

# QoS Extension to the Core Based Tree Protocol

Hung-Ying Tyan<sup>†</sup>, Jennifer C.Hou<sup>†</sup>, Bin Wang<sup>†</sup>, and Yao-Min Chen<sup>‡</sup>

<sup>†</sup>*Department of Electrical Engineering  
The Ohio State University  
Columbus, OH 43210  
{tyanh,jhou,bwang}@ee.eng.ohio-state.edu*

<sup>‡</sup>*Fujitsu Labs of America  
Sunnyvale, CA 94086  
ychen@fla.fujitsu.com*

## Abstract

*This paper describes extension to the core based tree (CBT) protocol to maintain a multicast tree with user-specified QoS properties. Specifically, it describes enhancements in the member join/leave and state update/refresh procedures to facilitate the deployment of additive (e.g., end-to-end delay bound), multiplicative (e.g., packet loss ratio along a path) and concave (e.g., minimum bandwidth available) QoS.*

*Eligibility tests are devised to verify whether or not a new member can join a multicast tree at adequate QoS, while not violating the QoS received by on-tree members. Management of router state is based on a simple state update and refresh procedure that can be readily integrated with the tree maintenance mechanism that exists in CBT (i.e., echo-requests and echo-replies).*

## 1 Introduction

Motivated by the enriching content of multimedia applications and by the changing requirements and functionalities at the user side, multicast service with different QoS requirements for multiple-point applications has become increasingly demanding. The main intent of this paper is thus to present a set of QoS enhancements in the member join/leave and state update/refresh procedures, to allow QoS deployment in the Core Based Tree (CBT) protocol [1, 2], with the minimal impact to the existing infrastructure.

The construction of multicast trees can be classified into *source-based* tree approaches and *core-based* tree approaches [3]. For example, Distance-Vector Multicast Routing Protocol (DVMRP) [11], Multicast extensions to Open Shortest Path First Protocol (MOSPF) [12], and Protocol Independent Multicast Dense Mode (PIM-DM) [13] fall in the category of source-based tree approaches, while the Core Based Tree (CBT) protocol [1, 2], the Protocol Independent Multicast Sparse Mode (PIM-SM)<sup>1</sup> [13, 14], and very recently, Simple Multicast (SM) [10] are representatives of core-based tree ap-

proaches. From the viewpoint of network management, core-based trees offer more favorable scaling characteristics than source-based trees (by a factor of the number of active sources). Moreover, routers that are not on a multicast tree are not involved in group membership maintenance activities. The price core-based multicast routing has to pay is, however, that the resulting multicast tree may be sub-optimal with respect to some source(s).

One important issue that is not addressed in core-based multicast routing is how to provide QoS in the form of additive QoS (e.g., end-to-end delay bound), multiplicative QoS (e.g., maximum packet loss ratio), and concave QoS (e.g., minimum bandwidth available) [4], and their variations. In this paper, we consider these QoS parameters and devise a unified QoS extension framework based on the CBT protocol. In particular, we (i) devise eligibility tests to verify whether or not a new member can join a multicast tree at adequate QoS, while not violating the existing QoS to the other on-tree members; (ii) determine the set of states needed to conduct eligibility tests; and (iii) devise a state update/refresh procedure that is based on soft state and can be readily integrated with the tree maintenance mechanism that already exists in CBT, e.g., sending of *echo-requests* and *echo-replies* in CBT.

Although we focus on CBT in this paper, the proposed QoS extension can be applied to any bi-directional multicast routing protocol with an explicit member join procedure and a soft state refresh procedure, such as Simple Multicast [10].

The rest of the paper is organized as follows. In Section 2, we present the network model considered and the QoS supported. The CBT protocol is summarized and an overview of the QoS framework is given there. In Sections 3–6, we present the QoS extension in the cases of additive QoS parameters, multiplicative QoS parameters, concave QoS parameters, and the maximum inter-destination difference of (additive/multiplicative) QoS parameters along the paths from a source to any two receivers, respectively. In Section 7, we present, based on the soft state concept, the state update and refresh procedure. In Section 8, we discuss how to extend the work in Sections 3–5 to the case of heterogeneous receiver-initiated QoS requirements. In Section 9, we evaluate

The work reported in this paper was supported in part by DARPA under Contract No. N66001-97-C-8526 and by NSF under Grant No. NCR-9625064. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

<sup>1</sup>PIM-SM includes a mechanism to switch to a source-specific

shortest path tree in the case that the data rate of a source exceeds some threshold.

the effectiveness, and the message overhead, of the QoS extension in terms of the probability of constructing feasible multicast trees, message overheads, and scalability. We conclude the paper in Section 10.

## 2 Backgrounds

### 2.1 Network Model

We consider a network that supports both best-effort traffic and traffic with QoS requirements. The way in which network resources are split between the two classes is irrelevant to the paper, except for the assumption that each router in the network supports the QoS extension and is able to identify and advertise to their immediate neighbors the QoS parameters of interest experienced by packets when traversing links emanating from the router.

We represent a network by a weighted digraph  $G = (V, E)$ , where  $V$  denotes the set of routers and  $E$  the set of network communication links connecting the routers. We define a link-delay (available bandwidth) function  $f_D : E \rightarrow R^+$  ( $f_B : E \rightarrow R^+$ ) which assigns a non-negative weight to each link in the network. The value of  $f_D(\ell)$  is a measure of the delay which packets experience on link  $\ell$  in  $E$ . (The value of  $f_B(\ell)$  is a measure of the bandwidth available on link  $\ell$  in  $E$ .) Similarly, we define a packet loss function  $p_L : E \rightarrow [0, 1]$  which assigns a non-negative fractional weight to each link in the network. The value of  $p_L(\ell)$  is the probability of packet loss on link  $\ell$  in  $E$ .

