MQ: An Integrated Mechanism for Multimedia Multicasting

De-Nian Yang, Wanjiun Liao, *Member, IEEE*, and Yen-Ting Lin

***Abstract*—This paper studies the integration of multimedia multicasting, with the consideration of multicast with end-to-end QoS guarantees by resource reservation, dynamic join and departure of participants, user heterogeneity, scalability, robustness, and loop-free control. A protocol called MQ, Multicast with QoS, is proposed to support multimedia group communications with QoS guarantees for heterogeneous recipients. With MQ, while resource reservation is de-coupled from QoS multicast routing, they are integrated in a way to avoid the problem of "sender-oriented" path determination, a problem that occurs when RSVP is used in conjunction with QoS routing for heterogeneous reservations. Being a truly receiver-oriented and integrated scheme for multimedia multicasting, MQ supports such integration in a robust, scalable and loop-free way. It also accommodates heterogeneous users with varied QoS, dynamically adjusts QoS trees to improve resource utilization, and guarantees end-to-end QoS requirements. We have conducted simulations to evaluate the performance of the proposed mechanism. MQ demonstrates its advantages over the conventional loosely coupled integration of IP multicasting, resource reservation and QoS routing, in terms of better accommodation of heterogeneous users, higher scalability, lower blocking probability for users to join groups with service guarantees, and more efficient resource utilization to enhance system performance.**

***Index Terms*—Multicast with QoS (MQ), multimedia multicasting, quality-of-service (QoS), QoS multicasting.**

I. Introduction

MULTIMEDIA communications over the Internet is the trend, spurred on by the explosive growth of the Internet and the proliferation of the World Wide Web. Compared to traditional applications, multimedia applications pose more stringent demand on the quality of service (QoS) (e.g., a bound on delay or jitter) provided. IP multicasting is a one-to-many or many-to-many communications scenario, achieving resource sharing by avoiding transmitting packets from a sender to each of the receivers separately. A multicast packet contains a class D group address in the destination address field of its IP header, and is delivered to multicast group members with the same "best- effort" delivery as unicast IP data transmission. It is not necessary for a host to be a group member in order to send data to the group. An individual host is free to join or leave a multicast group at any time. No restriction is placed on group members in terms of their physical location and the number of groups they can participate in, and on the group size (the number of participants per group). The success of IP multicasting has been proven by the extensive use of the MBone (Multicast backBone) [1] for multimedia conferences on the Internet.

Resource reservation is an approach that guarantees the quantitatively specified QoS for a particular flow by setting aside certain resources [2]. Resources may be reserved either in a sender-oriented or receiver-oriented way. Sender-oriented (e.g., ST-II [3]) refers to the reservation initiated by a flow sender, and receiver-oriented (e.g., RSVP [4]), by a flow receiver. It has been shown that receiver-oriented reservation accommodates the heterogeneity of group recipients better and demonstrates better scaling property to allow a sizeable group [4]. RSVP is a promising, receiver-oriented QoS signaling protocol that establishes and maintains router states for resource reservation. It relies on *soft state* to cope with the dynamic changes of flow routes and to adapt to network dynamics. A sender initiates a path message (Path) toward all the receivers in a group once a flow transmission starts, and periodically refreshes the existing Path states by successive path messages, lasting for the life of the flow. Reservation works in a similar way. A receiver periodically sends a reservation request (Resv) once it has received the Path message from a sender of interest. The Path message propagates toward all the receivers, in the path determined by the underlying routing mechanism in use. It primes the routers along the path to expect reservations. The Resv message travels in the reverse direction of the same route as in the Path message, ensuring that resources are reserved in the correct path. The paradigm that Path messages start at the flow source and precede Resv messages gives the sender an opportunity to describe the flow specification to its receivers and leaves the responsibility to the receivers to reserve adequate resources based on their individual needs.

QoS routing [5]–[13] determines a transport path (or delivery tree in multicast) with adequate resource to meet the requested QoS level of a data source, with the consideration of the availability of network resources and the QoS constraints of source flows. Here QoS may refer to delay, jitter, loss ratio, or bandwidth. The two objectives of QoS routing are to determine a feasible path which satisfies the QoS constraint of a data flow, and to make efficient use of network resources. It is a challenge to fulfill these two objectives simultaneously, considering the diverse QoS requirements of users and dynamic network state changes. While de-coupled in many existing schemes, resource reservation in conjunction with QoS routing is considered a feasible way to ensure QoS for real-time traffic because resources

Manuscript received June 6, 2000; revised October 31, 2000. This work was supported by the National Science Council, Taiwan, R.O.C., under Grants NSC 89-2219-E-002-004 NSC 89-E-FA06-2. An earlier version of this paper was published in IEEE ICME 2000. The associate editor coordinating the review of this paper and approving it for publication was Dr. K. J. Ray Liu.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. (e-mail: wjliao@cc.ee.ntu.edu.tw).

Publisher Item Identifier S 1520-9210(01)01871-5.

can be reserved successfully only when the underlying routing protocol has found a feasible path that meets the user's needs.

The important issues of multimedia multicasting include QoS, resource reservation, user heterogeneity, scalability, dynamic joining and departure of participants, routing, and open loop control techniques [14]. While considerable research efforts have been made on each individual topic [2]–[26], an integrated framework accounting for all the issues collectively has received little attention. Such integration introduces new problems and challenges. For example, while best-effort IP multicast allows participants to freely join and leave a group of interest, when integrated with QoS routing and resource reservation, a series of dynamic joins and departures of participants, plus dynamic network load fluctuations, may lead to a skewed QoS tree, resulting in inefficient use of network resources, thereby increasing blocking probability for users to join the group with requested QoS. This problem is exacerbated in the presence of user heterogeneity. User heterogeneity is an important and practical consideration, since different users may request different QoS according to their individual needs. Without taking user heterogeneity into account, the utilization of network resources may deteriorate, causing higher blocking probability. Receiver-oriented reservation like RSVP is an accepted approach to accommodate user heterogeneity. However, when RSVP is used with QoS multicast routing protocols, the resultant integration becomes "sender-oriented," (the details of this phenomenon will be explained in the next section). It does not consider the heterogeneity of the user's need for path selection. Finally, the integrated mechanism should be scalable, robust, and loop-free while making efficient use of network resources to minimize the blocking probability for QoS services.

This paper studies the integration problem of multimedia multicasting, with the consideration of user heterogeneity, high scalability, multicast with end-to-end QoS guarantees by resource reservation, and loop-free control. We propose a new protocol called MQ, Multicast with QoS, as an integrated mechanism for multimedia multicasting. MQ targets the potential problems of the integration of IP multicasting, QoS routing, and resource reservations, in the presence of user heterogeneity, while making efficient use of network resources. We will demonstrate how MQ supports such integration in a robust, scalable and loop-free way, accommodates heterogeneous users with varied QoS, dynamically adjusts QoS trees to improve resource utilization, and benefits from being an integrated mechanism to guarantee end-to-end QoS requirements. The performance of MQ will be evaluated by simulation. We will show the advantages of MQ over the conventional, loosely coupled integration of resource reservation with QoS multicast routing in terms of blocking probability, efficiency of resource utilization, and scalability.

The rest of the paper is organized as follows. Section II identifies the potential problems of the integration of IP multicast and QoS protocols in the presence of user heterogeneity. Section III presents the proposed MQ mechanism, including tree branch growing and pruning, tree maintenance, tree reshaping, and loop-free control. Section IV shows some simulation results to compare the performance of MQ with loosely coupled integrated mechanisms for multimedia group communications. Finally, we conclude in Section V.

II. Problems of IP Multicast with QoS

This section investigates the potential problems of integration for multimedia group communications. As is mentioned earlier, there is much effort on each individual topic. However, the problem of integration accounting for all the issues has received little attention. For example, [15]–[19] focused on IP group communications. Reference [4] proposed a reservation protocol which collaborated with the underlying routing mechanism to ensure service quality. References [5]–[13] focused only on the QoS routing algorithms to determine a feasible path/tree which satisfied the QoS constraints of a flow source and optimized the use of network resources. These approaches constructed a QoS tree with given prior knowledge of network topology, the availability of network resources, and group membership, but without the consideration of such practical issues as dynamic join and departure of participants, scalability (in terms of routing table size and the overhead of control message exchanges), and receiver heterogeneity. Reference [21] investigated QoS routing in networks with inaccurate states. Reference [20] discussed the impact of message exchange frequency and link state granularity on computation cost and protocol overhead. Reference [22] proposed a QoS extension to CBT. In [23], the authors proposed QoSMIC, using a local search or a multicast tree search to locate the best path for a new user to join a QoS tree. QoSMIC, however, does not scale well, due to its use of reverse path multicasting for local search and employing a "manager" for multicast tree search. All routers are required to report the link state and group membership periodically to the manager so as to maintain the network and tree topology to assist in optimal path determination.

We have discussed the impact of dynamic join and departure of participants on the efficiency of resource utilization in the previous section. In the following, we will focus on the problem related to multicast with QoS in the presence of user heterogeneity, from the perspectives of the integration of IP multicasting, resource reservation, and QoS routing. To support multicast QoS in the presence of heterogeneous reservations, previous work like RLM [24] proposed to employ layered transmissions and to put different layers on different multicast groups. Such an approach requires as many multicast trees as the number of different layers associated with a single source in a multicast group, and hence have different reservations and distribution trees for the single source. We would like to investigate the same problem from a different angle, limiting ourselves to the layered transmission schemes where all the layers go on one tree.

In the following, we will examine the problem of integrating RSVP in conjunction with two types of multicast routing: shortest path multicast (SPM) and QoS multicast (QoSM). For ease of explanation, we use the network shown in Fig. 1 to illustrate how they work. Fig. 1 shows a network with eight nodes. Assume that S is the flow source, and $R1$, $R2$, $R3$, $R4$ are the flow recipients. The label (a, b) on a link describes the link bandwidth and delay, respectively. The number beneath a

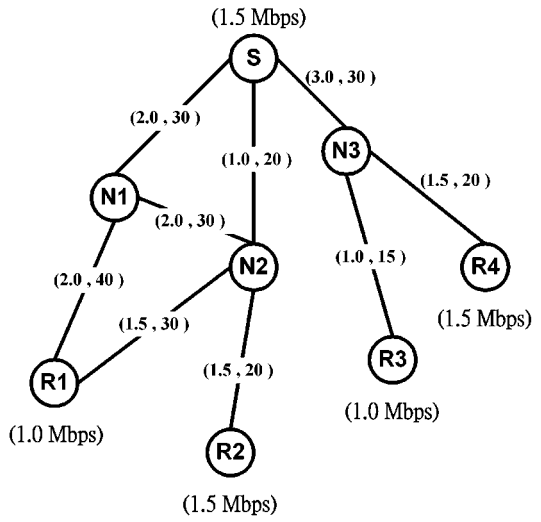

Fig. 1. Example network.

recipient indicates the bandwidth requirement corresponding to the requested QoS of the corresponding receiver. For example, the link between $N3$ and $R3$ has a bandwidth of 1.0 Mbps, a delay of 15 ms, and the bandwidth requirement of $R3$ is 1.0 Mbps. The flow specification (spec) is assumed to be 1.5 Mbps.

