route selection. If there is more than one minimum-hop QoS satisfying route, the least cost one is selected. But in the multicast situation, as we need to encourage routes using an existing multicast tree, longer hop routes (still satisfying QoS constraints) with less cost are to be explored. Only then will the new nodes be connected via the existing MC tree.

The loss probability of link e is symbolized as $l(e)$, and therefore the probability of successful transmission on link e is $l'(e) = 1 - l(e)$. The probability of successful transmission for the whole path could then be expressed as in equation (7). We will consider the guaranteed service class of the integrated service model on the Internet[3,25] which is the deterministic guaranteed service. The loss probability would be zero, i.e. we need to ensure that there will be no queuing loss (loss-free constraint). Thus we redefine the loss probability constraint (equation (6)) to be the buffer space constraint. Every node or switch along the selected route must have enough buffer space, where $f(e) = f(u, v)$. $f(e)$ and Δ^e_{buffer} is the amount of available buffer and required buffer space for no queuing loss at node v for the incoming link e from node u respectively.

$$\forall e = (u, v) \in P, \quad f(e) \geq \Delta^e_{buffer} \tag{7}$$

All the above discussions are with respect to point-to-point routing. And the same conditions are valid for routes for source to different destination in multicast routing.

Service Disciplines for QoS-based Routing Algorithm and Complexity Reduction

In this section we briefly describe an overview of the WFQ service discipline, the QoS bounds from which will be used in our routing algorithm. Finding a route that satisfies multiple QoS constraints, as already mentioned, is a NP-complete problem.[27] However, the proof has been done without any assumptions of the dependency of routing on service discipline. For any guaranteed communication, some service discipline has to be employed, and be using it is possible to find some QoS-bound expressions. Thus one important design issue for the QoS-based routing algorithm is to exploit those expressions derived from the service discipline.

The guaranteed service class of the Integrated Service model on the Internet[3,25] is based on Weighted Fair Queuing (WFQ).[22] It can be used to provide a tight upper bound on the end-to-end jitter and ensure that no packet is lost due to non-availability of a buffer at each switch and assuming no failure of network components. Table 1 shows the expressions to calculate such bounds and buffer space requirement (for no queuing loss) of a flow q, for which the traffic characteristic is in the form $(\sigma_q, \rho_q, S^q_{max})$.[32] Here σ_q is the token bucket size or maximum burst size, ρ_q is the token generation rate of the leaky bucket, S^q_{max} is the maximum packet size of the requested flow q respectively. r_q is the guaranteed rate for the connection or the amount of bandwidth to be reserved for the requested flow $q(r_q \geq \rho_q)$. S_{max} is the maximum packet size allowed in the network. $R_i(e)$ and $pd_i(e)$ are the total bandwidth (link capacity) and propagation delay of link e which is the ith hop on the route traversed by the connection. H is the number of hops of the route.

End-to-end jitter bound	$\frac{\sigma_q + HS^q_{max}}{r_q}$
buffer space at n^{th} switch	$\sigma_q + nS^q_{max}$

Table 1. End-to-end delay-jitter bound, and buffer space requirement of the Weighted Fair Queuing (WFQ) service discipline

The QoS routing problem has been proved to be NP-complete.

—Complexity Reduction—

The QoS routing problem has been proved to be NP-complete.[27] However, when there are two QoS constraints to satisfy, which are either additive or multiplicative, and if the value of at least one of them is the same on every link, we can find a QoS-constrained route satisfying both the constraints by the Bellman–Ford algorithm in polynomial time. Bellman–Ford (BF) is a breadth-first search algorithm which proceeds by searching the shortest-path route by increasing the number of the hop-count. This hop-count can be considered as one of the additive constraints where every link has the value of 1. The idea of complexity reduction stems from the fact that it is possible to (i) map *path constraint* to *link constraint*, and (ii) *express QoS constraints in terms of the number of hops*, i.e. the maximum allowed number of hops in the route. We can then map the different constraints to delay constraint and number of hops, and thus reduce the complexity of the problem to that of shortest-path routing. By employing WFQ as the service discipline, we can do the complexity reductions (by using the expressions in Table 1), when QoS constraints are bandwidth, delay, jitter, and loss-free. The complexity reduction procedure is as follows:

- **Delay constraint**: Link delay is composed of queuing, transmission, and propagation delay. Queuing delay of the path is the function of the number of hops H. We define a *link-delay function*, D which assigns a non-negative weight to each link in the network. The value $D(l)$ associated with link $l \in E$ is a measure of the delay that packets experience on that link, including the queuing, transmission and propagation components.

  The delay of each link is different. So, we have to find a path that can satisfy the delay constraint of the application. That means the summation of the total link-delay should be less than or equal to the required delay constraint.

  We will first find H_{max}. Then we will calculate the minimum delay path within H_{max}. If it cannot satisfy the delay requirement, then there is no path. If available, then within the H_{max} hop length, we will calculate both the *delay* and *cost* of those possible paths. These two computations can be done simultaneously. The least-cost path that can satisfy the delay will be selected as the desired route to connect the new node.
- **Bandwidth constraint**: Because bandwidth is a link constraint, we can simply eliminate all links in the network which cannot satisfy the bandwidth constraint. Before running the algorithm, we assign infinite cost to those links.

$$c(e) = \begin{cases} \infty & \text{if } b(e) < \Delta_{bandwidth} \\ c(e) & \text{otherwise} \end{cases} \tag{8}$$

  Thus bandwidth constraint is mapped to link cost.
- **Delay jitter constraint**: Delay jitter is clearly a function of the number of hops of the route (see Table 1). Before running the BF algorithm, the maximum hop path which will still satisfy delay jitter can be calculated as shown in equation (9). By setting this value to be the maximum number of iteration in the BF algorithm, which is the maximum number of hops, we can ensure that the route will have lower delay jitter than the required jitter constraint:

$$H_{max}^{jitter} = \left\lfloor \frac{\Delta_{jitter} \cdot r_q - \sigma_q}{S_{max}^q} \right\rfloor \tag{9}$$