We denote as  $M_g$  (which is a subset of  $V$ ) a set of routers to which group- $g$  hosts are attached. For notational simplicity, we call the set  $M_g$  a multicast group with each router  $v$  in  $M_g$  as a group member. (Actually the multicast group is the set of hosts that are directly attached to routers in  $M_g$ .) We consider the most general many-to-many multicast paradigm in which each router  $v$  in  $M_g$  can be a source, in addition to being a receiver in the group. Let  $M_s$  denote the set of group members that are also sources. Packets originating from router  $v$  in  $M_s$  have to be delivered to a set of receiver routers  $M_g - \{v\}$ . We use  $P_T(v_s, v_d)$  to denote the path from a source router  $v_s$  to a receiver router  $v_d$  in  $M - \{v_s\}$  in the tree  $T$ . We also use “ $x \in P_T(v_s, v_d)$ ” to denote that  $P_T(v_s, v_d)$  traverses either router  $x$  or link  $x$ .

### 2.2 QoS Parameters Considered

We define  $m(\ell)$  as a QoS metric for link  $\ell$ . For any path  $P_T(u, v) = (u, i, j, \dots, k, v)$ , we say metric  $m$  is additive if

$$m(u, v) = m(u, i) + m(i, j) + \dots + m(k, v). \quad (2.1)$$

For example, the end-to-end delay  $d(u, v)$ , which packets forwarded from router  $u$  to router  $v$  experience, is additive and is equal to the sum of individual link metric  $d(i, j)$  along the path  $P_T(u, v)$ .

We say metric  $m$  is multiplicative if

$$m(u, v) = m(u, i) \times m(i, j) \times \dots \times m(k, v). \quad (2.2)$$

For example, the probability,  $1 - p_L(u, v)$ , for a packet to reach router  $v$  from router  $u$  along  $P_T(u, v)$  is multiplicative and is equal to the product of individual link metric  $1 - p_L(i, j)$  along the path  $P_T(u, v)$ .

We say metric  $m$  is concave if

$$m(u, v) = \min[m(u, i), m(i, j), \dots, m(k, v)]. \quad (2.3)$$

For example, the bandwidth  $b(u, v)$ , available along a path from router  $u$  to router  $v$ , is concave and is equal to the minimum bandwidth  $f_B(i, j)$  among the links in path  $P_T(u, v)$ .

A QoS requirement may be specified as (i) an upper/lower bound on an additive, multiplicative, or concave QoS parameter, e.g., an upper bound on the end-to-end delay, an upper bound on the packet loss ratio, or a lower bound on the bandwidth available, from any source to a receiver; or (ii) an upper bound on the inter-destination difference of an (additive or multiplicative) QoS parameter along the paths from a source to any two receivers, e.g., an upper bound on the end-to-end inter-destination delay jitter defined as the difference between the end-to-end delays along the paths from a source router to any two receiver routers.

The need for a bounded end-to-end delay, bounded probability of packet loss, and minimal available bandwidth has been well justified [5, 6]. The situation in which a bounded inter-destination delay jitter among all the group members arises is also not rare [7]. One possible scenario occurs during a teleconference in which any current speaker should be heard by all participants at approximately the same time to achieve the feeling of multi-party interactive face-to-face discussions. Another application domain is the distributed interactive simulation in which an inter-destination delay jitter bound is needed to constrain the time during which the simulation engines are in inconsistent states.

### 2.3 The Core-Based Tree Protocol

In the CBT protocol, one router for each group is selected as the core [1, 2] (or termed in [14] as a rendezvous point) for the group. A tree rooted at the core is then constructed to span all the group members. A host first expresses its interest in joining a group by multicasting an IGMP host membership report [8] to its local router which then sends a *join-request* message to the next hop on the path toward the group’s core router. The *join-request* sets up *transient* join state (in the form of  $\langle \text{group, downstream interface, upstream interface} \rangle$ ) in the routers it traverses. If the transient join state is not “confirmed” with a *join-acknowledgment* message from upstream, the state is eventually timed out. The *join-request* message travels hop-by-hop toward the core until it reaches the core or an on-tree router, at which point a *join-acknowledgment* message is sent back along the reverse path, forming a new branch from the tree

to the requesting router. When a router receives a *join-acknowledgment* message, it updates its forwarding cache to reflect the fact that it now becomes an on-tree router, and forwards the *join-acknowledgment* message back to the requesting router. The state created in the routers by the sending/receiving of a *join-acknowledgment* is group specific — it consists of the group address and a list of local interfaces over which *join-requests* for the group have previously been acknowledged.

“Tree maintenance” is achieved by having each downstream router periodically send a CBT “keepalive” message (i.e., *echo-request*) to its parent router on the tree. The receipt of a keepalive message over a valid child interface prompts a response (i.e., *echo-reply*) that carries a list of groups for which the corresponding interface is a child interface. If no response is forthcoming within *group\_expire\_time* seconds, (e.g., a router’s upstream neighbor becomes unreachable), the router sends a *quit-notification* message upstream, and flushes all of its downstream branches by sending *flush-tree* messages, allowing them to individually rejoin if necessary. In the case that a member leaves the group, if the local router to which the leaving member is attached does not have any other directly attached members or downstream on-tree routers, the router sends a *quit-notification* message to its parent router on the tree and deletes the corresponding forwarding cache.

During the data transmission phase, data packets flow from any source to its parent and children. The parent router forwards packets to all the children other than the source and to its parent until data packets reach the core. Data packets are then sent down all the other branches, ensuring that all group members receive them.<sup>2</sup>

To accommodate the situation in which a source is not on the multicast tree, the local router to which the sender host is attached encapsulates the data packet and unicasts it to the core, where it is decapsulates and disseminated over the tree.

**Why the CBT protocol is selected as the example protocol:** The reason why we choose the CBT protocol as the example protocol is three-fold: (1) as discussed in Section 1, the core-based tree approach is more scalable than the source-based approach; (2) CBT is simple<sup>3</sup>, elegant, and has explicit member join/leave and state refresh procedures; and (3) CBT is receiver-initiated, and is thus especially well-suited to support heterogeneous receiver reservation.