A. RSVP with Shortest Path Multicast

Shortest path multicast (SPM) routing protocols such as MOSPF [16] and DVMRP [17] use the Dijkstra and Bellman–Ford algorithms, respectively, to compute the shortest path delivery tree between a sender and each of its receivers. The routing objective focuses on the network topology (or a single metric such as hop-count or delay) only, but not on network resources or conditions such as bandwidth or delay, thereby imposing limitations on the provision of integrated services.

When used with RSVP, resource reservation for a recipient succeeds only when the route the Path message traverses contains sufficient resources to satisfy the requested QoS level. Otherwise, reservation failure for the recipient occurs even if other paths with sufficient resources may exist. Fig. 2 is a simple illustration of RSVP with SPM applied to the network in Fig. 1. Fig. 2(a) shows the RSVP Path messages being forwarded along the shortest path delivery tree determined by the metric of link delay. Fig. 2(b) shows that $R2$ fails in making reservation on the link between S and $N2$, even though there exists a path that meets R2's service requirements (i.e., $R2 \rightarrow N2 \rightarrow N1 \rightarrow S$). A ResvErr message is then sent back to $R2$ to report reservation errors.

B. RSVP with QoS Multicast Routing

Unlike SPM which characterizes trees by a single metric, QoS multicast routing like QOSPF (QoS extensions to OSPF) [25], [26] calculates routes based on multiple constraints and given the full knowledge about network resources, topology, membership information, and QoS requirements of a flow source. Once a feasible QoS path has been determined, the path must contain sufficient resources for the flow transmission, even though the route found may not be the shortest path as in SPM.

When RSVP is used with QoSM, the process of finding a QoS tree is triggered by a flow sender (Path message), using the flow spec as the routing criteria and placing the highest demands on link capability to determine feasible QoS routes. For example, suppose a data source coded with a layered encoding scheme requires at most a rate of 1.5 Mbps for playback. A flow recipient may request to reserve resource at a rate of, say, 64 Kbps, 512 Kbps, etc., according to their individual needs, but not exceeding 1.5 Mbps. The QoS routing determination initiated in this way, however, does not take the heterogeneity of receivers' needs into consideration. Those who have lower QoS requirements may fail to participate in the QoS tree even if there are other paths in the network able to meet their needs with lower QoS demand. Thus, such a path selection is considered "sender-oriented." Fig. 3 depicts such a case. Fig. 3(a) shows the RSVP Path messages being forwarded along the QoS delivery tree determined by the metric of link delay, bandwidth, and flow spec. According to QoS routing, $R3$ cannot participate in the group because no route meets the need of the flow toward $R3$ (failure in the link between $R3$ and $N3$), even though $R3$ demands a lower service level (1.0 Mbps rather than 1.5 Mbps) and the path $R3 \rightarrow N3 \rightarrow S$ does satisfy the request. Fig. 3(b) shows the resultant delivery tree. Note that no recipient on the delivery tree will receive an error message (ResvErr) for a Resv message when using RSVP with QOSPF, because the QoS requirement of the flow has been considered when determining the delivery trees.

To summarize, the fundamental problem of integrating RSVP with QoSM like QOSPF is that the path selection for a flow source to any of the recipients is the reverse of the path from the recipient to the data source. This route determination feature makes RSVP, especially when used in conjunction with QOSPF-like QoS routing protocols, "sender-oriented." The route calculation is initiated by the sender and based on the flow spec, thereby placing the highest QoS level that the flow recipients may request, without considering the heterogeneity of their individual needs. Recall that a receiver-oriented protocol such as RSVP accommodates heterogeneous receivers better than a sender-oriented one such as ST-II. It would be desirable to have a truly "receiver-oriented" integration approach for multicast QoS in the presence of user heterogeneity.

III. MQ: An Integrated Mechanism for Multicast with QoS

Based on the discussion above, we propose an integrated mechanism called MQ, Multicast with QoS, to fulfill the design objective of being a truly "receiver-oriented" mechanism to support heterogeneous users for multimedia group communications in a scalable, robust, efficient, and loop-free manner. MQ sets up a QoS multicast tree with "explicit" join (Join_Request and Join_Ack) from group members, and thus consumes no more network bandwidth than necessary for control message overhead. Besides, MQ adopts the soft state mechanism to maintain constructed QoS multicast trees for the sake of robustness. Finally, MQ dynamically expands, shrinks,

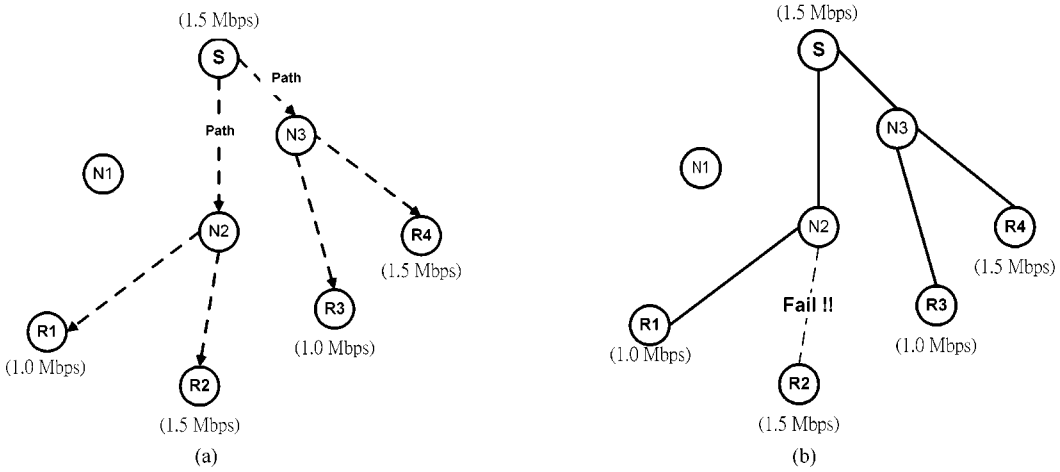

Fig. 2. RSVP with shortest path multicast. (a) Path setup by RSVP with SPM and (b) resultant multicast delivery tree.

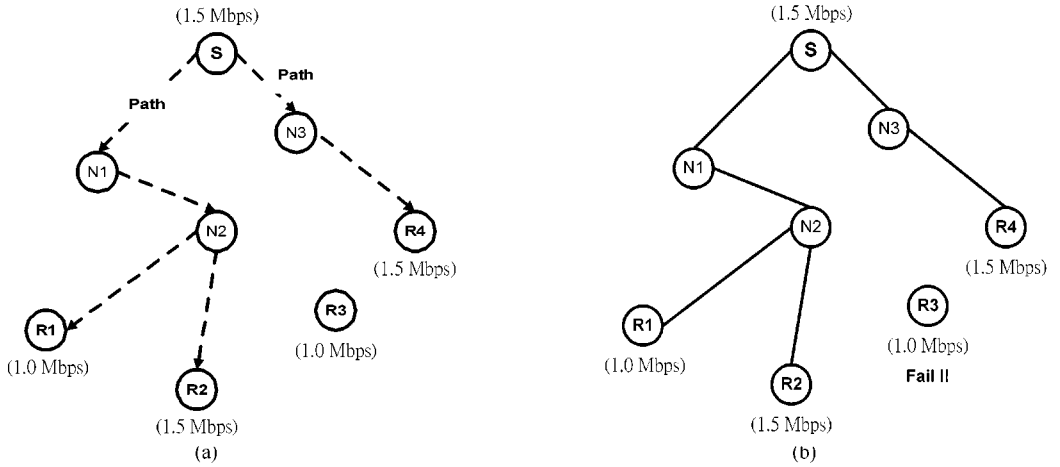

Fig. 3. RSVP with QoS multicast. (a) Path setup by RSVP with QoS multicasting and (b) resultant QoS multicast delivery tree.

and reshapes QoS trees to adjust reservation levels for more efficient resources utilization.

The paradigm of using "Join_Request" and "Join_Ack" makes the proposed approach suitable for both source-based and center-based (shared trees) multicast mechanisms. Two existing, promising shared multicast protocols (i.e., Core-Based Tree (CBT) [18], and Protocol Independent Multicast—Sparse Mode (PIM-SM) [19]) employ "Join_Request" and "Join_Ack" for tree construction as well. In the rest of the paper, we will focus only on source-based multicast protocols. The result of how MQ works in the context of shared trees will be reported in the future. Note that in the rest of the paper, we will explain how MQ works by using bandwidth and hop count as the constraints of QoS routing. The mechanism is applicable to the case of employing multiple QoS metrics such as bandwidth, jitters, delay, etc.

A. Tree Construction

The Flow_Ad message serves to advertise the QoS spec of a data source. When a transmission starts, the sender sends a Flow_Ad message to all flow recipients in the group, multicasting through the shortest path delivery tree to the destinations. Unlike an RSVP Path message, MQ Flow_Ad does not trigger the QoS routing mechanism to determine a feasible path for the data flow, and the "path" states are not saved in the intermediate routers during data forwarding, but merely used for flow advertisement. Upon receiving a Flow_Ad of the source for whose data a receiver wishes to make reservations, the receiver sends a Join_Request message back using the (possibly modified) flow spec from the incoming Flow_Ad as its requested QoS. It is the receiver that initiates the QoS route determination. Since a feasible flow route is determined by the recipient, rather than by the sender, the actual individual needs can be taken into account. A Join_Request message is guided to traverse the feasible QoS route newly determined toward the sender (i.e., upstream). Upon receiving a Join_Request, the intermediate routers in the joining path record the path states and temporary reservation states for the corresponding receiver.