  Thus the delay jitter is mapped to the maximum allowed number of hops in the route.
- **Loss-free or buffer space constraint**: This constraint is a special case when loss probability equals to zero. It can be satisfied if there is enough buffer space available at all nodes along the established route. For WFQ, depending on the position of the particular switch from the source node (henceforth we will use the term hop-count), the required buffer size at that node to ensure no packet loss could be determined from Table 1. We see that the required buffer size increases with

this hop-count. Without loss of generality, we assume that each node has its associated buffer corresponding to every incoming link. We can say that a link e has enough buffer if its hop-count is less than or equal to $m(e)_{max}$ defined by the following equation (10), where $f(e)$ is the buffer size corresponding to link e. Thus, if $m(e)_{max}$ is the maximum allowed hop-count of link e, loss-free transmission is ensured. In the Bellman–Ford algorithm the number of iterations is the same as the number of hops at that particular stage of searching. So the hop-count of every link is same in a particular iteration, and equal to that iteration number. The algorithm would search up to the *ith* iteration only, where $i \leq m(e)_{max}$. We assume that the source node has enough buffer space ($\geq \sigma_q + S^q_{max}$).

$$m(e)_{max} = \left\lfloor \frac{f(e) - \sigma_q}{S^q_{max}} \right\rfloor - 1, \quad e = (u, v) \in E \quad (10)$$

Thus we can map the buffer space constraint to a *link constraint*.

QoS-based Dynamic Multicast (*MQ-DMR*) Routing Algorithm

In the last section we explained how to reduce the complexity of the QoS routing problem by using QoS bound expressions for a WFQ service discipline. There are two distinct objectives of the proposed algorithm: (1) how to use part of the existing MC tree while connecting a new participant, so that the traffic over the network is minimized over the whole session period; (2) how to connect a new node to the source so that QoS requirements are met? Before explaining how the above two objectives are accomplished, we will outline the assumptions we made. Finally we will present a correctness and complexity analysis of the *MQ-DMR* algorithm.

—Assumptions—

We are considering a full-connected, single source, flat network. The cost is different for different links between the nodes in the network, where cost is considered as the reverse of the available bandwidth (mentioned earlier).

Delay of a particular application is fixed, Δ_{delay}, for every participants in the session. We assume that the topology of the network will remain unchanged, and no failure will occur during the MC session.

Every node or switch has its limit of buffer size, which does not change during the session.

We also assume that the delay constraint could be translated to the maximum number of hops. Or in other words the delay per hop is same. This is a common assumption and the basis is already explained in the previous section. Thus, for a given Δ_{delay}, we can calculate the maximum number of hops allowed.

We will assume that a participation request is always accompanied with the duration of connection. If not the duration of connection will be assumed until the end of the session, by default. It is reasonable to assume this in a situation where cost is involved according to the duration of participation.

—Informal Description of *MQ-DMR*—

In addition to satisfying QoS constraints, the aim of the algorithm is to minimize the total traffic in the network for the whole duration of running the MC session. When we need to connect a new destination, we should use the existing MC tree as part of the route from the source, so as to minimize the extra traffic. But if it is done in a greedy way, i.e. to connect the new node to the nearest one in the existing multicast tree, the cost over the whole session period may be high. In the MC tree, there could be many links whose life-time is much shorter than that required by the new participant. If those nodes are used as Steiner nodes, they would be unnecessarily blocked for longer periods resulting in higher cost at the end of the session. We proposed a new way of dynamically assigning cost of a link that incorporates the time duration for which it would be in use, so that at the end of the session the overall cost of the MC tree is minimized. This could only be possible when the participants declare the approximate duration for which they would like to be connected. Details of this cost calculation appear in the subsequent sections.

The basic routing algorithm in *MQ-DMR* is similar to that of the Bellman–Ford (BF) shortest-

path algorithm. We have the specific source node and the existing MC tree. The cost of the links which constitute the MC tree are lower compared to the yet unused links. Among those already in the MC tree, the cost of the links which have a longer life-time is less than those with a shorter life-time. *MQ-DMR*, like the BF algorithm, in the first iteration will find one-hop least-cost paths to all destinations. As the optimization criterion is the minimum cost, the tendency would be to select links already in use in the MC tree, and links with large available band-width. In the second iteration it tries two-hops least-cost paths, and continues to higher iterations with longer hops. The algorithm terminates after H_{max} number of iteration steps are tried. Then the least-cost path from source to destination is selected. Here we always try a higher iteration with a longer hop to find a path with less cost and still satisfying the constraint (delay and delay jitter). This is because a path using the existing links of the MC tree may have a longer hop length, but would be more economical. The way we define the cost of the links, both outside the existing MC tree and inside it, the least-cost route will ensure optimum use of network band-width resources when more of the existing tree is used.

While we go to higher iterations of the BF algorithm, we will neglect those links unsuitable to satisfy QoS requirements, as explained in the previous section. From the allowable delay, Δ_{delay}, we can calculate the maximum number of hops from source to destination, say H_{max}^{delay}. From the delay jitter constraint, Δ_{jitter}, we can have the upper bound of hop count, H_{max}^{jitter}, as already shown in equation (9). For loss-free transmission at any iteration if the buffer size to the particular node for that iteration level is insufficient (as described by equation (10)) then that link is discarded. The cost of those links with insufficient bandwidth are considered as infinity and thereby discarded. Thus the maximum number of iterations, H_{max}, for which the BF algorithm is run, is the minimum of H_{max}^{delay} and H_{max}^{jitter}, and could be easily calculated *a priori*. If the destination could not be reached in H_{max} iterations, we declare that there exists no such route from source to destination satisfying all QoS constraints.

The detail of the joining and deleting of MC nodes is described in the following subsection, where the cost of links constituting the existing MC tree is explained.