The major drawback of the CBT protocol (and any core-based tree protocol) is that it concentrates traffic from multiple sources on a few links that are part of

<sup>2</sup>The way a source host sends to a group in PIM-SM is slightly different. In PIM-SM, a source router initially encapsulates data packets in register messages and unicasts them to the RP. The RP decapsulates each register message and forwards the data packet natively to downstream members. PIM-SM also includes a mechanism for switching to a source-specific shortest path tree in case that the data rate of a source exceeds some threshold.

<sup>3</sup>CBT defines only 7 types of messages.

the CBT tree. When the reservation model is not in the shared mode (e.g., the fixed filter reservation style for multiple senders), the network resources (e.g., bandwidth) of certain on-tree links may have been exhausted by group members that are already on-tree. If a joining group member whose shortest route to the core contains these on-tree links, either the QoS of existing on-tree members degrades or the QoS required by this member cannot be fulfilled. The admission tests proposed in this paper can be used to tackle this problem: they will detect occurrence of such conditions and reject the new member. The new member may then attempt on an alternative route to join the tree. This requires modification of the off-tree route search stage of the CBT protocol and is beyond the scope of this paper.

Another drawback of the CBT protocol is that although CBT uses “keepalive” messages to refresh group membership, it does not refresh the on-tree routes between a group member and the core. That is, once the multicast tree is built, it never changes even if the underlying unicast route from a group member to the core has changed in adaption to load changes. To deal with this problem, we propose to include in state-refresh messages the QoS information collected by the underlying unicast protocol. An on-tree router is then able to know, upon receipt of a state-refresh message, whether or not the QoS requirement is still met. If not, the router that detects the QoS violation may simply flush the subtree below it and ask downstream group members to individually rejoin the multicast tree. More details on state update and refresh will be given in Section 7.

## 2.4 Overview of the QoS Extension

We introduce the following QoS extension to the current CBT specification: each *join-request* message carries, in addition to the interface information, the QoS parameters of interest. When a *join-request* message reaches the core or an on-tree router, the core/on-tree router performs a set of eligibility tests. Although it is preferable that eligibility tests be conducted locally, the on-tree router may not be able to approve a *join-request* based only on its local state, and may have to collaborate with other on-tree routers to conduct the eligibility tests. Moreover, the state kept at some other on-tree routers may have to be changed because of the member join. Only after the branch survives the eligibility tests will it be eligible to join the multicast tree (in which case a *join-acknowledgment* message is then sent back). In the case of member leave, the state kept at the other on-tree routers may have to be updated and proper procedures be taken to notify the other on-tree routers of the need to update their states. Note that while changes to the current CBT protocol specification seem unavoidable, we have attempted to keep them as small as possible.

To realize the above QoS extension, we consider, for each type of QoS requirements, the following issues:

1. What is the minimum set of states each on-tree

router has to keep in order to conduct the proposed eligibility tests?

2. What are the eligibility tests conducted (sometimes collaboratively among on-tree routers) at the time of member join?
3. How is the set of states kept at each on-tree router updated in response to member leave?
4. How is the set of states maintained and refreshed at each on-tree router?

### 3 The QoS Extension in the Case of Imposing a Bound on Additive QoS Parameters

We first consider the case in which the QoS requested is expressed in terms of additive QoS parameters. Without loss of generality, we use the end-to-end delay as an example, and impose an upper bound  $D$  on the acceptable end-to-end delay perceived by a receiver router along any path from a source router to the receiver router in a multicast group.

#### 3.1 The State Kept at Each On-Tree Router

To satisfy the end-to-end delay bound, each on-tree router  $u$  keeps the following states for each downstream interface:

- (1)  $d_i^{max}(u, *)$ : the maximum delay among the on-tree paths from router  $u$  to the downstream on-tree group members reachable on interface  $i$ , where interface  $i$  is a downstream interface (recall that downstream is defined with respect to the core).
- (2)  $d_i^{max}(*, u)$ : the maximum delay among the on-tree paths from all the downstream on-tree source group members to router  $u$  reachable on a downstream interface  $i$ .

The per-node state  $d^{max}(u, *)$  is naturally defined as the maximum value of  $d_i^{max}(u, *)$ 's for all  $i$ . Similarly,  $d^{max}(*, u)$  is defined as the maximum value of  $d_i^{max}(*, u)$ 's for all  $i$ . In addition, we define the maximum outgoing/incoming delay between router  $u$  and all its downstream on-tree group members *except those reachable on interface  $\ell$*  as

$$d_{I \setminus \{\ell\}}^{max}(u, *) \triangleq \max_{i \in I \setminus \{\ell\}} d_i^{max}(u, *)$$

and

$$d_{I \setminus \{\ell\}}^{max}(*, u) \triangleq \max_{i \in I \setminus \{\ell\}} d_i^{max}(*, u),$$

where  $I$  is the set of downstream interfaces of router  $u$ . Let  $T_s(u)$  denote the subtree rooted at router  $u$ , then each on-tree router only keeps the state information for  $T_s(u)$ . The reason for keeping per-downstream-interface (instead of per-node) state will be clearer when we discuss the member join/leave procedure in the following sections.

#### 3.2 Eligibility Tests and Member Join Procedure

When a *join-request* message from a joining router  $v$  arrives at an on-tree router  $u$  on interface  $i$ , router  $u$  checks if

$$d^{max}(*, v) = d(u, v) + d_{I \setminus \{i\}}^{max}(*, u) \leq D. \quad (3.1)$$

In addition, if the joining member  $v$  is also a source, router  $u$  further checks if

$$d^{max}(v, *) = d(v, u) + d_{I \setminus \{i\}}^{max}(u, *) \leq D. \quad (3.2)$$

Both parameters  $d_{I \setminus \{i\}}^{max}(u, *)$  and  $d_{I \setminus \{i\}}^{max}(*, u)$  are obtained from the states kept at router  $u$ . Both parameters  $d(v, u)$  and  $d(u, v)$  can be carried in the *join-request* message and updated as the message travels from router  $v$  to router  $u$ .

If Eq. (3.1) holds, it implies the QoS requirement of the new member  $v$  is fulfilled by all the source group members in  $T_s(u)$ . Similarly, if Eq. (3.2) holds, it implies the QoS requirements for the group members in  $T_s(u)$  are not violated due to join of the new source member  $v$ .