A Join_Request travels upstream only as far as the closest point of the delivery tree where the requested reservation level is met, from where a Join_Ack message is returned

following the same path in the reverse direction traversed by the Join_Request message. The Join_Ack also confirms the reservation in the intermediate routers. If the reservation level already made in a router cannot meet the QoS requirement of an incoming Join_Request, the router first tries to expand the reservation level to satisfy the new request, and then forward the request upstream. The reservation expansion continues as far as the breakout point of the tree, the resource of which is insufficient to meet the requested QoS. The breakout router uses QoS routing to determine a new feasible path with sufficient resources on behalf of the Join_Request for the source. If such a path is found, the router forwards the request toward the new path and waits for an acknowledgment; otherwise, the router returns a Join_Fail message to the immediate downstream router from where the Join_Request comes. Upon receiving a Join_Fail message, the router acts as a breakout router, using QoS routing to determine a new feasible path with sufficient resources. This operation repeats until either condition holds: 1) an on-tree router located at the joining path has found a feasible path based on QoS routing and has received a Join_Ack from the new path, or 2) all the on-tree routers located at the joining path have failed to find a feasible path, causing the recipient to receive a Join_Fail message. If 2) is the case, the recipient may consider resubmitting a new request with a lower QoS. Otherwise, upon receiving a Join_Ack, the breakout router forwards the Join_Ack downstream to confirm the reservation, and sends a ResvRev (reservation remove) message upstream in the old path to relinquish resources reserved by the previous requests. This makes the resources reserved sufficient to meet the new request with a higher QoS level. Since we refrain from tearing down the reservation on the existing path to the source while attempting to find a higher-capacity path over another path, the existing flow recipients not be disrupted by the new request which discovers that an on-tree link currently has insufficient capacity to serve. As a result, the resources reserved on the current path are not affected by admission failures on the alternative path, therefore preventing a new receiver from accidentally denying service to the existing receivers simply by requesting lots of bandwidth.

Fig. 4 illustrates how MQ works on the network in Fig. 1. Fig. 4(a) shows that source S multicasts a Flow_Ad message to recipients $R1$, $R2$, $R3$, and $R4$ through the shortest path tree. Assume that 1) $R1$ receives the Flow_Ad message earlier than $R2$, and 2) $R4$ receives the Flow_Ad message earlier than $R3$. Fig. 4(b) shows $R1$ $(R4)$ issuing the Join_Request message that triggers the QoS routing mechanism finding $R1 \rightarrow N2 \rightarrow S$ $(R4 \rightarrow N3 \rightarrow S)$ to be the optimal path to source S. The Join_Request message is forwarded; the path states and the temporary reservation states in the intermediate routers are set up until reaching source S, from where the Join_Ack is returned and the resource reservation is confirmed accordingly. Later $R3$ receives the Flow_Ad and sends its Join_Request message toward S, finding $R3 \rightarrow N3 \rightarrow S$ as the optimal path. $R3$'s Join_Request message propagates only as far as router $N3$, the closest point of the delivery tree where the requested QoS level is met, as in Fig. 4(c). Then, $R2$ receives the Flow_Ad and sends its Join_Request message that finds $R2 \rightarrow N2 \rightarrow N1 \rightarrow S$ as the optimal path. $R2's$ Join_Request message travels to router S in which the reservation level already made cannot meet the request. A Join_Fail message is then returned to $N2$, as in Fig. 4(d). Since the upstream link does not have enough resources for reservation expansion, $N2$ (i.e., breakout router) uses QoS routing to locate a new path, and forwards $R2$'s Join_Request to $N1$, and to S. Upon receipt of the Join_Ack from $N1$, $N2$ then issues a ResvRev message to the source $(N2 \rightarrow S)$ to relinquish the resources reserved previously, as in Fig. 4(e). Fig. 4(f) shows the resultant MQ delivery tree.

Note that for those group members without specific demands on QoS, they can just join the MQ tree with a Join_Request indicating "best-effort" service in the field of QoS requirement. The Join_Request message is routed toward the MQ tree without making reservations. The Join_Request again travels as far as the closest point of the MQ tree, from where the Join_Ack is sent back in the reverse path of the Join_Request.

B. Tree Maintenance

Tree maintenance refers to the operations, once an MQ tree has been built, to 1) maintain tree robustness and loop-freedom in the face of network dynamics (e.g., route changes or link failures) and source changes, and 2) enable the existing users to change the requested QoS and the newly arrived users to obtain the flow spec to make requests for QoS services. With MQ, a Join_Request message triggers the calculation of a feasible QoS path for a flow and makes the reservation at the requested service level, awaiting confirmation by the receipt of a Join_Ack message. Using a Join_Request (with Join_Ack, Join_Fail, and ResvRev) message to serve both path finding and resource reservation purposes allows "one" message to be sent periodically to refresh the existing unified states of both the path and reservation for an MQ tree, rather than sending Path and Resv messages separately as in RSVP. Thus the refreshing overhead can be greatly reduced. The two main messages used to maintain a constructed MQ tree are described in the following.

1) Flow_Ad: It is sent by a source on three occasions: periodic distribution, whenever there is a change in the source, and per request. A source periodically multicasts a Flow_Ad message to the group, transmitting through the shortest path delivery tree to all the recipients. This ensures all the group members are informed of the flow condition, and enables the newly arrived receivers to obtain the flow spec to make proper reservations. A change in the source (e.g., the source goes away) triggers a Flow_Ad, allowing the MQ tree to learn promptly and to adapt gracefully to source changes. Upon receiving a Flow_Ad message that indicates a source change, receivers issue Join_Request messages for path finding and for reservations. A newly arrived receiver is allowed to solicit for the flow spec to the source using a Flow_Solicit to further reduce join latency. Here, join latency refers to the time elapsed between when a user joins a group and when it starts receiving the data with the requested QoS. Since Flow_Ad is used for flow advertisement, and can be sent per request as well, the multicast frequency of the Flow_Ad can be even lower than that of the Refresh messages for both path and reservation.

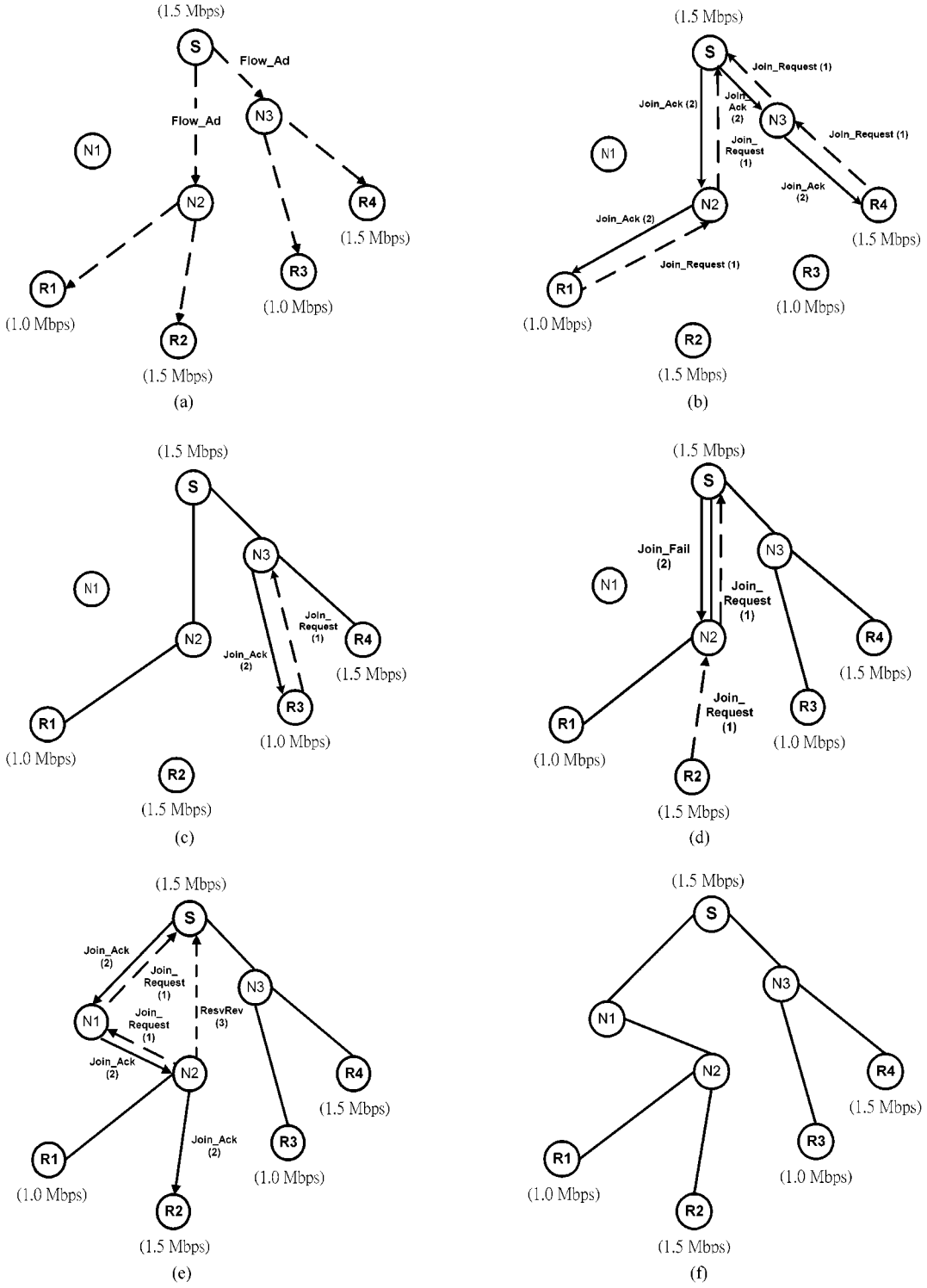

Fig. 4. MQ tree construction. (a) Flow_Ad message, (b) Join_Request and Join_Ack messages, (c) Merge operation, (d) Join_Fail message, (e) ResvRev message, and (f) resultant QoS multicast deliver tree.

2) Refresh: It is sent by a receiver to periodically refresh the existing unified states of both "path" and "reservation." A Refresh message serves two purposes: keeping existing reservation alive and requesting a change in QoS. Once having successfully joined a group, the receiver periodically sends a Refresh message upstream, following the constructed MQ tree. Each node in the multicast tree is associated with a timer in each downstream link (interface). A timer is reset upon receipt of a Refresh message from the corresponding interface. On expiry of the timer, the node assumes the downstream receivers no longer exist and issues a TearDown message to the interface to release all the previously reserved resources. On receiving a TearDown message from the upstream interface, the node

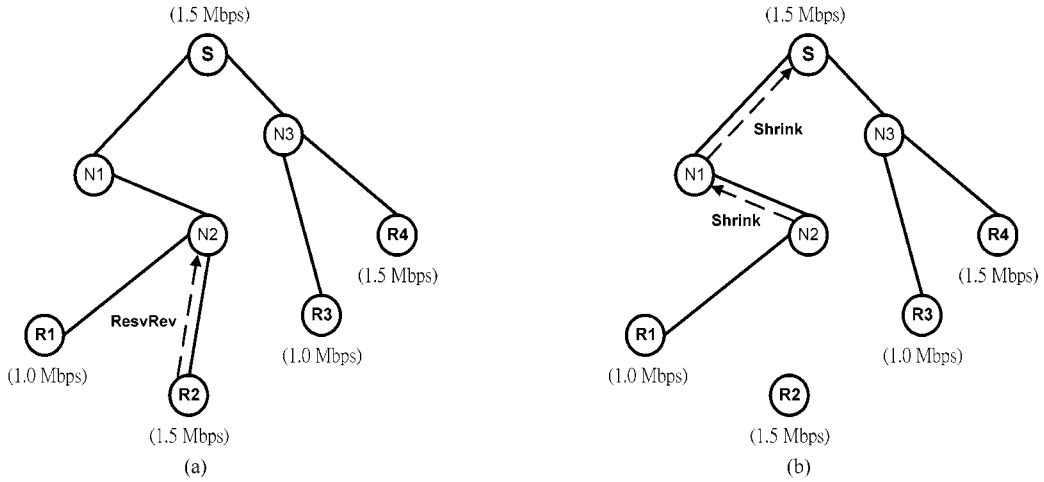

Fig. 5. MQ tree pruning.

issues a TearDown message to each of its downstream interfaces. When a receiver would like to change the requested QoS, a Refresh message is issued and the QoS requirement field of the Refresh message is modified accordingly. The Refresh message is then forwarded to its directly attached node upstream, from where a new Refresh message is issued (to change the QoS). Note that the change QoS request shrinks the tree as in the departing process.