—Joining and Deleting Destination Nodes—

Add new node to the tree—The session will begin only with the source node. Connection request from destination node z comes to source node s in the form $Req[z, T_{ter}]$, where T_{ter} is the termination time. The earliest arrived node will be the first to join the source node, by the shortest path (Bellman–Ford algorithm) to form the initial multicast tree. With every link we associate a pair $[w, tag]$, where w is the *link-cost* and *tag* is a time stamp. The effective cost (c) of a link is derived from w and the duration of time the link remains in the multicast tree. The *tag* represents the time the link will have to be connected to the multicast tree. It is further explained later in this section, and in the Appendix.

The departure time of the joining node will be the tag for all the links from the source to the newly joined node. This *tag* represent the remaining time of the links forming the path from the source to the requested node. The tag for links not in the multicast tree is set to *zero*. The tag-values of the nodes and links will be reduced as time passes, unless there is any update due to the establishment of a new route involving them.

When the next new node sends a connection request to the source node, the different link costs of the network, both in the existing multicast tree and outside it, will be calculated according to the new node's remaining time with respect to the existing links' tag-value, at that time instant. Effective cost (c) will be calculated for every link according to the following rule:

Link-cost calculation: Suppose link e with actual cost C_e connects node u and v. If T_{new} is the new node's staying time and T_l is the termination tag of the link between (u, v), the link-cost:

1. between two nodes u and v, forming a link, which is outside the existing multicast tree: $C_e \times (T_{new})$.
2. between two nodes, forming a link in the existing multicast tree, with $T_{new} > T_l$: $C_e \times (T_{new} - T_l)$.
3. between two nodes, forming a link in the existing multicast tree, but $T_{new} < T_l$: considered as zero.

Once a route is found, the time stamps of all intermediate links with fewer termination time stamps than T_{new} would be updated to T_{new}.

Figure 1 shows that, when a new node (node *d*) arrives with a *time-tag* of 7(*seven*) as a staying time, every link of the network will be updated according to the *Link-cost Calculation*, mentioned above. Two values are attached with every link, shown as $\alpha \rightarrow \beta$. The α value is the original cost (w) of the link, and the β value is the *effective cost* calculated according to the *Link-cost Calculation* algorithm, for example the link between *a* and *S* whose time tag value is *zero*, as ($a \rightarrow S$) belongs to 'Category 1' (being an outside link of the existing multicast tree). The original cost of link ($a \rightarrow S$) was '4' and new node's staying time is '7'. So the *effective cost* of link ($a \rightarrow S$) is now ($4 \times (7 - 0) = 28$), as shown in Figure 1. As a link between ($f \rightarrow b$) belongs to 'Category 2', the effective cost of link ($f \rightarrow b$) will be its original cost (2) multiplied by the difference of staying time of that link and new node's staying time, i.e. ($7 - 5 = 2$). Thus the *effective cost* will be ($2 \times (7 - 5) = 4$). Similarly as the link between ($e \rightarrow b$) belongs to 'Category 3', so the β value of that link will be *zero*. This is because the time tag of this link is greater than that of new node's staying time (as $8 > 7$). So the *effective cost* of this link will be the original cost of this link, multiplied by *zero*, which is *zero*. The node *e* and the subsequent link ($e \rightarrow h$) has to stay connected in the tree irrespective of whether node *d* joins through it or not. In *line 10* of the pseudo-algorithm this effective cost is calculated.

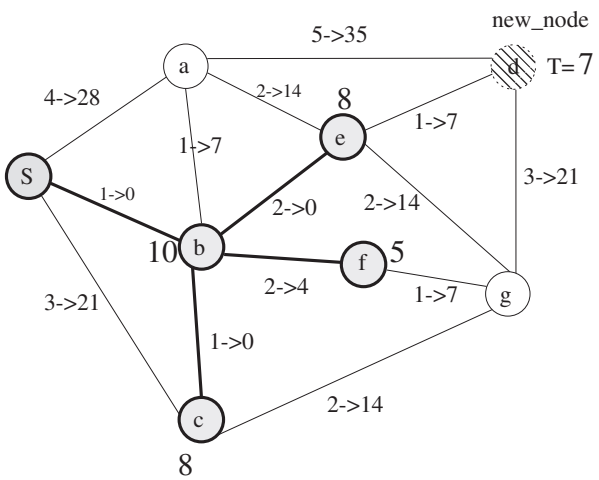

Figure 1. Recalculation of link-weight, at the time of the new node's arrival

With the complexity reduction technique just explained, the path found from source to destination satisfies all QoS constraints. The cost of the path will be minimized, as we will try to find the minimum cost path within the allowable hop-length. The link cost calculation which takes care of the time the link remains connected would produce a nearly optimum minimum cost tree. As we consider the cost as the inverse of available bandwidth, by minimizing cost of the multicast tree we can minimize the overall traffic utilization.

Deletion of nodes—When the departure time of a node is equal to the current global time, that node is supposed to leave the group. The deletion is initiated by *deletion request* from the receiving node.

When the deleted node is a leaf node, it can be deleted instantaneously. Links and nodes in that route can also be deleted recursively, until a node is reached that has a *time-tag* more than the deleted node. In other words, when the *time-tag* value of the nodes and links become zero, they will be deleted. Nodes and links with a *time-tag* value greater than zero will remain in the tree. For any internal node, even when its duration of connection ends, it may not leave, as it may be acting as a Steiner node for other links. Because these nodes' *time-tag* had been updated according to the node attached through it, their *time-tag* value has changed from their initial value.

After the node is deleted from the spanning tree, the corresponding time-tags for those links will also be erased from the tag-table and will get its initial value of zero. For details of node deletion procedure please see lines 32 to 39 of the pseudocode.