If Eq. (3.1) (or Eq. (3.2) in the case that router  $v$  is also a source) does not hold, the join request is immediately rejected and a *rejection-reply* message is sent back to the joining router  $v$ . Otherwise, router  $u$  forwards upstream to its parent router  $w$  the *join-request* with the updated accumulative delay information ( $d(w, u) + d(u, v)$  and, in addition,  $d(v, u) + d(u, w)$  if router  $v$  is also a source). Upon receipt of the *join-request* on interface  $i$ , router  $w$  conducts the eligibility test (i.e., Eqs. (3.1)–(3.2) except that  $u$  is replaced by  $w$  in the expressions) to verify if join of router  $v$  violates the QoS requirements of all the on-tree group members in  $T_s(w)$  as well as if the QoS requirement of router  $v$  is fulfilled by all the on-tree source members on  $T_s(w)$ . If the eligibility test succeeds, router  $w$  forwards upstream to its parent router the *join-request* with the updated accumulative delay information; otherwise, it sends downstream to its child router a *rejection-reply* message.

The process repeats until the *join-request* is either rejected at some upstream on-tree router or forwarded to, and approved by, the core, whichever occurs first. In the former case, the on-tree router  $u$  where the *join-request* arrives initially sends back a *rejection-reply* message to the joining router  $v$ . In the latter case, the core sends back a *join-acknowledgment* message along the reverse on-tree path, and each on-tree router  $w$  on the path between router  $u$  and the core updates its state upon receipt of this *join-acknowledgment* message from upstream, if  $d(w, v) > d_i^{max}(w, *)$  and/or  $d(v, w) > d_i^{max}(*, w)$ , where interface  $i$  is the interface on which the corresponding *join-request* arrived. Finally, router  $u$  sends back a *join-acknowledgment* message.

In short, each on-tree router  $u$  keeps the state information only for its subtree. When a *join-request* arrives at an on-tree router  $u$ , it has to pass a sequence of eligibility tests, where each test is performed router by router on the tree-path from router  $u$  to the core.

Note that to avoid potential QoS conflicts among multiple members that intend to join a multicast group at the same on-tree router, an on-tree router will not process the next request until it finishes processing the current *join-request* and sends back either a *rejection-reply* message or a *join-acknowledgment* message. A new *join-request* may be blocked from the time when a previous *join-request* is forwarded upstream and the corresponding response returns. We argue that since only the on-tree routers between router  $u$  and the core are involved in jointly approving a *join-request*, this transient period cannot be prohibitively long.

### 3.3 Member Leave Procedure

When a member router  $v$  leaves a multicast tree, it sends a *quit-notification* message (with the accumulative delay information  $d(v, u)$  and  $d(u, v)$ ) to its parent router  $u$ . Upon receipt of a *quit-notification* message on interface  $i$ , if router  $u$  does not have any other local members on the directly attached subnet or downstream on-tree routers, it sends a *quit-notification* message to its parent router  $w$  and deletes the corresponding forwarding cache. Otherwise, router  $u$  first deletes from the forwarding cache interface  $i$ , and then checks if

$$d(u, v) < d_{I \setminus \{i\}}^{max}(u, *), \quad (3.3)$$

and in addition, in case that router  $v$  is also source router,

$$d(v, u) < d_{I \setminus \{i\}}^{max} i(*, u). \quad (3.4)$$

If Eq. (3.3) (and, in addition, Eq. (3.4), in the case that router  $v$  is also a source router) holds, it implies the state kept at router  $u$  as well as other upstream on-tree routers will not change as a result of this member leave. Otherwise, router  $u$  first updates its per-node state to reflect the fact that router  $v$  has left the multicast tree and sends to its parent router  $w$  a *leave-update* message that carries  $d(u, v) + d(w, u)$  (and in addition,  $d(v, u) + d(u, w)$ , if the leaving member is also a source). Upon receipt of a *leave-update* message on interface  $i$ , router  $w$  checks whether or not its state is affected by the member leave (i.e., Eqs. (3.3)–(3.4) except that  $u$  is replaced by  $w$ ). The update process repeats until either the test succeeds at some upstream router or the *leave-update* message reaches the core.

The reason why on-tree routers have to keep per-interface state should now be clear. If only per-node state were kept, it would be impossible to obtain  $d_{I \setminus \{\ell\}}^{max}(u, *)$  or  $d_{I \setminus \{\ell\}}^{max}$  needed in Eqs. (3.1)–(3.4) for the member join/leave procedures.

## 4 The QoS Extension in the Case of Imposing a Bound on Multiplicative QoS Parameters

We now consider the case in which the QoS requested is expressed in terms of multiplicative QoS parameters. Recall that for any path  $P_T(u, v) = (u, i, j, \dots, k, v)$ , we say metric  $m$  is multiplicative if it satisfies Eq. (2.2). Taking logarithm of Eq. (2.2), we have

$$\log m(u, v) = \log m(u, i) + \log m(i, j) + \dots + \log m(k, v).$$

That is, the QoS extension discussed in Section 3 can be readily applied if the state kept at each on-tree router is in the logarithmic form of a multiplicative QoS metric. Without loss of generality, we use the packet loss ratio as an example and impose an upper bound,  $1 - L$  ( $0 \leq L < 1$ ), on the acceptable packet loss ratio along an on-tree path leading to the receiver router.

### 4.1 The State Kept at Each On-Tree Router

The packet loss ratio,  $p_L(u, v)$ , along a path from router  $u$  to router  $v$  can be expressed as

$$1 - p_L(u, v) = \prod_{l \in P_T(u, v)} (1 - p_L(l)).$$

Taking logarithm of the above expression, we have

$$\log(1 - p_L(u, v)) = \sum_{l \in P_T(u, v)} \log(1 - p_L(l)).$$

We define the successful transmission index,  $s(u, v)$ , on the path  $P_T(u, v)$  as  $s(u, v) = \log(1 - p_L(u, v))$ .