C. Tree Pruning

To leave an MQ tree, a receiver sends a ResvRev message toward the root to clear the states and release the resources reserved in the intermediate routers. If a router detects that the departing interface (i.e., the interface where the ResvRev message comes from) has the highest QoS, the router sends a Shrink message upstream to reduce the reservation in the upstream link to the maximum reserved bandwidth among all the other downstream interfaces, avoiding over-reserving resources. This shrinking process continues as far as the closest point of the tree where the departing interface is no longer the one with the highest QoS among all the interfaces in the on-tree router.

Fig. 5 demonstrates how MQ shrinks the tree constructed in Fig. 4. Originally $R2$ has reserved a bandwidth of 1.5 Mbps on the path $S \rightarrow N1 \rightarrow N2 \rightarrow R2$. To leave the group, $R2$ sends a ResvRev to its upstream router $N2$ [Fig. 5(a)]. $N2$ finds that the reserved bandwidth in the departing interface (i.e., 1.5 Mbps) is higher than the maximum reserved bandwidth among all the other downstream interfaces (i.e., 1.0 Mbps), and thus sends a Shrink message to the upstream router $N1$ to shrink the reserved resource. Having reserved 1.5 Mbps, $N1$ reduces the reservation to 1.0 Mbps and then forwards the Shrink message upstream to router S, where the resource is shrunk accordingly.

D. Tree Reshaping

The shape of QoS trees (i.e., the number of links with reserved network resources, and how routers are connected) determines how efficient network resources are used. In real networks, available link bandwidth may be changed dynamically, due to transient network load fluctuations, connections and disconnections, and links going up and down. Besides, a group member may dynamically join or leave the participating multicast group. A new branch grows in a tree at the time a new recipient joins the group, and is trimmed upon the departure of a recipient. Without taking this into consideration, the constructed QoS tree may become skewed after a series of joining and departure of participants. In the following, we will describe how MQ reshapes a QoS tree so as to make efficient use of network resources.

To reduce the routing table size for the sake of scalability, an MQ router keeps the information about network topology and the residual bandwidth per link, rather than storing tree topology and the reserved bandwidth per (source, group) pair in each link, i.e., information is stored on a per network basis, not per group.[1] Each router alone cannot determine if a tree should be reshaped. Therefore, information is exchanged with other routers. Tree reshaping is performed only when the reshaped tree consumes less network resources than the original one. Thus, if an MQ capable router 1) finds a new feasible path, from the source to itself, which satisfies the QoS requirements of all the downstream interfaces, and 2) finds that the resource reserved for the new path is less than the old path, the multicast tree is reshaped by adding the new path and then removing the old path from the tree. The operation of tree reshaping is described as follows.

1) A router employs QoS routing with the maximum reserved bandwidth among all the downstream interfaces as the QoS metric to determine if a feasible path exists. When the upstream router in the newly determined path is different from that in the current tree, the router will consider reshaping a tree, and the router is called a reshaping router.
2) The reshaping router sends an Off_Tree_Query to the new path, with information on the maximum bandwidth reserved among all the downstream interfaces, hop

[1]While maintaining states on a per group basis may lead to a lower blocking probability to join a QoS tree and to better resource utilization, at the expense of low scalability, MQ opts to employ the approach based on per-network states so as to render better scalability.

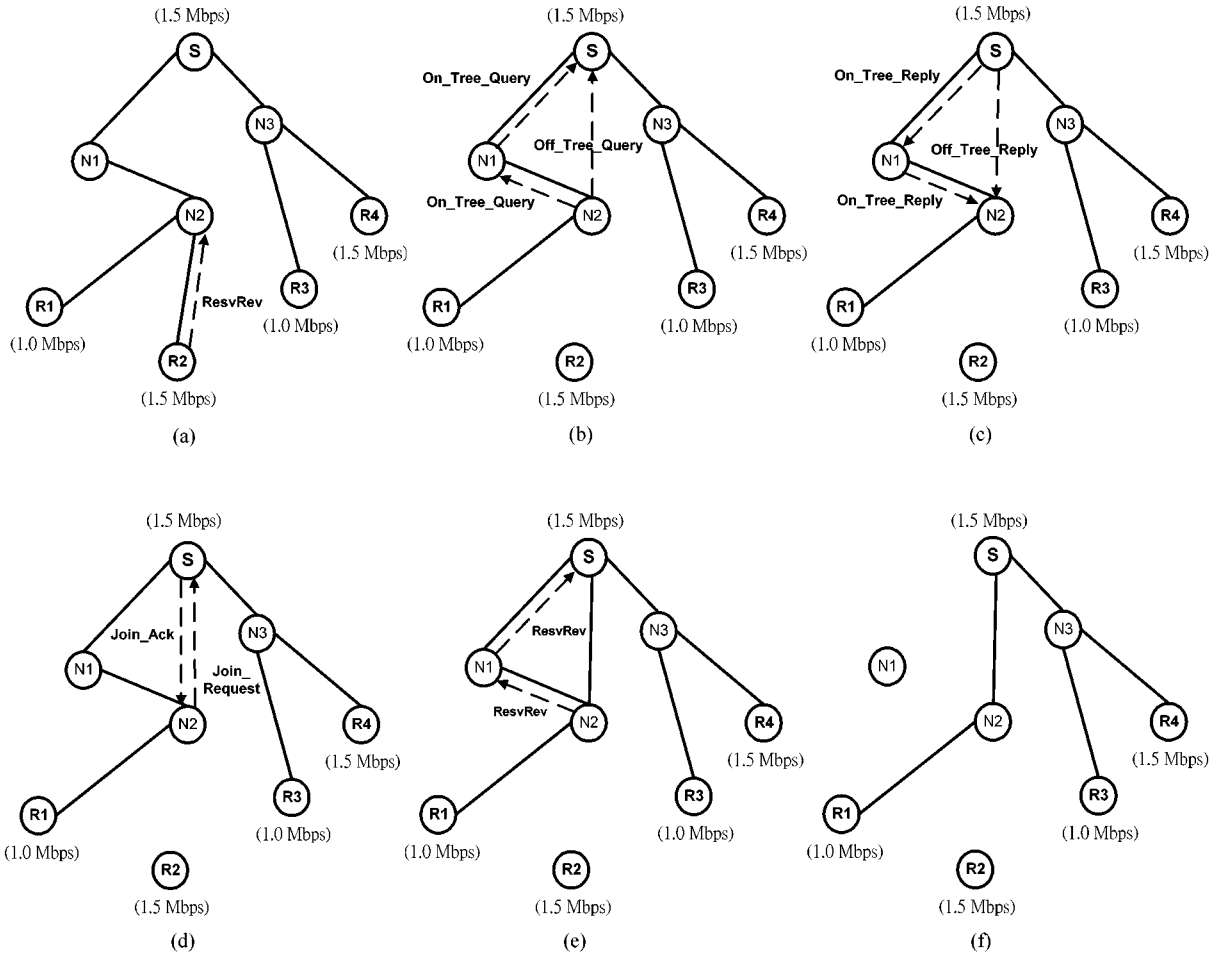

Fig. 6. Tree reshaping.

count, and the address of the reshaping router. The hop count field of the Off_Tree_Query is incremented by one whenever a router is traversed. Upon receipt of an Off_Tree_Query, an on-tree router compares the bandwidth reserved in its upstream link with the bandwidth requested by the Off_Tree_Query, and responds with an Off_Tree_Reply. If the bandwidth reserved in the upstream link is larger than that of the Off_Tree_Query, the hop-count value in the Off_Tree_Query is copied to the Off_Tree_Reply; otherwise, the hop count in the returned Off_Tree_Reply is set to infinity.

3) The reshaping router also sends an On_Tree_Query toward the upstream router in the original tree, with the On_Tree_Query's hop-count. The hop-count in the On_Tree_Query is incremented by one whenever a router is traversed. The On_Tree_Query travels upstream along the multicast tree until reaching an on-tree router with more than one downstream interface, from where an On_Tree_Reply with a copy of the hop count of the On_Tree_Query is returned.
4) Upon receipt of both the Off_Tree_Reply and the On_Tree_Reply, the reshaping router compares the hop count values of both messages. A smaller hop count in the Off_Tree_Reply implies less bandwidth is required for the reshaped tree. Thus, only when the Off_Tree_Reply is smaller will the tree be reshaped. To start reshaping the tree, the reshaping router issues a Join_Request message along the new path, waiting until a Join_Ack message is returned, at which time a ResvRev message is sent toward the upstream router in the original tree to complete tree reshaping.

Fig. 6 demonstrates how MQ reshapes a tree. When recipient $R2$ leaves the multicast group, a ResvRev message is sent toward its upstream router $N2$. Upon receipt of the message, $N2$ uses the bandwidth of 1.0 Mbps as the metric of QoS routing, finding that there is a path (i.e., $N2 \rightarrow S$) having an upstream router different from that in the original multicast tree. $N2$ then sends an Off_Tree_Query and an On_Tree_Query to the upstream router in the new path and in the original multicast tree, respectively. Since the Off_Tree_Query reaches the on-tree router S in one hop only, while the On_Tree_Query reaches an on-tree router with more than one downstream interface in two hops, the router $N2$ starts a reshaping process with a Join_Request to the new path. Once the Join_Ack has received, $N2$ sends a ResvRev message to the previous upstream router $N1$ to complete the reshaping process.

While the reshaping process may make efficient use of network resources, a closely related issue to consider is when a router should start a reshaping procedure. One possibility is to start when triggered by changes in network load. Once the residual bandwidth in a link exceeds a threshold, the detecting router initiates a tree reshaping. Another possibility is to start upon the departure of a downstream interface. Upon receiving a ResvRev message from a downstream router, if the router detects the maximum bandwidth to be reserved among all the downstream interfaces other than the departing interface is less than the bandwidth reserved in the upstream link, a reshaping process starts. Both approaches, however, suffer from large overhead in computing and communications, and low successful rate in reshaping. In the following, we suggest two schemes to determine when a reshaping process should be initiated.