—Correctness and Complexity Analysis—

Theorem 1: *When service discipline used in the network is WFQ, and QoS constraints are bandwidth, delay, jitter and loss-free,* MQ-DMR *could always construct a route which satisfies such QoS constraints, if there exists one.*

Proof: The proof is straightforward. Any links which do not have enough bandwidth available

will be eliminated before running the algorithm. A route found by *MQ-DMR* will not contain such unsatisfiable links.

Delay jitter of a route solely depends on the number of hops of the route. The maximum allowable value is computed before running the algorithm using equation (9) to guarantee that the delay jitter is satisfied. *MQ-DMR* will search the path for which the maximum number of hops is limited by the number of iterations (H_{max}). By setting this number less than the restricted maximum number of hops imposed by the delay jitter when the route is found, it would automatically ensure that jitter constraint is satisfied.

The loss-free constraint requires that any switch node in the route needs to have enough buffer space, to ensure no queuing loss of packet at its incoming link. We have already explained how the maximum allowable hop-count ($m(e)_{max}$) could be calculated for every link depending on its available buffer size. Because the Bellman–Ford algorithm searches all possible i hops shortest path from $i-1$ hops shortest path, at the *ith* iteration, any links with $m(e)_{max}$ less than the iteration number will be eliminated. This would ensure loss-free communication.

In our *MQ-DMR*, searching continues until reaching maximum allowable hop-count. If we fail to find one, it is safe to say that no such path exists.

Thus *MQ-DMR* can find a route that satisfies bandwidth, delay, delay jitter and buffer space constraints, if there exists one, for WFQ service discipline.

Theorem 2: *The worst case computation time of* MQ-DMR *is the same as original Bellman–Ford algorithm, i.e. of the order of* $O(|E| \cdot H_{max})$, *where* $|E|$ *is the number of links in the network, and* H_{max} *is the maximum number of iterations of the Bellman–Ford algorithm.*

Simulation Results and Analysis

The communication network used in our experiments are optical fiber, full duplex and directed, with homogeneous link capacity bandwidth of 155.52 Mbps (OC3). Random networks of 100 nodes are generated by modifying the program described in reference 28. Nodes of all these networks were so connected that the degree of each node is at least 2, and the average node degree is 4, which is close to the average node degree of current networks.[24] In every experiment, we initialize the background traffic in each link with random values from 5 to 150 Mbps.

Video traffic was simulated with a flow specification of $(\sigma_{video}, \rho_{video}, S_{max}^{video}) = (10S_{max}, 1.5\,\text{Mbps}, S_{max})$, where S_{max} is equal to the size of an ATM cell, 53 bytes. In our simulations, for every routing algorithm, we restrict the bandwidth reservation to 1.5 Mbps, which is the token rate of the requested traffic. The buffer space for each incoming link at a node is randomly selected between 4.6 kb and 128 kb. Thus with this flow specification and initialization, all the links will satisfy the bandwidth constraint. Any node could be chosen as the source node.

In each network, we select node 1 as the source node. The number of destination nodes is varied. For a fixed number of nodes in a network a different set of destination nodes has been used, using different arrival and departure times for those destination nodes.

We have implemented Greedy[28] and Naive multicast routing[8], and *DMG*[6] and the simple Least-Hop algorithm for comparison. *DMG* works similar to the proposed *MQ-DMR*, but without satisfying QoS constraints and searches the shortest path based on Dijkstra's shortest-path algorithm.

We have shown the simulation results of 100 nodes network here. Comparison has been shown with other dynamic multicast routing algorithms, along with our previously proposed algorithm *DMG*.[6] Though all these three algorithms are unconstrained algorithms, the results shows that *MQ-DMR* performance is still comparable, even after satisfying different QoS constraints. Obviously the performance of *MQ-DMR* depends upon the strictness of the different QoS requirements. A very relaxed QoS constraint application MQ-DMR will perform as well as DMG and better than other unconstrained algorithms.

—Average Tree Cost—

The average tree cost for the whole duration of a session has been observed with several different joining-and-leaving request sets, with different delay constraints (in msec). Figures 2 and 3 show