To satisfy the upper bound on the packet loss ratio, each on-tree router  $u$  keeps the following state for each downstream interface:

- (1)  $s_i^{min}(u, *)$ : the minimum value of  $\log(1 - p_L(u, v))$  among the on-tree paths from router  $u$  to the downstream on-tree group routers  $v$  reachable on a downstream interface  $i$ .
- (2)  $s_i^{min}(*, u)$ : the minimum value of  $\log(1 - p_L(v, u))$  among the on-tree paths from the downstream on-tree source routers  $v$  to router  $u$  reachable on interface  $i$ .

Also,  $s^{min}(u, *)$ ,  $s^{min}(*, u)$ ,  $s_{I \setminus \{i\}}^{min}(u, *)$ , and  $s_{I \setminus \{i\}}^{min}(*, u)$  are defined in a similar way as  $d^{max}(u, *)$ ,  $d^{max}(*, u)$ ,  $d_{I \setminus \{i\}}^{max}(u, *)$ , and  $d_{I \setminus \{i\}}^{max}(*, u)$  are defined.

### 4.2 Eligibility Tests and Member Join Procedure

Now when a *join-request* message from a joining router  $v$  arrives at an on-tree router  $u$  on interface  $i$ , router  $u$  checks if

$$s^{min}(*, v) = s(u, v) + s_{\Gamma \setminus \{i\}}^{min}(*, u) \geq \log L. \quad (4.1)$$

In addition, if router  $v$  is also a source, router  $u$  further checks if

$$s^{min}(v, *) = s(v, u) + s_{\Gamma \setminus \{i\}}^{min}(u, *) \geq \log L. \quad (4.2)$$

Both parameters,  $s(u, v) = \log(1 - p_L(u, v))$  and  $s(v, u) = \log(1 - p_L(v, u))$ , can be carried in the *join-request* message and updated as the message travels from router  $v$  to router  $u$ .

The process of conducting the eligibility test (Eqs. (4.1)–(4.2)), forwarding the join request upstream toward the core, and updating state in the case that the eligibility test succeeds is virtually the same as that described in Section 3.2.

### 4.3 Member Leave Procedure

The procedure taken upon receipt of a *quit-notification* message is virtually the same as that described in Section 3.3, except that Eqs. (3.3)–(3.4) are now replaced by

$$s(u, v) > s_{\Gamma \setminus \{i\}}^{min}(u, *), \quad (4.3)$$

and

$$s(v, u) > s_{\Gamma \setminus \{i\}}^{min}(*, u). \quad (4.4)$$

## 5 The QoS Extension in the Case of Imposing a Bound on Concave QoS Parameters

We now consider the case in which the QoS requested is expressed in terms of concave QoS parameters. Without loss of generality, we use the bandwidth available along a path as an example and impose a lower bound  $B$  on the minimum bandwidth that must be available along any path from a source router to any receiver router in a multicast group.

### 5.1 The State Kept at Each On-Tree Router

To satisfy the minimum bandwidth requirement  $B$  for a multicast group, each on-tree router  $u$  keeps only the state  $B$  and a counter  $C$  of how many source routers currently exist in the subtree  $T_s(u)$  rooted at router  $u$  (initially  $C = 0$ ).

### 5.2 Eligibility Tests and Member Join Procedure

When a *join-request* message from a joining router  $v$  arrives at an on-tree router  $u$ , router  $u$  checks if

$$b(u, v) \geq B, \quad (5.1)$$

and in addition, if router  $v$  is also a source, router  $u$  further checks if

$$b(v, u) \geq B. \quad (5.2)$$

Both parameters,  $b(v, u)$  and  $b(u, v)$ , can be carried in the *join-request* message and updated as the message travels from router  $v$  to router  $u$ .

If Eq. (5.1) (or, in addition, Eq. (5.2) in the case that router  $v$  is also a source) does not hold, the *join-request* is immediately rejected and a *rejection-reply* message is sent back to the joining router  $v$ . Otherwise, we consider two cases:

(C1) In the case that router  $v$  is only a receiver, router  $u$  sends back a *join-acknowledgment* message immediately to reserve bandwidth of amount  $B$ , along the reverse path, for the multicast group.<sup>4</sup> There is no need to forward the *join-request* upstream, because the first receiver router in the subtree  $T_S(u)$  rooted at router  $u$ , has reserved bandwidth of amount  $B$  on the path from the core to router  $u$ .

(C2) In the case that router  $v$  is both a receiver and a source, if the counter  $C$  is zero (i.e., no source router exists in the subtree  $T_s(u)$  so far), router  $u$  (i) reserves bandwidth of amount  $B$  on the upstream on-tree link  $(u, w)$ , (ii) forwards upstream to its parent router  $w$  the *join-request* in order to reserve the bandwidth of amount  $B$  on the path from router  $w$  to the core for the multicast group; and (iii) increments the counter  $C$  by one. The process repeats until the *join-request* is either rejected because of insufficient bandwidth on its upstream on-tree link or approved at some upstream on-tree router  $x$ , whichever occurs first. In the former case, each on-tree router  $w$  on the path between router  $u$  and router  $x$  releases the reserved bandwidth in the upstream direction, and router  $u$  sends back a *rejection-reply* message to the joining router  $v$ . In the latter case, router  $u$  sends back a *join-acknowledgment* message to reserve bandwidth of amount  $B$  along the reverse path.

On the other hand, if some source router(s) exist in the subtree  $T_s(u)$  and the bandwidth of amount  $B$  has been reserved on the path from router  $u$  to the core by the first source router in  $T_s(u)$ , router  $u$  sends back a *join-acknowledgment* to reserve bandwidth of amount  $B$ , along the reverse path to the requesting router.