1) Each router periodically checks the residual bandwidth in each link within a subgraph centered at the router and a finite number of hops from it. Tree reshaping is performed if either condition holds: 1) if the increase of the total residual bandwidth of all links within the checked subgraph reaches a predefined threshold, or 2) if the increase in the number of links with sufficient residual bandwidth within the checked subgraph reaches a predefined threshold.
2) Upon receiving a ResvRev message, a router compares the reserved bandwidth in the upstream link with the maximum reserved bandwidth among all the downstream interfaces other than the departing interface. If the difference exceeds a given threshold, a tree reshaping is performed; otherwise, the router just sends a Shrink message to the upstream router to reduce the reserved bandwidth.

E. Loop-Free Control

A loop in packet delivery may be caused due to transient routing table inconsistencies. To reduce the negative impact of loops on networks, IP time-to-live (TTL) is often employed to limit the maximum number of hops each IP packet is allowed to travel. In MQ, to avoid loops, loop detection is performed whenever there is a change in tree topology, which may happen in the joining process (to grow a new branch) or in the reshaping process (to reshape[2] a tree). During the joining process, when an on-tree router (i.e., breakout router) finds the bandwidth reserved in its upstream link cannot satisfy a received Join_Request, the router will employ QoS routing to locate a new feasible path with sufficient resource for the source. Once such a new path has been determined, the router sends the Join_Request upstream toward the new path. If there is a loop formed by the new path to the original multicast tree, the Join_Request will eventually come back to the breakout router and thus the loop can be detected accordingly. Upon receipt of the Join_Request, the breakout router returns a Join_Fail message to the downstream router in the new path from where the Join_Request comes. Those routers that receive a Join_Fail message then in turn clear the temporary states and re-initiate QoS routing to determine if another new feasible path exists.

Fig. 7 demonstrates how loop-free control is ensured in the joining process. Fig. 7(a) shows the network topology with 11 nodes in which node S is the flow source, Rx is a flow recipient, and Ny is a router, $x = 1 \cdots 4$, $y = 1 \cdots 6$. The label (a, b) on a link denotes the total link bandwidth and the bandwidth reserved for source S, respectively. The number beneath a recipient indicates the bandwidth requirement corresponding to the requested QoS of the recipient. Fig. 7(b) shows an MQ tree, with three recipients $R2$, $R3$, and $R4$. Later, $R1$ would like to join the multicast group by issuing a Join_Request with a bandwidth of 1.5 Mbps to $N2$. In Fig. 7(c), since the reserved bandwidth in $N2$ can be expanded to satisfy the Join_Request, $N2$ forward the Join_Request upstream to $N1$. Since the reserved bandwidth of $N1$ still can be expanded to satisfy the Join_Request, the Join_Request is forwarded to S. On detecting that the link $(S, N1)$ cannot meet the Join_Request, S returns a Join_Fail to $N1$. Based on QoS routing, $N1$ finds no other feasible path to reach S and returns a Join_Fail to $N2$. In Fig. 7(d), using QoS routing, $N2$ finds there is a feasible path, i.e., $N2 \rightarrow N4 \rightarrow N5 \rightarrow N6$, to reach S, and thus sends a Join_Request to $N4$. Since $N4$ is on-tree and the bandwidth reserved can be expanded to meet the Join_Request, $N4$ then sends the Join_Request to the upstream router $N3$ which in turn sends the Join_Request to its upstream router $N2$, because the reserved bandwidth in the upstream link $(N3, N4)$ can be expanded to meet the request, too. Fig. 7(e) shows that upon receipt of the Join_Request, $N2$ detects the loop and sends a Join_Fail to $N3$. On receiving the Join_Fail, $N3$ finds that no other feasible path to the source exists and sends a Join_Fail to $N4$. $N4$ uses QoS routing to locate a feasible path (i.e., $N4 \rightarrow N5 \rightarrow N6$) to reach S, and thus sends a Join_Request to $N5$. Fig. 7(f) shows that since the reserved bandwidth in the upstream link $(N6, N5)$ can satisfy the Join_Request, $N5$ then returns a Join_Ack to $N4$. Upon receiving the Join_Ack, $N4$ sends a Join_Ack to $N2$ and sends a ResvRev to $N3$. Fig. 7(g) shows that upon receipt of the Join_Ack, $N2$ sends a Join_Ack to $R1$, and sends a ResvRev to the previous upstream router $N1$ and to S. Fig. 7(h) shows the resultant tree.

During the reshaping process, loop detection is performed explicitly. The bandwidth request of an Off_Tree_Query issued by an on-tree router (i.e., the query sender) is the maximum bandwidth reserved among all the downstream interfaces of the query sender. On receiving an Off_Tree_Query, if an on-tree router can meet the request of the Off_Tree_Query, a loop may occur. Thus, to avoid loops, even if an on-tree router can meet the request of a received Off_Tree_Query, the router (loop-detection sender) first sends upstream in the tree a Loop_Detection message with the address of the query sender. If there is a loop, the Loop_Detection will eventually reach the query sender; otherwise, the Loop_Detection will eventually go to the source, from where a Loop_Reply message is returned, informing the loop-detection sender of loop freedom. Upon receiving a Loop_Detection, the query sender returns a Loop_Reply message to warn the loop-detection sender the existence of a loop. Once the Loop_Reply has been

[2]Note that in tree reshaping, MQ does not reconstruct the whole QoS tree, but replaces an old branch in a multicast tree with another new feasible branch consuming less network resources.

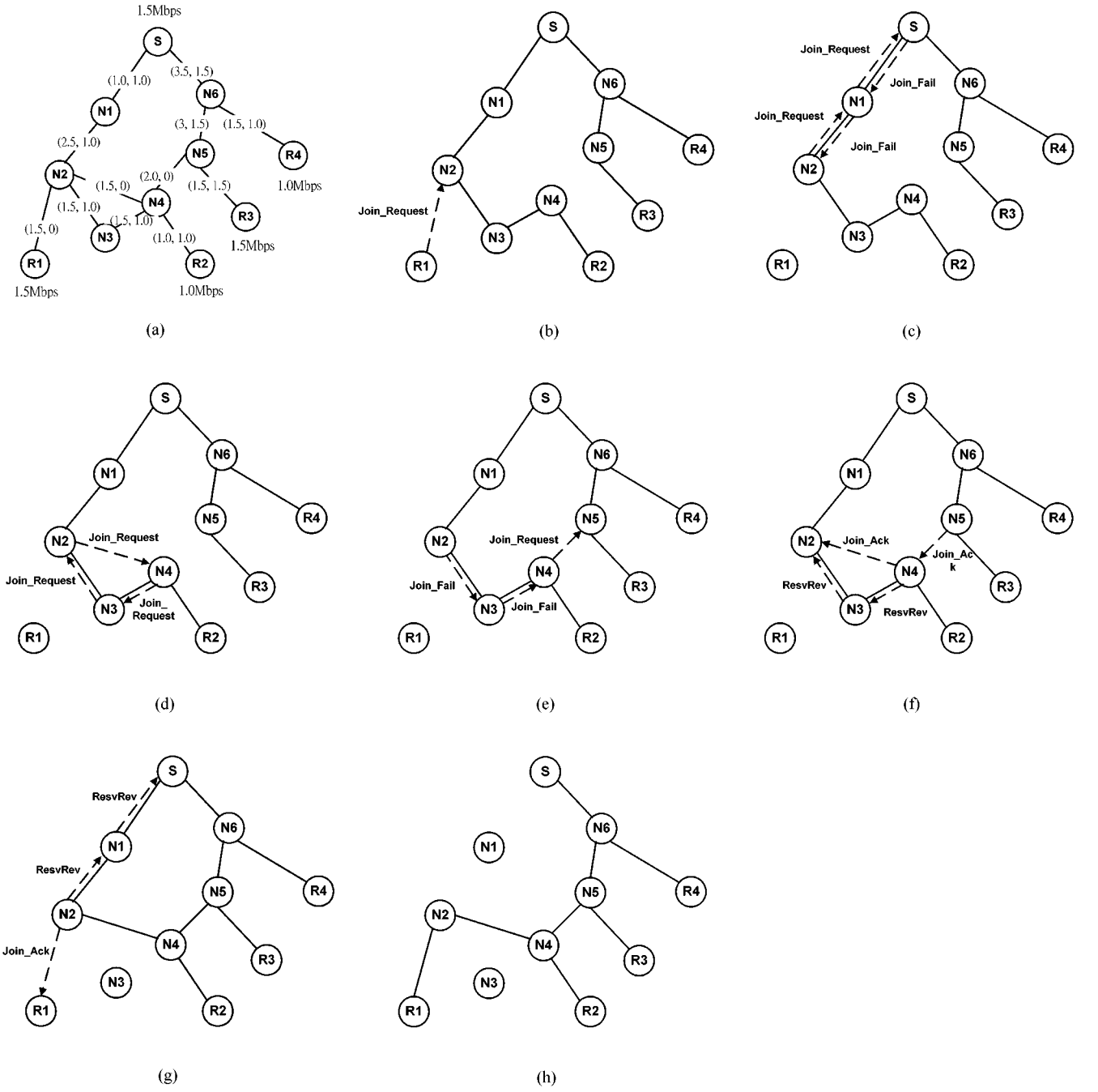

Fig. 7. Loop-free control in the joining process.

received, the loop-detection sender sets the hop count in the Off_Tree_Reply to *infinity* to avoid a loop.

F. Other QoS Metrics

This section describes how to provide guarantees on QoS metrics other than bandwidth, such as end-to-end delay, end-to-end delay jitter, and packet loss rate, in MQ. Since all three metrics are additive constraints in QoS routing, we will focus only on the end-to-end delay constraint in the following. The same approach is applicable to end-to-end delay jitter and loss rate.

To describe how the protocol works, some notations are defined first.

N	router in the network;
N_u	upstream router of router N;
N_d	downstream router of router N;
R_k	recipient k in an MQ tree;
AD_{R_k}	accumulated delay carried by Join_Request in the joining path from recipient R_k;
AD_S	accumulated delay carried by Join_Ack in the returning path from sender S;
D_{R_k}	end-to-end delay constraint requested by recipient R_k;

$D_{S,N}$ total delay in the path between sender S and router N in an MQ tree;

$d_{x,y}$ delay in the link between router x and router y in an MQ tree;

I_N^j set of all recipients in the subtree rooted at router N from interface j in an MQ tree;

D_{N,R_k} total delay in the path between router N and recipient R_k, $R_k \in I_N^j$;

D_N^j minimum partial delay constraint among all the recipients downstream from interface j of router N (i.e., I_N^j), where

$$D_N^j = \min\{D_{R_i} - D_{N,R_i} | \forall R_i \in I_N^j\}.$$

To guarantee end-to-end delay, MQ should be extended to maintain the following states in each on-tree router, say router N: $D_{S,N}$ and D_N^j for each downstream interface j of router N, $j = 1, 2, \cdots, n$, as shown in Fig. 8. With $D_{S,N}$, an on-tree router N can determine locally if the delay constraint requested by a newly joining recipient can be satisfied, without forwarding the Join_Request all the way to source S. With D_N^j, the delay constraints imposed by all the recipients in I_N^j can be summarized in a single state, rather than maintaining individual states for all the downstream recipients.