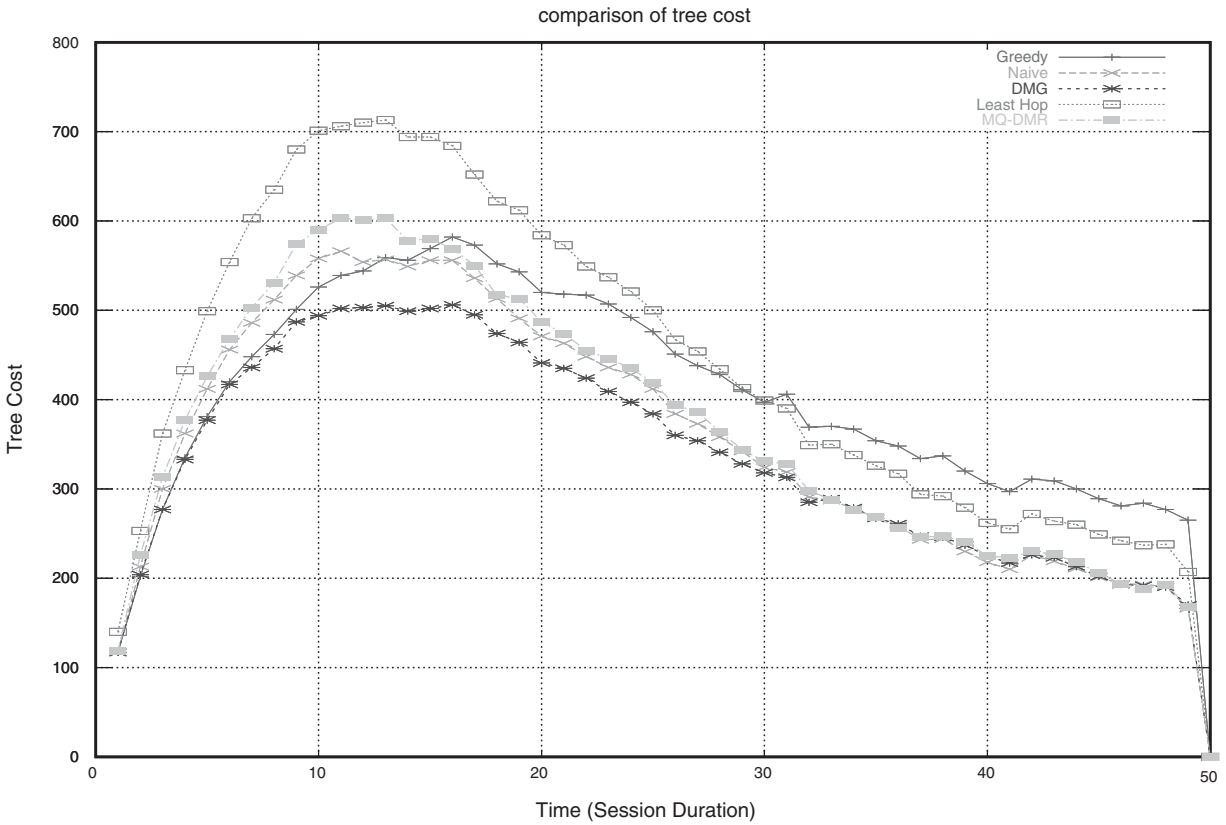

Figure 2. 100 nodes random net, with maximum possible number of destinations set at 30 nodes

tree cost per time instant and their corresponding cumulative value for a 100-node network. Here the delay/hop is considered constant with a value of 2 *ms*. Different values of delay constraint from 8 ms to 14 ms are used, and the corresponding tree costs during the session were simulated.

We have only considered those situations when the rejection rate is zero, meaning every request of connection to the existing multicast tree is possible. If we increase the allowable delay further to such a level that no more relaxation is possible, then *MQ-DMR* will perform as well as *DMG*. At that stage it is not possible to minimize the tree cost with further relaxation of delay constraint or increasing hops. There resultant cost of MQ-DMR in Figures 2 and 3 is higher than most other unconstrained algorithms. This is because the strictness which has been used for this particular experiment restricts the allowable hop length of the MQ-DMR algorithm. Any further relaxation will reduce the tree cost, but may violate the pre-defined QoS constraints.

The success rate, i.e., how often the algorithm could find a route that can satisfy all the QoS constraints, is shown in Figure 5. Keeping the delay jitter constraint the same, we have observed the success rate varying the delay constraint and for different network sizes.

It is obvious that the stricter the constraints, the lower will be the success rate.

It is obvious that the stricter the constraints, the lower will be the success rate. When the QoS constraints are relaxed, at some point the success rate reaches 100%, meaning it could always find a path

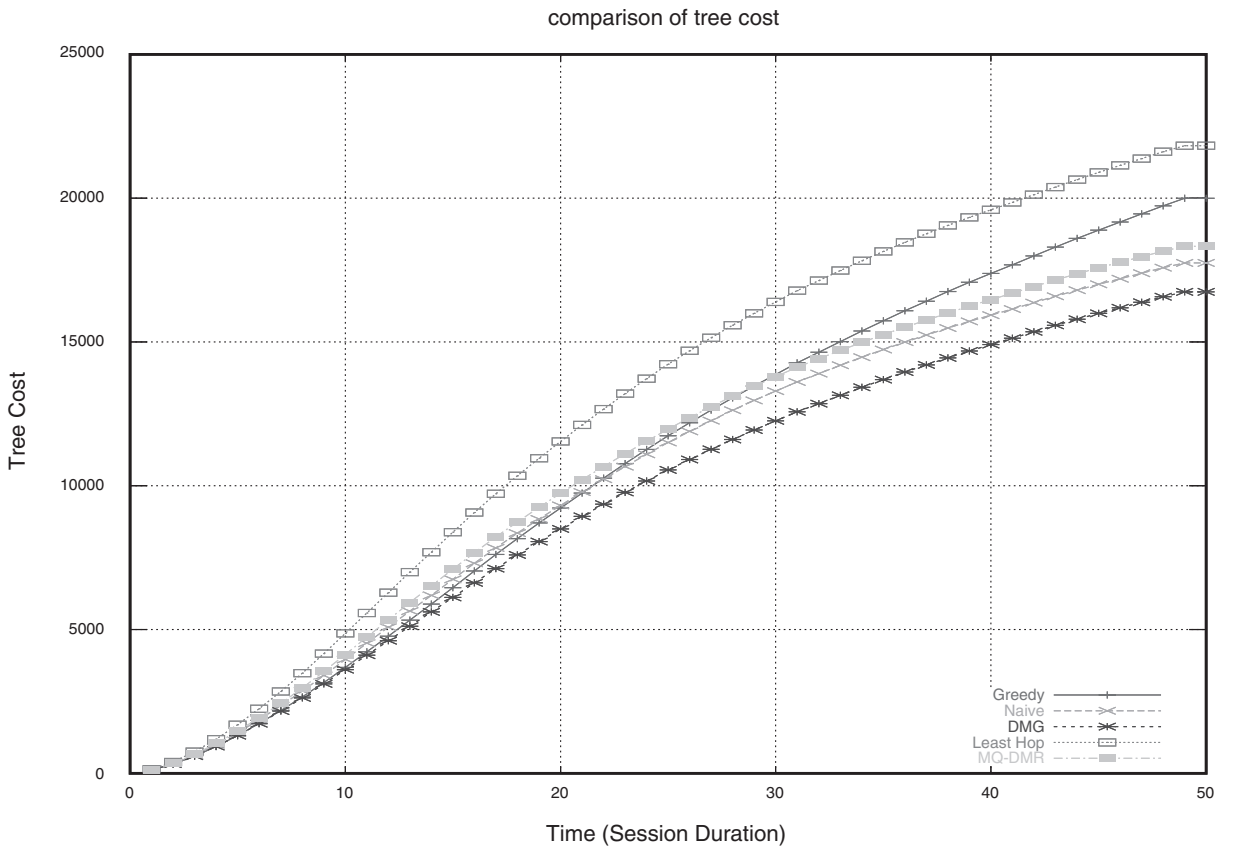

Figure 3. Corresponding cumulative tree cost

that satisfies multiple QoS constraints. As is obvious, this delay constraint value is lower for smaller networks. For a 25-nodes network it is 8 *ms*, wheres for a 100-node network the delay value is 12 *ms*. Any further relaxation of the delay constraint will gradually reduce the cost of the tree, as the algorithm will then try to find a less costlier path with a longer hop length, if available.