### 5.3 Member Leave Procedure

When a member router  $v$  leaves a multicast tree, it sends a *quit-notification* message to its parent router  $u$ . If router  $u$  does not have any other local members or

<sup>4</sup>Note that if bandwidth reservation is not combined with routing, and a separate signaling protocol such as RSVP [9] is used for resource reservation, then the bandwidth is not reserved. In this paper, we assume that bandwidth reservation is coupled with routing.

downstream on-tree routers, it sends a *quit-notification* message to its parent router  $w$  and deletes the corresponding forwarding cache. Otherwise, router  $u$  deletes from the forwarding cache the interface on which the *quit-notification* message arrives. In the latter case, if router  $v$  is also a source, router  $u$  decreases its counter  $C$  by one. If after decrement,  $C$  becomes zero, router  $u$  releases the bandwidth of amount  $B$  on link  $(u, w)$ , and sends to its parent router  $w$  a *leave-update* message to release bandwidth reserved in the upstream direction. Upon receipt of a *leave-update* message, router  $w$  (i) decrements its counter  $C$  by one, and (ii) releases bandwidth and forwards the *leave-update* message upstream if  $C$  hits zero. The update process repeats until either the *leave-update* message reaches the core or stops at some upstream router with more than 1 source router in its subtree.

## 6 The QoS Extension in the Case of Imposing a Bound on the Inter-Destination Difference of QoS Parameters

Without loss of generality, we use the inter-destination delay jitter as an example and impose an upper bound  $\delta$  on the acceptable difference between the end-to-end delays along the paths from a source router to any two receiver routers in a multicast group.

### 6.1 Eligibility Tests

Eligibility tests in the case of inter-destination delay jitter are more complicated, and have to be designed with differentiation between the cases that a *join-request* is sent by a receiver router or by a receiver router that is also a source.

#### 6.1.1 Eligibility Tests in the Case of Receiver Member Join

One may be tempted to think that similar to Sections 3–6 eligibility tests as simple as

$$d(u, v) - d_{I \setminus \{i\}}^{\min}(u, *) \leq \delta \quad (6.1)$$

and

$$d_{I \setminus \{i\}}^{\max}(u, *) - d(u, v) \leq \delta \quad (6.2)$$

would suffice when a *join-request* from a requesting router  $v$  arrives at an on-tree router  $u$  on interface  $i$ . However, the above criterion only establishes the QoS for sources that are outside the subtree  $T_s(u)$  rooted at router  $u$ . To demonstrate this, consider the scenario in Fig. 1 (assuming each link incurs 1 unit time of delay). Suppose  $\delta = 2$ . Now router  $v$  intends to join a multicast tree at router  $u$ . Since the state kept at router  $u$  is  $d_{I \setminus \{i\}}^{\max}(u, *) = d(u, s) = 4$  and  $d_{I \setminus \{i\}}^{\min}(u, *) = d(u, x) = 3$ , the *join-request* passes the eligibility test (Eq. (6.1)), but the QoS requirement with respect to the source router  $s$  is not met ( $d(s, v) - d(s, x) = 6 - 3 = 3 > \delta$ ). This problem may arise when there exist source routers

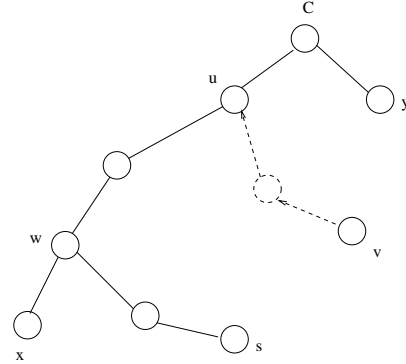


Figure 1: A scenario in which Eqs. (6.1)–(6.2) do not suffice to serving as eligibility tests. Routers  $x$ ,  $y$ , and  $s$  are group receiver routers and in addition router  $s$  is also a source router. Router  $v$  intends to join the multicast tree at router  $u$ .

in the subtree rooted at router  $u$ . To deal with this problem, each router  $u$  has to keep certain state information of downstream source routers. Take Fig. 1 as an example. Suppose now router  $u$  keeps, in addition to its own state,  $d_{I \setminus \{i\}}^{\min}(w, *) - d(w, u)$  and  $d_{I \setminus \{i\}}^{\max}(w, *) - d(w, u)$ . Upon receipt of a *join-request* from router  $v$ , router  $u$  checks on behalf of router  $w$  and hence source  $s$  if

$$d(u, v) - (d_{I \setminus \{i\}}^{\min}(w, *) - d(w, u)) = d(w, u) + d(u, v) - d_{I \setminus \{i\}}^{\min}(w, *) \leq \delta \quad (6.3)$$

and

$$(d_{I \setminus \{i\}}^{\max}(w, *) - d(w, u)) - d(u, v) = d_{I \setminus \{i\}}^{\max}(w, *) - (d(w, u) + d(u, v)) \leq \delta \quad (6.4)$$

in addition to Eqs. (6.1)–(6.2). In other words, router  $u$  checks, on behalf of router  $w$ , whether or not Eqs. (6.3)–(6.4) hold at router  $w$ .

Now there are three problems that should be addressed in order to formalize the eligibility tests and the member join procedure:

- (P1) does router  $u$  have to keep the state information for every source router in its subtree?
- (P2) can the eligibility test of Eqs. (6.1)–(6.2) be consolidated with that of Eqs. (6.3)–(6.4)?
- (P3) how does router  $u$  obtain the state information of downstream source routers?

The answer to (P1) is fortunately no. As a matter of fact, router  $u$  needs only to keep for each interface  $i$  (i) the state of the downstream router  $w$  that has the smallest value of  $(d_{I \setminus \{i\}}^{\min}(w, *) - d(w, u))$  reachable on interface  $i$  and (ii) the state of the downstream router  $x$  that has the largest value of  $(d_{I \setminus \{i\}}^{\max}(x, *) - d(x, u))$  reachable on interface  $i$ . Specifically, let  $M_s^i(u)$  denote the set of routers that have downstream source routers reachable from interface  $i$  and are in the subtree  $T_s(u)$  rooted at router  $u$ , then each on-tree router  $u$  keeps the following per-interface state:

$$\min_{w \in M_s^i(u)} (d^{min}(w, *) - d(w, u))$$

and

$$\max_{w \in M_s^i(u)} (d^{max}(w, *) - d(w, u)).$$

(P2) can be answered and the eligibility tests can be formalized as follows. When a *join-request* from a receiver router  $v$  arrives at an on-tree router  $u$  on interface  $j$ , router  $u$  checks if

$$d(u, v) - \min_{w \in \cup_{i \in I \setminus \{j\}} M_s^i(u)} (d^{min}(w, *) - d(w, u)) \leq \delta \quad (6.5)$$

and

$$\max_{w \in \cup_{i \in I \setminus \{j\}} M_s^i(u)} (d^{max}(w, *) - d(w, u)) - d(u, v) \leq \delta. \quad (6.6)$$