Upon joining a multicast group, a new recipient R_k includes D_{R_k} and AD_{R_k} in the Join_Request message. AD_{R_k} is set to zero initially. On receiving a Join_Request from interface j, a router N which is not on-tree first adds the link delay d_{N,N_d} to AD_{R_k}, and then creates a temporary state with D_N^j set to the value of the updated AD_{R_k}. Note that here we assume the upstream router (router N in this example) instead of the downstream one (router N_d in this example) knows the link delay. To proceed, N forwards the Join_Request with R_k and the updated AD_{R_k} in the next best hop upstream toward the source. Such a process is repeated until the message meets the first on-tree router. When a Join_Request reaches the first on-tree router from interface j, the router, say router N, will perform the following two operations if $D_{R_k} - D_{N,R_k} \geq D_{S,N}$.

1) Return a Join_Ack message which contains AD_S to acknowledge success in joining the tree. AD_S is set as $AD_S = D_{S,N} + d_{N,N_d}$, the sum of $D_{S,N}$ and the link delay between router N and its downstream router N_d. Upon receipt of a Join_Ack, a router N becomes on-tree and confirms the state established by the corresponding Join_Request message. Router N then sets its $D_{S,N}$ as the value of the AD_S carried in the received Join_Ack, and updates AD_S accordingly before passing the message downstream to N_d. The Join_Ack message follows the same path but in the reverse direction as the corresponding Join_Request. The procedure continues until R_k receives the Join_Ack message.
2) Check if D_N^j is minimum among all the downstream interfaces, namely $D_N^j = \min_{\forall i}\{D_N^i\}$. If this is the case, router N sends a Refresh message with D_N^j toward its upstream router, say router N_u, to reflect the change; otherwise, the change is suppressed by other interfaces (i.e., no Refresh message is issued). Upon receipt of a Refresh message from interface j, the upstream router N_u updates the corresponding partial delay constraint $D_{N_u}^j$ with $D_{N_u}^j = D_N^j + d_{N_u,N}$, the sum of D_N^j and the link delay between routerN_u and router N, and again determines if another Refresh message should be forwarded to its upstream router. This operation is repeated until the change is suppressed by other interfaces.

When a router receives a ResvRev message, it checks if $D_{N_u}^j$ should be modified. If this is the case, the router will forward a Refresh message to its parent router as described in 2) above. To find a new upstream path either in the joining or reshaping process, an on-tree router N uses $\min_{\forall i}\{D_N^i\}$ as the delay constraint for QoS routing to forward the Join_Request. If router N finds a new feasible upstream path that leads to a change in $D_{S,N}$, it will send a Refresh message with the updated $D_{S,N}$ to each of its downstream routers to reflect the new delay information.

Use Fig. 1 as an example. Assume that the end-to-end delay constraints of $R1$, $R2$, $R3$, and $R4$ are 100 ms, 120 ms, 80 ms, and 40 ms, respectively, and the joining sequence is $R3$, $R4$, $R1$, and $R2$. First, $R3$ finds a feasible path to the source and joins successfully. $R4$ fails to join because the minimum delay from the sender to $R4$ is 50 ms while the demand of $R4$ is 40 ms. Then $R1$ joins the group successfully and creates a state in $N2$. Later $R2$ would like to join the group by sending a Join_Request with delay constraint 120 ms. Upon receiving the Join_Request from $R2$, $N2$ finds that the delay between the sender and $N2$ is 20 ms, smaller than the partial delay constraint of the outgoing interface to $R2$, 100 ms. However, since the reserved bandwidth of its upstream link cannot meet $R2$'s requirement, $R2$ tries to use the following two metrics for QoS routing to check if a feasible path exists: 1) the minimum partial delay bounds among all the downstream interfaces (70 ms in this example), and 2) the maximum bandwidth requirements among all the downstream recipients (1.5 Mbps in this example). It then finds that the path $S \rightarrow N1 \rightarrow N2$ can satisfy both requirements, and thus join successfully. Later $R2$ leaves the group. After $R2$ has left this group, $N2$ initiates a reshaping process to reshape the tree (as described in Section III-D). It uses 70 ms and 1.0 Mbps as delay and bandwidth constraints to locate a new path in the reshaping procedure.

IV. Performance Evaluation

In this section, we present simulation results to verify the arguments we have made in the previous sections. The simulation setup is described first, followed by the simulation results.

A. Simulation Setup

Fig. 9 shows the network topologies used in the simulation. Fig. 9(a) and (b) were generated by network topology generator GT-ITM developed by Georgia Tech., adopting the random graph model designed by Waxman [28] to reflect the structure of real internetworks. This model distributes the nodes randomly in the plane, and for an edge between pairs of nodes (u, v), the edge probability is given by:

$$P(u, v) = \beta \exp \frac{-d(u, v)}{\alpha L}$$

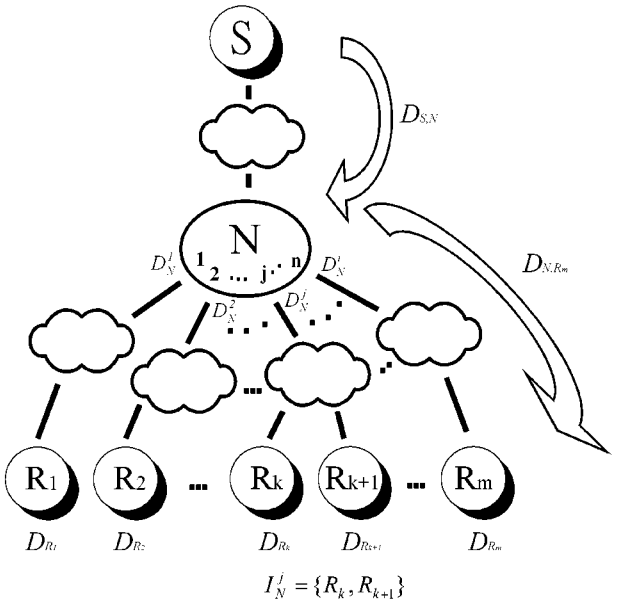

Fig. 8. Notations in MQ.

where $0 < \alpha, \beta \leq 1$ and $d(u, v)$ is the Euclidean distance from node u to v. An increase in α increases the number of connections of distant nodes, while an increase in β increases the edge densities. The network shown in Fig. 9 has 100 nodes with α equal to 0.2 and β equal to 0.2. Note that we have simulated different pairs of values of α and β, and the results were consistent with those shown in this paper. To conserve space, only the results using $(\alpha, \beta) = (0.2, 0.2)$ were included. The QoS metrics were based on bandwidth and hop count. The bandwidth was uniformly distributed between 128 Kbps and 3.088 Mbps. The bandwidth assigned to each link was chosen randomly in the range of ($B1 = 128$ kbps, $B2 = 3.088$ Mbps), so that the aspect ratio (i.e., the ratio of the downstream bandwidth to the upstream bandwidth) φ between a pair of complementary links was no more than three using the formula in [29]:

$$b(u, v) = \text{rand}\,(B1, B2),$$
$$b(v, u) = \text{rand}\,\left(\max(B1, b(u, v)/\varphi), \min(B2, \varphi b(u, v))\right)$$

where rand(X, Y) denotes a random number uniformly distributed between X and Y.

Each node may have one of the following requested bandwidth requirements from the users: 256 kbps, 128 kbps, 64 kbps, and 32 kbps. The inter-arrival time was exponentially distributed with mean 6 min, and the time a receiver spends in a group was exponentially distributed with mean 60 min. The measurement lasts over a 3-h time span.

B. Performance Metrics

The simulations were conducted to compare MQ with the loosely-coupled integration of IP multicasting, QoS routing, and resource reservation. We evaluated three approaches, namely RSVP with SPM, RSVP with QoSM, and our proposed MQ. In the simulation, we used MOSPF for SPM and QOSPF for QoS multicasting. MOSPF employed single metric of the hop count to calculate the shortest paths between a sender and any of the receivers. For QOSPF and MQ, the QoS routes awere determined using the metrics of hop count, bandwidth on a link, and the QoS levels of the (modified) flows. For a fair comparison, MQ used the same QoS routing mechanism as in QOSPF shown in the Appendix.

The following performance metrics were measured.

1) Blocking probability: The probability that a receiver is blocked from joining the QoS tree with resources reserved at the requested QoS level.
2) Protocol overhead: The total number of control messages generated during tree construction, tree pruning, tree reshaping, and tree maintenance, with the consideration of loop-free control. RSVP with MOSPF may have Path, Resv, ResvErr messages; RSVP with QOSPF may have Path and Resv; while MQ may have Flow_Ad, Join_Request, Join_Ack, ResvRev, Join_Fail, Off_Tree_Query, On_Tree_Query, Off_Tree_Reply, On_Tree_Reply, Loop_Detection, Loop_Reply, and Refresh. Routing overhead was not considered in the simulation. For a fair comparison, we normalized the collective protocol overhead by the total number of admitted recipients per group (the users successfully joining a group) because the overhead increased as the total number of admitted receivers per group increased.
3) Resource utilization: Resource utilization is defined as the reserved bandwidth over the total link bandwidth. Again, since the resource consumption increases as the number of admitted receivers increased, for a fair comparison, we normalized the resource utilization by the total number of admitted recipients per group.

C. Simulation Results

1) Blocking Probability Comparisons: Fig. 10 shows the changes of the blocking probabilities for the three approaches as a function of group number: Fig. 10(a) for the flat model and Fig. 10(b) for the hierarchical model. Both depict that the blocking probabilities of the three approaches increase as group number increases, caused by a lack of network resources to satisfy the required QoS as the number of groups increases. Fig. 10 demonstrates that MQ has the lowest blocking probability among the three approaches in both types of network topologies. MQ outperforms RSVP with MOSPF because MQ considers QoS in routing while the latter has a high possibility of mismatch of the receivers' QoS with the network resources in the delivery tree. Besides, MQ has lower blocking probability than that of RSVP with QOSPF, by more than 25%. The reason is that RSVP with QOSPF is sender-oriented so that its path selection procedure may fail even if there are paths that meet the QoS requirements of those receivers with lower QoS. On the contrary, MQ is receiver-oriented and will find a feasible path if at least one meets its QoS requirements.