—Average Number of Hops—

These experiments were done with the same network topology but with different sets of arrival and departure nodes and their corresponding duration of staying time. We have considered only those situations, for *MQ-DMR*, when there is no rejection of connection request due to unavailability of route, satisfying multiple QoS. Figure 6 shows the comparison of mean hop-length per destination node for different algorithms. For the constrained algorithm *MQ-DMR* both strict and relaxed delay constraint situations were simulated.

Simulation results shows that *MQ-DMR* performs quite close to Shortest Path or Naive multicast algorithms. It actually produces fewer hop-length paths from source to an individual destination, in case the delay constraint is very strict. In those cases, the total cumulative cost becomes a little higher. Even when the delay constraint (i.e. the allowable maximum hops) is relaxed, and thus facilitates searching for a much less costly tree for the duration of the session, the individual hop distance from source to destination does not increase greatly. On the average hop distance remains almost the same as that of the shortest path (SP) algorithm.

Table 2 shows the average number of hop length when hop-length is measured at the instant of con-

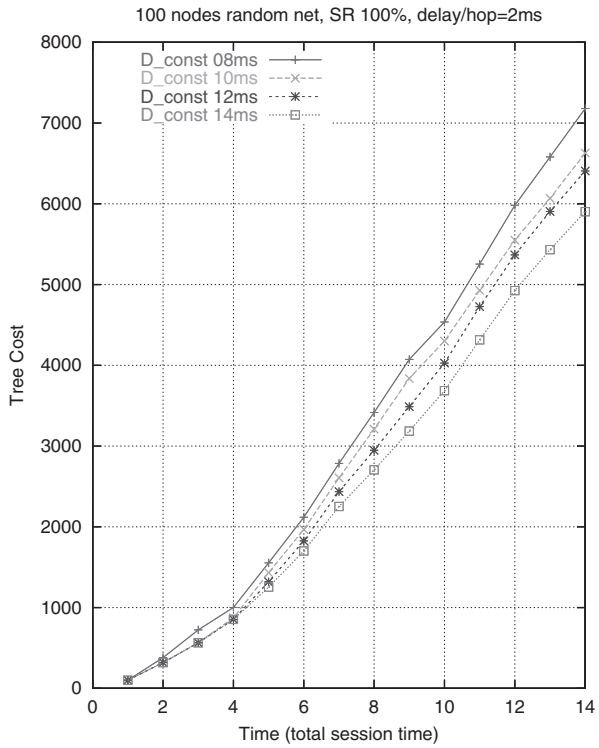

Figure 4. Comparison of cumulative cost for different delay constraint

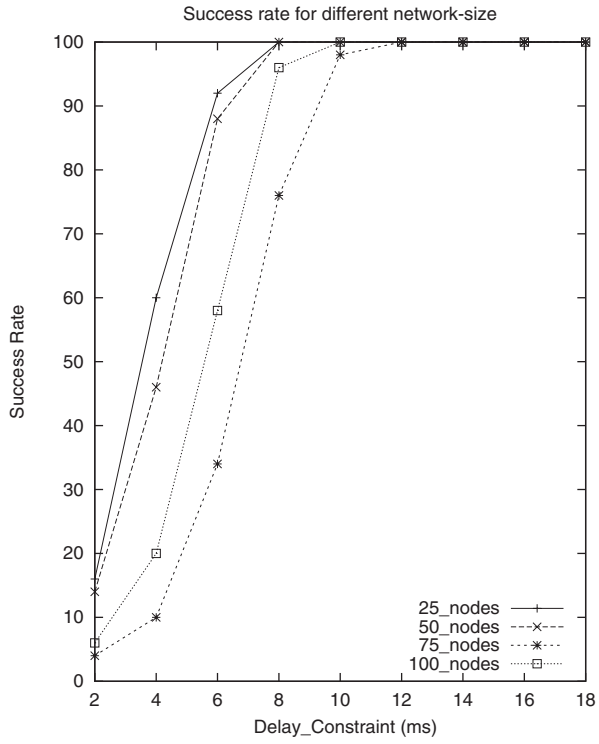

Figure 5. Comparison of success ate with varying delay constraint for networks with different sizes