Note that Eq. (6.5) (Eq. (6.6)) checks if the branch from router  $w$  to router  $v$  becomes the least-delay (largest-delay) on-tree branch. Also note that Eqs. (6.5)-(6.6) actually includes Eqs. (6.1)-(6.2) and the two eligibility tests (Eqs. (6.1)-(6.2) and Eqs. (6.3)-(6.4)) can be consolidated into one. If the *join-request* survives the eligibility test, it is forwarded to router  $u$ 's parent router  $w$ , with the updated accumulative delay information  $d(w, v) = d(w, u) + d(u, v)$ ; otherwise, it is immediately rejected.

The process of conducting the local test (Eqs. (6.5)–(6.6)) and forwarding the *join-request* upstream toward the core in the case that the local test succeeds is virtually the same as that described in Section 3.2, and repeats until the *join-request* is either rejected at some upstream on-tree router or forwarded to, and approved by, the core, whichever occurs first. In the former case, the first on-tree router  $u$  at which the *join-request* arrives sends back a *rejection-reply* message to the joining router  $v$ . In the latter case, the core sends back a *join-acknowledgment* message along the reverse on-tree path, and each on-tree router  $w$  on the path between router  $u$  and the core updates its state if  $d(w, v) > d^{max}(w, *)$  or  $d(w, v) < d^{min}(w, *)$ . Finally, router  $u$  sends back a *join-acknowledgment* message.

(P3) can be answered as follows. When a *join-request* sent by a source router  $v$  passes the eligibility test at an on-tree router  $u$  and is being forwarded upstream to router  $u$ 's parent router  $w$ , it includes the information  $d^{min}(u, *) - d(u, w)$  and  $d^{max}(u, *) - d(u, w)$  in addition to the other information. If the *join-request* is eventually approved at the core, each on-tree router  $w$  on the path between router  $u$  and the core updates its state on interface  $i$  if the value of  $d^{min}(u, *) - d(u, w)$  is smaller than  $\min_{x \in M_s^i(w)} (d^{min}(x, *) - d(x, u))$  or  $d^{max}(u, *) - d(u, w)$  is greater than  $\max_{x \in M_s^i(w)} (d^{max}(x, *) - d(x, u))$ .

### 6.1.2 Eligibility Tests in the Case of Source Member Join

Let  $D^{max}(v, *)$  and  $D^{min}(v, *)$  denote, respectively, the maximum and minimum outgoing delay between a source router  $v$  and *all* the other on-tree group routers. Note the difference between  $D^{max}(v, *)$  and  $d^{max}(v, *)$ : the former is the *global* delay information, while the latter is the *partial* delay information on the subtree  $T_s(v)$  rooted at router  $v$ . When a *join-request* sent by a source router  $v$  arrives at an on-tree router  $u$ , the inter-destination delay jitter bound imposes

$$D^{max}(v, *) - D^{min}(v, *) \leq \delta. \quad (6.7)$$

Eq. (6.7) can be rewritten as

$$(d(v, u) + D^{max}(u, *)) - (d(v, u) + D^{min}(u, *)) \leq \delta, \quad (6.8)$$

or

$$D^{max}(u, *) - D^{min}(u, *) \leq \delta. \quad (6.9)$$

Let  $\hat{d}^{max}(u, *)$  and  $\hat{d}^{min}(u, *)$  denote, respectively, the maximum and minimum outgoing delay between router  $u$  and all the upstream (with respect to the core) on-tree group members, then Eq. (6.9) can be rewritten as

$$\max[\hat{d}^{max}(u, *), d^{max}(u, *)] - \min[\hat{d}^{min}(u, *), d^{min}(u, *)] \leq \delta. \quad (6.10)$$

To keep state update and maintenance simple, each on-tree router  $u$  maintains only the state information of its subtree ( $d^{max}(u, *)$  and  $d^{min}(u, *)$ ), thus it can only conduct the eligibility test partially (Eq. (6.10)) and has to rely on the upstream routers for jointly conducting the eligibility test. Consequently, in the case that the local eligibility test succeeds, router  $u$  sends upstream to its parent router  $w$  a *join-request* with the following information:  $d(u, w)$ ,  $d^{max}(u, *)$ , and  $d^{min}(u, *)$ . When router  $w$  receives the *join-request* on interface  $\ell$ , it conducts the eligibility test on behalf of router  $u$ , with respect to all the group members in  $T_b(w)$ . Router  $w$  checks whether or not

$$\max[d(u, w) + d_{I \setminus \{\ell\}}^{max}(w, *), d^{max}(u, *)] - \min[d(u, w) + d_{I \setminus \{\ell\}}^{min}(w, *), d^{min}(u, *)] \leq \delta \quad (6.11)$$

holds. Note that the first parameter of the max/min function in Eq. (6.11) verifies whether or not the QoS is fulfilled from router  $u$  to all the group members that are in the subtree  $T_s(w)$  and are not reachable on interface  $\ell$ .

Similarly to the cases of imposing a bound on additive and multiplicative QoS parameters (Sections 3–4), the process repeats until the *join-request* is either rejected at some upstream on-tree router or forwarded to, and approved by, the core, whichever occurs first. In the former case, the first on-tree router  $u$  at which the *join-request* arrives sends back a *rejection-reply* message to the joining router  $v$ . In the latter case, each on-tree router  $w$  on the path between router  $u$  and the core updates its state if necessary (as described in Section 6.1.1) and router  $u$  sends back a *join-acknowledgment* message.