The performance advantage of QOSPF over MOSPF in terms of low blocking probability becomes significant only when the network is lightly loaded and thus the network resource is sufficient. Under the circumstance of light traffic, feasible paths may exist during the construction of a delivery tree. Since the construction of a QOSPF delivery tree is based on more prior

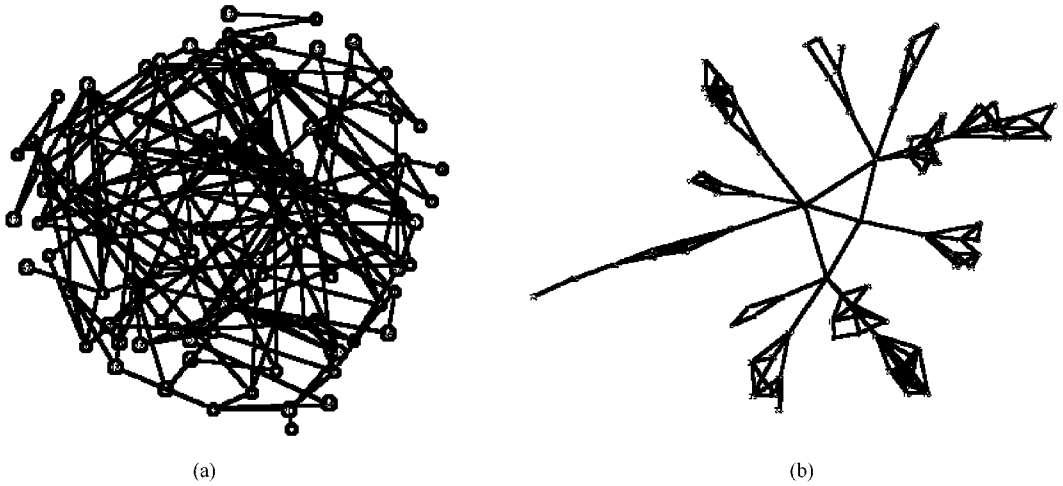

Fig. 9. Network topologies used in the simulation. (a) Flat graph model and (b) hierarchical graph model.

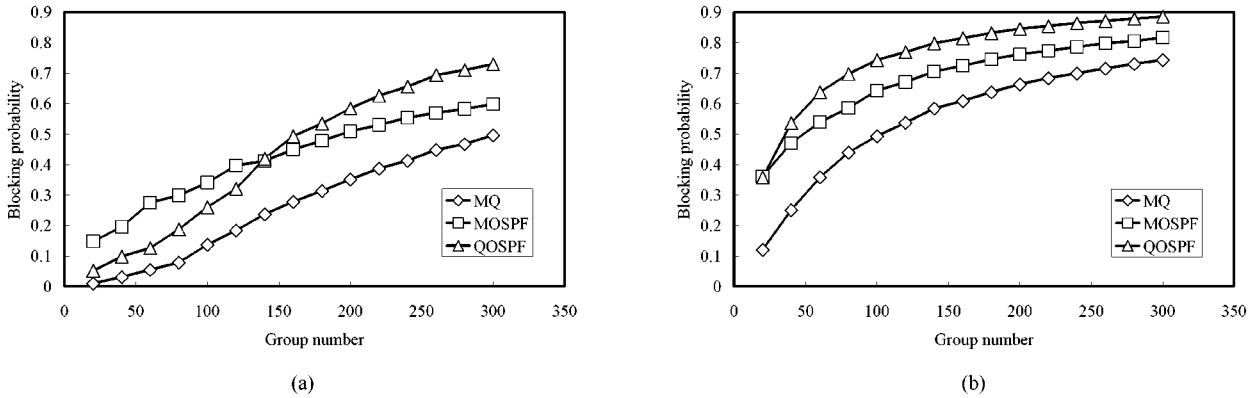

Fig. 10. Blocking probability comparisons. (a) Flat graph model and (b) hierarchical graph model.

knowledge about QoS information and network resources, so long as there is a feasible path from a source to any of the receivers, a recipient attempting to join the tree with the requested QoS must be admitted. To join an MOSPF-based QoS tree, on the other hand, a recipient is admitted only when the shortest path itself contains sufficient resources, irrespective of whether other paths with sufficient resources do exist. However, as the resources are consumed, MOSPF performs better than QOSPF, due to QOSPF quickly consuming system resources in an inefficient way (i.e., sender-oriented). The argument can be easily verified from Figs. 10 and 11 (Fig. 11 compares the resource utilization among the three approaches). The cross point of the QOSPF and MOSPF curves in Fig. 10(a) is approximately at the group number of 150 where the resource consumption curve of QOSPF in Fig. 11(a) becomes flat. In other words, in Fig. 10(a), when group number is less than 150, namely, the network is lightly loaded, the blocking probability of MOSPF is higher than that of QOSPF. However, to the right of the cross point, the network becomes heavily loaded, and hence MQSPF performs better than QOSPF.

2) Resource Utilization Comparisons: Figs. 11 and 12 show the comparisons of resource utilization for the three approaches, without and with being normalized by the total number of admitted users, respectively. Again both Figs. 11(a) and 12(a) are for the flat model, and both Figs. 11(b) and 12(b) are for the hierarchical model. Fig. 11 shows that MQ makes the best use of network resources, while MOSPF, the worse. In other words, MQ can make more efficient use of available resources to enhance system performance in terms of low blocking probability, while MOSPF does not have this advantage, irrespective of whether the network is still lightly loaded. QOSPF in Fig. 11 seems to make good use of network resources as well. However, as shown in Fig. 10, QOSPF has a rather high blocking probability. Besides, as shown in Fig. 12, when normalized by the total number of admitted users, each QOSPF recipient uses the most resources due to the QoS trees of QOSPF being constructed in an inefficient sender-oriented way, thus rendering highly blocking probability. Fig. 12 shows MOSPF consumes the lowest resources per admitted user among the three approaches, thanks to an MOSPF tree being constructed as the shortest path tree. Resource consumed by MQ is similar to that of MOSPF. Fig. 12 also shows resource consumed actually decreases as group number increases starting from a certain point [e.g., group number 100 in

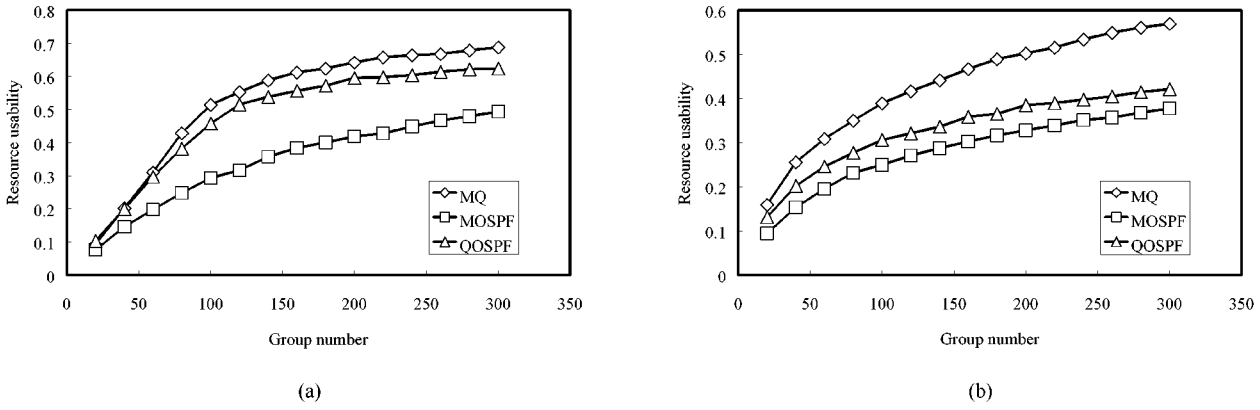

Fig. 11. Resource utilization comparisons. (a) Flat graph model and (b) hierarchical graph model.

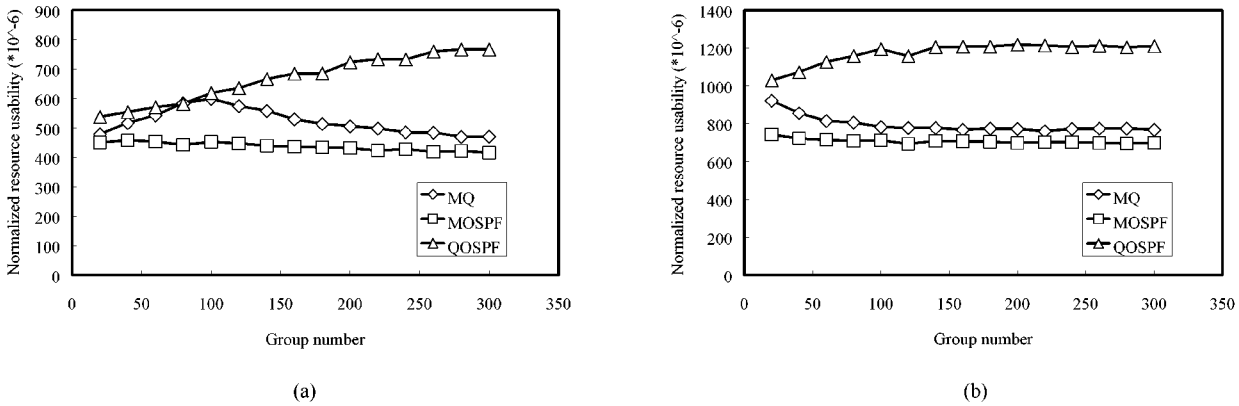

Fig. 12. Normalized resource comparisons. (a) Flat graph model and (b) hierarchical graph model.

Fig. 12(a)]. This is because in heavily loaded networks, while QOSPF keeps all users from joining the QoS tree, MQ still allows the users with smaller QoS requests to join.

3) Overhead Comparisons: Fig. 13 shows the comparison of protocol overhead for the three approaches. Fig. 13(a) shows the results of the flat model. Each approach sets up and maintains states by exchanging control messages to provide the requested QoS for the admitted receivers. Our simulation measured the number of control messages for tree construction, pruning, reshaping, and tree maintenance. The results were taken with a flow of two hours in length and with the states being refreshed every 30 s. In Fig. 13(a), QOSPF generates more control messages than MOSPF, because the normalized QOSPF tree is larger than that of MOSPF (as MOSPF tree is the shortest path tree) in size, and the control overhead is proportional to the number of links per tree. MQ outperforms the other two approaches, thanks to a smaller number of refresh messages. (MQ sends only one Refresh message, while RSVP sends both Path and Resv messages.) As a result, the savings obtained in using Refresh messages compensate for the extra messages (the use of the Join_Request, Join_Ack, Join_Fail, ResvRev, etc.) to create the tree. Note that MQ also multicasts Flow_Ad messages periodically, with a frequency less than Refresh, because it is only used for flow advertisement. Here, we show the results with the frequency of the Flow_Ads message one half that of Refresh messages. The overhead of MQ decreases as the multicast frequency of Flow_Ad decreases. Fig. 13(b) shows the case of the hierarchical model. Still, MQ has the lowest overhead, MOSPF is next, and QOSPF has the highest. Since bottleneck links exist in this topology, all the three overhead curves are a bit flatter as compared to those of the flat model.