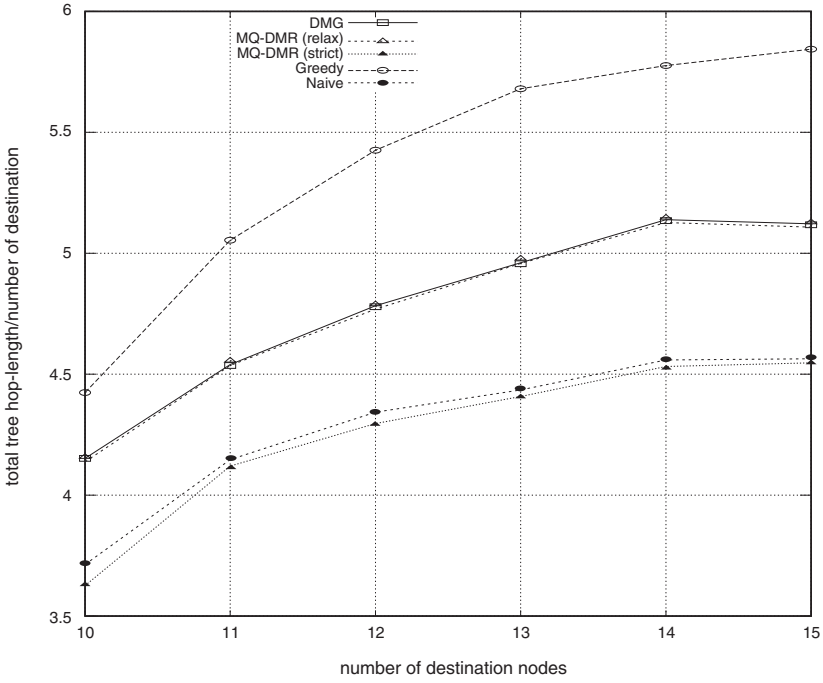

Figure 6. Comparison of mean hoplength/node of the multiast tree for different number of destination nodes

Algorithm name	Average number of hops	Corresponding tree cost
Greedy	5.29	408.12
Naive	3.47	362.18
DMG	4.61	341.67
Least Hop	1.43	445.37
MQ-DMR	**1.04**	**373.96**

Table 2. Average number of hops

Algorithm name	Number of failures	Corresponding tree cost
Greedy	13.82	408.12
Naive	6.52	362.18
DMG	11.10	341.67
Least Hop	0.74	445.37
MQ-DMR	**NONE**	**373.96**

Table 3. Number of failures to find a path

nection for individual nodes for different unconstrained algorithms. Least-Hop always performs way ahead of other unconstrained algorithms in hop-count, though *MQ-DMR* performs better than the Least-Hop algorithm. For the same reason the number of failures for the Least-Hop algorithm is much lower than for others, though it produces at unacceptably high tree cost.

—Number of Failures—

For unconstrained dynamic multicast routing, the route that has been constructed to join a new node may not satisfy the multiple QoS requirements of an application. This we have defined as 'failure'. We observed the average number of such failures for the unconstrained routing algorithms, with a 100-node network and a fixed number of maximum destination nodes (Table 3).

We found that the number of failures for Greedy and DMG is much higher than for Naive and Least-Hop algorithms. It is obvious for the Least-Hop algorithm to have the least number of failures, as hop count is directly related to constraint. Simulation results show (Figures 2 and 3) that our *MQ-DMR* not only satisfies multiple QoS requirements, the resultant tree cost for *MQ-DMR* is much less than that of the Least-Hop algorithm and even less than the Greedy algorithm. In fact further relaxation of QoS requirements can even reduce the tree cost.

Dynamic multicast routing satisfying QoS constraints required for multimedia transmission is an important problem.

Conclusion

Dynamic multicast routing satisfying various QoS constraints required for multimedia (MM) transmission is an important problem. While finding such routes from source to destination is important, it is also important to see that the use of network resources is optimized. We achieved this dual goal through our *MQ-DMR* algorithm. *MQ-DMR* used a unique way of cost calculation for the links constituting the existing MC tree. The links which will remain connected to the MC tree for longer periods will have less cost. Therefore they will be used more to connect newer destinations, to minimize the overall bandwidth use of the network. For MM communication we need to support resource reservation. If we use the WFQ strategy, then we know some upper bound hop

length for delay jitter constraint and loss-free communication. We used those expressions to find the route from source to destination that satisfies all the required QoS constraints. Different simulations show that though *MQ-DMR* searches QoS satisfied routes, it is almost as efficient as other algorithms.

References

1. Ballardie T, Francis P, Crowcroft J. Core based trees (CBT), an architecture for scalable inter-domain multicast routing. *SIGCOMM'93*, Sept. 1993; 85–95.
2. Bharat Kumar K, Jaffe JM. Routing to multiple destinations in computer networks. *IEEE Trans. Communication* 1983; **COMM-31**, No.3, 343–351, March.
3. Braden R, Clark D, Shenker S. Integrated services in the Internet architecture: an overview. *RFC 1633*, June 1994.
4. Clark D, Shenker S, Zhang L. Supporting real-time applications in an integrated service packet network: architecture and mechanism. *Proc. SIGCOMM'92*, 1992; 14–26.
5. Pornavalai C, Chakraborty G, Shiratori N. A New Distributed QoS Routing Algorithm for Supporting Real-Time Communication in High-Speed Networks. *IEICE Transactions on Communications*. 1997; **E80-B**. No.10. October.
6. Chakraborty D, Chakraborty G, Pornavalai C, Shiratori N. Optimal routing for dynamic multipoint connection. *European Transactions on Telecommunication (ETT)*, 1999; **10.** No.2, March–April.
7. Chakraborty D, Chakraborty G, Shiratori N. Multiple QoS constraints satisfying dynamic multicast routing. *ICOIN-14*, 2000; Hsin-Chu, Taiwan, R.O.C.
8. Doar M, Leslie I. How bad is naive multicast routing? *IEEE INFOCOM*, San Francisco, 1993; 1, 82–89.
9. Garey MR, Johnson DS. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
10. Chakraborty G, Pornavalai C, Charkraborty D, Shiratori N. Routing in multimedia communication. *Proceedings of International Conference on Computers and Devices for Communication (CODEC-98)*, Jan. 1998.
11. Guerin R, Kamat S, Orda A, Przygienda T, Williams D. QoS routing mechanisms and OSPF extensions. *Internet Draft, draft-guerin-qos-routing-ospf-03.txt*, March 1998.
12. Guerin R, Orda A. QoS routing in networks with inaccurate information: theory and algorithms. *IEEE/ACM Trans. on Networking*, 1999; **7**, No.3, June.
13. Gupta A. Improved performance for multi-party communication through advance resource reservation. *Proc. of Sixth International Workshop on Network and Operation System Support for Distributed Audio and Video*, April 1996.
14. Huang PC, Tanaka Y. Multicast routing based on predicted traffic statistics, *IEICE Trans. Communication*, 1994; **E77-B**, No.10, Oct.
15. Huitema C. *Routing in the Internet*. Prentice Hall, Englewood Cliffs, NJ, 1995.
16. Hwang FK, Richards DS. Steiner Tree Problems. *Networks*. 1992; **22**, No.1, 55–89. Jan.
17. Karp RM. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, Miller and Thatcher (eds), Plenum Press, New York. 85–103, 1972.
18. Kompella VP, Pasquale JC, Plyzos GC. Multicast routing for multimedia communication. *IEEE/ACM Trans Networking* 1993; **1**, No.3, pp. 286–292, June.
19. Kou L, Markowsky G, Berman L. A fast algorithm for steiner trees. *Acta Informatica*. 1981; **15**. No.11.
20. Ma Q, Steenkiste P. Quality-of-service routing for traffic with performance guarantees. In *IFIP Fifth International Workshop on Quality of Service (IWQOS'97)*, September 1997.
21. Orda A. Routing with end-to-end QoS guarantees in broadband networks. *IEEE/ACM Trans. on Networking*, 1999; **7**, No.3, June.
22. Parekh AK, Gallager RG. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Networking* 1994; **2**(2):137–150, April.
23. Rouskas GN, Baldine I. Multicast routing with end-to-end delay and delay variation constraints. *IEEE JSAC*, 1997; **15**, No.3, April.
24. Salama HF, Reeves DS, Viniotis Y. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *IEEE Journal on Selected Areas In Communications*. 1997; **15**, No.3, 332–345, April.
25. Shenker S, Partridge C, Guerin R. Specification of guaranteed quality of service. *Internet Draft*, August 1996.
26. Turner JS. New directions in communications (or which way to the information age?). *IEEE Communication Magazine*, 1986; **24**, No.10, October.
27. Wang Z, Crowcroft J. Quality of service routing for supporting multimedia applications. *IEEE J. Selected Areas in Communication*, 1996; **14**, No.7. 1228–1234.
28. Waxman BM. Routing of multiple connections. *IEEE Journal on Selected Areas in Communications* 1986; **6**. No.9.
29. Winter P. Steiner problem in networks: a survey. *Networks*, **17**, 129–167.