V. Conclusions

In this paper, we have proposed MQ as an integrated mechanism to support multimedia group communications with end-to-end QoS guarantees for heterogeneous users. In MQ, while resource reservation is still de-coupled from routing, they are integrated in such way as to avoid the "sender-initiated" feature caused by employing RSVP used with multicast QoS routing. Being an integrated mechanism for multimedia multicasting, MQ enjoys scalability, robustness, efficiency, loop-freedom, and the support of user heterogeneity. When a new recipient joins a group of interest, if the reserved bandwidth in the current multicast tree cannot satisfy the QoS requirement of the joining recipient, MQ first attempts to expand the reservation upstream toward the source. Only when the original

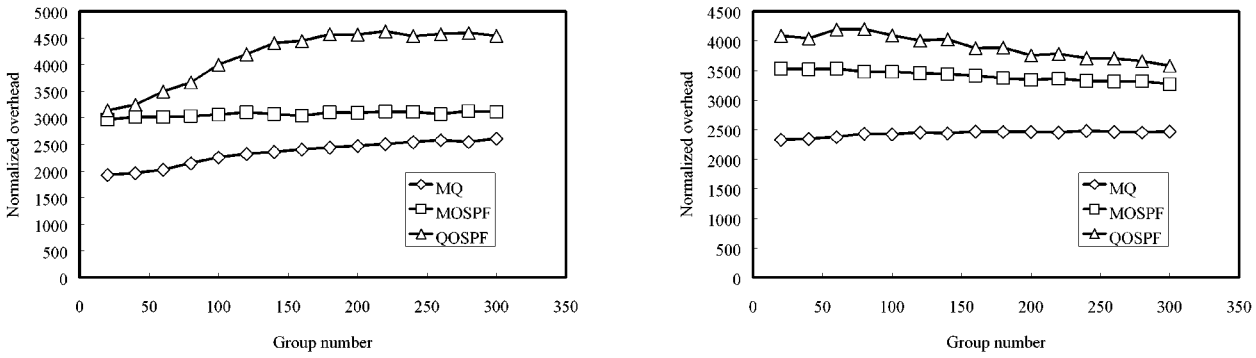

Fig. 13. Overhead comparisons, (a) flat graph model and (b) hierarchical graph model.

reservation plus the residual bandwidth in the current path cannot meet the QoS requirement of the new recipient will MQ consider growing a new branch. The tree may be reshaped upon a change in network condition or a departure of a recipient, in a way that minimizes the service disruption to the existing multicast tree. These features, together with a loop-free mechanism to ensure end-to-end QoS guarantees, allows MQ to make efficient use of network resources to minimize the blocking probability while supporting user heterogeneity with diverse QoS services, and to achieve high scalability with minimum information stored in the routing table.

To verify our conclusions, we have conducted simulations to evaluate the performance of MQ. Being a truly receiver-initiated, soft state, and integrated scheme for multicast QoS services, MQ demonstrates lower blocking probability for users to join the group of interest with requested QoS, much reduced protocol overhead, and more efficient resource utilization, as compared to traditional approaches that employ loosely coupled integration of IP multicasting, resource reservation, and QoS routing for multimedia group communications.

APPENDIX
MULTICAST QOS ROUTING

Metrics

The QoS routing module uses a link-state based routing mechanism modified from the Dijkstra algorithm. A feasible path satisfies the QoS requirement of a flow, which is determined by the flow spec, receiver capability, and available resources on the network. Each link is associated with two metrics: available bandwidth and propagation delay.

Link-State Information Dissemination

The link state information is advertised across the network so that each router can compute accurate and consistent QoS route for each flow. The link-state information distribution is similar to that of OSPF. Link-state advertisement is sent periodically or triggered by significant changes in the value of the metrics since the last advertisement.

QoS Routing Algorithm

Input (G, R, S, RB), where G is the network topology, R is the designated router of a joining receiver, S is the flow sender, and RB is the required bandwidth of R.

Output Bandwidth constrained delay optimal path from S to R, or ERROR when the path calculation fails.

$b_{i,j}$: Available bandwidth on link (i, j), set to 0 when there is no direct connection between nodes i and j.

$d_{i,j}$: Propagation delay on link (i, j), set to ∞ when there is no direct connection between nodes i and j.

D_i: Propagation delay of current chosen path from S to i, set to ∞ initially.

π_i: Previous router of router i on the chosen path from S to i; initially NIL.

$Adj(i)$: Set of routers which are adjacent to i.

Step 1) $L \leftarrow \{S\}$, $D_i \leftarrow d_{S,i}$ for $\forall\, i \in Adj(S)$.

Step 2) Find k so that $D_k = \min_{j \notin L} D_j$, $L \leftarrow L \cup \{k\}$.

Step 3) If $k = R$, return the path determined by π_R and previous routers of π_R, and stop.

Step 4) For $\forall\, i$, $i \in Adj(k)$, $b_{k,i} \geq RB$, $D_i > D_k + d_{k,i}$, and $i \notin L$, do $D_i \leftarrow D_k + d_{k,i}$, $\pi_i \leftarrow k$.

Step 5) Go to Step 2.

ACKNOWLEDGMENT

The authors would like to thank SiS Education Foundation for their support.

REFERENCES

[1] H. Eriksson, "MBone: The multicast backbone," *Commun. ACM*, vol. 37, no. 8, pp. 54–60, Aug. 1994.

[2] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP)—Version 1 functional spec,", IETF RFC 2205, Sept. 1997.

[3] L. Delgrossi and L. Berger, " Internet stream protocol—Version 2 (ST2) protocol specification,", IETF RFC 1819, Aug. 1995.

[4] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.

[5] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Network*, vol. 12, pp. 64–79, Nov./Dec. 1998.
[6] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communication," *IEEE/ACM Trans. Networking*, vol. 1, pp. 286–292, June 1993.
[7] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source-based algorithm for delay-constrained minimum-cost multicasting," in *IEEE INFOCOM '95*, vol. 1, pp. 377–385.
[8] S. Hong, H. Lee, and B. H. Park, "An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership," in *IEEE INFOCOM '98*, vol. 3, pp. 1433–1440.
[9] G. N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE J. Select Areas Commun.*, vol. 15, pp. 346–356, Apr. 1997.
[10] W. M. Moh and B. Nguyen, "An optimal QoS-guaranteed multicast routing algorithm with dynamic membership support," in *IEEE ICC '99*, vol. 2, pp. 727–732.
[11] Y. Im and Y. Choi, "A distributed multicast routing algorithm for delay-sensitive applications," in *IEEE Int. Conf. Parallel and Distributed Systems*, 1998, pp. 232–239.
[12] H.-C. Lin and S.-C. Lai, "VTDM—A dynamic multicast routing algorithm," in *IEEE INFOCOM '98*, vol. 3, pp. 1426–1432.
[13] L. Guo and I. Matta, "QDMR: An efficient QoS dependent multicast routing algorithm," in *IEEE Real-Time Technology and Applications Symp.*, 1999, pp. 213–222.
[14] J. C. Pasquale, G. G. Polyzos, and G. Xylmoenos, "The multimedia multicasting problem," *Multimedia Syst.*, vol. 6, no. 1, pp. 43–59, 1998.
[15] W. Fenner, "Internet group management protocol—Version 2,", IETF RFC 2236, Nov. 1997.
[16] J. Moy, "Multicast routing extensions for OSPF," *Commun. ACM*, vol. 37, no. 8, pp. 61–66, Aug. 1994.
[17] S. Deering, C. Partrige, and D. Waitzman, "Distance vector multicast routing protocol,", IETF RFC 1075, Nov. 1988.
[18] A. Ballaradie, J. Crowcroft, and P. Francis, "Core based tree (CBT)—An architecture for scalable inter-domain routing protocol," in *ACM SIGCOM '93*, Oct. 1993, pp. 85–95.
[19] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "An architecture for wide-area multicast routing," in *ACM SIGCOMM '94*, pp. 126–135.
[20] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *ACM SIGCOMM '98*, pp. 17–28.
[21] R. A. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Trans. Networking*, vol. 7, pp. 350–364, June 1999.
[22] J. Hou, H. Y. Tyan, and B. Wang, "QoS extension to CBT,", IETF Internet draft draft-hou-cbt-qos-00.txt, February 1999.
[23] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of service sensitive multicast Internet protoCol," in *ACM SIGCOMM '98*, pp. 144–153.
[24] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM '96*, Sept. 1996, pp. 117–130.
[25] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, "Quality of services extensions to OSPF,", IETF Internet draft, Work in progress, Sept. 1997.
[26] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, "QoS routing mechanisms and OSPF extensions,", IETF Draft (draft-guerin-qos-routing-ospf-04.txt), Dec. 1998.
[27] D.-N. Yang, W. Liao, and Y.-T. Lin, "MQ: An integrated mechanism for multimedia multicasting," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, New York, Aug. 2000.
[28] M. Doar and I. Leslie, "How bad is naïve multicast routing?," in *Proc. IEEE INFOCOM '93*, San Francisco, CA, 1993, pp. 82–89.
[29] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Trans. Networking*, vol. 4, no. , pp. 558–568, Aug. 1996.

De-Nian Yang was born in Taiwan, R.O.C., in 1977. He received the B.S. degree from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, in 1999. Exempted from the normal entrance exam to the graduate school, he is currently pursuing the Ph.D. degree in the Electrical Engineering Department, NTU.

His research interests include broadband Internet, QoS, and multicasting.

My. Yang received the Best Student Paper Award in ICME 2000.

Wanjiun Liao (S'96–M'97) was born in Taiwan, R.O.C., in 1968. She received the B.S. and M.S. degrees from National Chiao Tung University, Hsinchu, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1997.

She worked at the Telecommunication Labs of the Ministry of Transportation and Communications during 1992–1993. She joined the Department of Electrical Engineering, National Taiwan University, Taipei, as an Assistant Professor in 1997. Since August 2000, she has been an Associate Professor. Her research interests include broadband Internet, wireless Internet, and media streaming services.

Dr. Liao is a member of Phi Tau Phi, was a recipient of Acer Long-Term Thesis Award, the Chinese IEE (CIEE) Graduate Student Thesis Award, Outstanding Research Award at USC in 1997, and Best Student Paper Award in ICME 2000. She was selected as an outstanding Young Electrical Engineer by *EE Times* in 1997. She was also elected as one of the ten most outstanding young females in Taiwan in 2000.

Yen-Ting Lin was born in Taiwan, R.O.C., in 1975. He received the B.S. degree from the Department of Atmospheric Sciences and the M.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 1997 and 1999, respectively.

Mr. Lin received the Best Student Paper Award in ICME 2000.