30. Widyono R. The design and evaluation of routing algorithms for real-time channels. *Technical Report TR-94-024*, University of California at Berkeley, June 1994.
31. Zhu Q, Parsa M, Garcia-Luna-Aceves JJ. A source-based algorithm for delay-constrained minimum-cost multicasting. In *Proc. IEEE INFOCOM, Boston, MA*, 377–385, April. 1995.
32. Zhang L, Deering S, Estrin D, Shenker S, Zappla D. RSVP: A new resource reservation protocol. *IEEE Network* 1993; September.

Appendix: *MQ-DMR* Pseudo-code

Input:
A directed weighted graph $G = (V, E, w)$. $v \in V$ represents a node in the network, and $e(u, v) \in E$ represents the direct link e, from node u to v.
s : source node.
Set of Connection Req $< z_i, T_{ter_i} >$: Request from node z_i for connection until T_{ter_i}.
Disconnection Req $< z_I >$: Disconnection request from node z_i.
$w : E \in R$: Link Cost.
$tag : E \in R$: Link time-tag.
Thus associated with every link there is a pair $< w, tag >$.
$c : E \in R$: Effective link cost $c = w \times$ (*duration of connection*).
$d[h][v]$: The least-cost minimum-hop path estimate from s to v.
$\pi[h][v]$: List of nodes for shortest path from s to v (excluding s and including v) of h-hop length.
$P(v)$: Is the minimum hop shortest path from e to v.

Output:
$P(s, z_i)$: Series of nodes describing least-cost minimum-hop path from s to z_i.

```
INITIALIZE(G, s)
01  for each hop h = 0 to H_max
02    for each predecessor p = 1 to N
03      for each node n = 1 to N
04        do d[h][n] ← ∞
04        do tag[p][n] ← N I L
05        do π[h][v][p] ← NIL
06        if n == s
07          d[h][v][p] ← 0
RELAX(h, v)
08  for v = 1 to N
09    for u = 1 to N
10    if d[h][v] > d[h − 1][u] + w(u, v) × Ψ(T_ter − tag(u, v))
11      then d[h][v] ← d[h − 1][u] + w(u, v) × Ψ(T_ter − tag(u, v))
12        π[h][v] ← π[h − 1][u] ⊕ v
where π[h − 1][u] ⊕ v is the list obtained by adding v to the end of π[h − 1][u] and Ψ(x) = x for x ≥ 0
        = 0 otherwise

EXTRACT_MIN(v)
13  d[0][v] = ∞
14  for h = 1 to H_max
15    if d[h][v] < d[h − 1][v]
16      hop ← h
17      d_min ← d[h][v]
18    h ←(h + 1)
19  P[z] ← π[hop][z]   /* Path P(z) from s to z will be stored at z */
20  return d_min
connection_REQ(z, T_ter)
21  INITIALIZE(G, s)
22  for h = 1 to H_max
23  RELAX(h)
24  EXTRACT_MIN(z)
25  n ← 1
26  u ← s
27  while(P[z][n] ≠ NULL)   /* until destination z is reached */
28    if (tag(u, P[z][n]) < T_ter
29      tag(u, P[z][n]) ← T_ter
30    u ← P[z][n]
31    n ← n + 1
deletion_REQ(z)
/* disconnection initiated by node < z > to source */
32  n ← 1
33  u ← s
34  while (P[z][n] ≠ NULL)   /* until destination z is reached */
35    if (tag(u, P[z][n]) ≤ T_ter)
36      release the link e(u, P[z][n]) from multicast tree
37      tag(u, P[z][n]) ← 0
38    u ← P[z][n]
39    n ← n + 1
```