## 6.2 The State Kept at Each On-Tree Router

From the discussion in the above subsection, it is clear that each on-tree router  $u$  keeps the following state:

- (1)  $d_i^{max}(u, *)$ : the maximum delay among the on-tree paths from router  $u$  to all the downstream on-tree group members reachable on interface  $i$ .
- (2)  $d_i^{min}(u, *)$ : the minimum delay among the on-tree paths from router  $u$  to all the downstream on-tree group members reachable on interface  $i$ .
- (3)  $\min_{w \in M_s^i(u)}(d^{min}(w, *) - d(w, u))$ : the minimum value of  $d^{min}(w, *) - d(w, u)$  among the set of downstream routers that have downstream source routers, are in the subtree  $T_s(u)$  rooted at router  $u$ , and are reachable on interface  $i$ .
- (4)  $\max_{w \in M_s^i(u)}(d^{max}(w, *) - d(w, u))$ : the maximum value of  $d^{max}(w, *) - d(w, u)$  among the set of downstream routers that have downstream source routers, are in the subtree  $T_s(u)$  rooted at router  $u$ , and are reachable on interface  $i$ .

## 6.3 Member Leave Procedure

Again we consider two cases: the leaving router is a receiver and the leaving router is both a receiver and a source.

### 6.3.1 Leave of a Receiver Router

In the case that a receiver router  $v$  leaves a multicast tree, it sends a *quit-notification* message with the accumulative delay information  $d(u, v)$  to its parent router  $u$ . If router  $u$  does not have any other local members (on the directly attached subnet) or downstream on-tree routers, it sends a *quit-notification* message to its parent router  $w$  (with the accumulative delay information  $d(w, v) = d(w, u) + d(u, v)$ ) and deletes the corresponding interface from the forwarding cache. Otherwise, router  $u$  first deletes from the forwarding cache the interface  $i$  on which the *quit-notification* message arrives, and then checks if

$$d_{T \setminus \{i\}}^{min}(u, *) < d(u, v) < d_{T \setminus \{i\}}^{max}(u, *). \quad (6.12)$$

If Eq. (6.12) holds, it implies the state kept at the other on-tree routers will not change as a result of this member leave. Otherwise, router  $u$  first updates its state to reflect the fact that router  $v$  has left the multicast tree and sends upstream to its parent router  $w$  a *leave-update* message that carries  $d(w, v) = d(u, v) + d(w, u)$ . Upon receipt of a *leave-update* message at interface  $j$ , router  $w$  checks whether or not its state is affected by the member leave (i.e., Eq. (6.12) except  $u$  is replaced by  $w$  and interface  $j$  is excluded). The update process repeats until either the state of some upstream router is not affected by the member leave or the *leave-update* message reaches the core.

### 6.3.2 Leave of a Source Router

In the case that a source router  $v$  leaves a multicast tree, it sends a *quit-notification* message to its parent router  $u$ . If router  $u$  does not have any other local members on the directly attached subnet or downstream on-tree routers, it sends a *quit-notification* message to its parent router  $w$  (with the information  $d^{max}(u, *) - d(u, w)$  and  $d^{min}(u, *) - d(u, w)$ , in addition to  $d(w, v)$ ) and deletes the corresponding forwarding cache. Otherwise, router  $u$  first deletes from the forwarding cache the interface  $i$  on which the *quit-notification* message arrives, and then checks, in addition to Eq. (6.12), if

$$d^{min}(v, *) - d(v, u) > \min_{x \in M_s^i(u)}(d^{min}(x, *) - d(x, u)) \quad (6.13)$$

and

$$d^{max}(v, *) - d(v, u) < \max_{x \in M_s^i(u)}(d^{max}(x, *) - d(x, u)). \quad (6.14)$$

If Eqs. (6.12)–(6.14) hold, it implies the state kept at the other on-tree routers will not change as a result of this member leave. Otherwise, router  $u$  first updates its state to reflect the fact that router  $v$  has left the multicast tree and sends upstream to its parent router  $w$  a *leave-update* message that carries  $d^{min}(v, *) - d(v, w)$  and  $d^{max}(u, *) - d(u, w)$ , in addition to  $d(w, v)$ . Upon receipt of a *leave-update* message, router  $w$  checks whether or not its state is affected by the member leave (i.e., Eqs. (6.12)–(6.14) except  $u$  is replaced by  $w$ ). The update process repeats until either the state of some upstream router is not affected by the member leave or the *leave-update* message reaches the core.

## 7 State Update and Refresh Based on the Soft State Concept

We first discuss how the state is established at each on-tree router. Then, we present the state update and refresh mechanism that exploits the soft state approach.

### 7.1 State Establishment

When a requesting router  $v$  joins a multicast tree, each router  $w$  on the path from router  $v$  to router  $u$  establishes its state using the delay information carried in the *join-request* and *join-acknowledgment* messages. For example, the accumulative delay information contained in the *join-request* message received on interface  $\ell$  between router  $v$  and the current router  $w$  can be used by router  $w$  to establish its transient state ( $d_\ell^{max}(w, *) = d(w, v)$  and  $d_\ell^{min}(w, *) = d(v, w)$ ). When an off-tree router  $w$  with a pending *join-request* receives the corresponding *join-acknowledgment* from its upstream router, it updates its forwarding cache to reflect that it now becomes an on-tree router, and updates its transient state as an established one. Router  $w$  also forwards the message downstream to the requesting router.

The other on-tree routers update their state only if they receive a *join-request* message (i.e., they are asked to jointly approve the *join-request*).

## 7.2 State Refresh and Update

To be robust to message loss, many Internet protocols (e.g., DVMRP [11], PIM [14], CBT [1, 2], Simple Multicast[10]) have adopted the soft state approach for state maintenance, i.e., the state kept at each router is created and periodically refreshed by certain state-refresh messages. A state is deleted if no matching refresh messages arrive before the expiration of its associated timer. For example, CBT maintains its tree by having each downstream router periodically send a CBT *echo-request* message to its upstream neighbor router. The receipt of an *echo-request* message over a valid child interface prompts an *echo-reply* message that carries a list of groups for which the corresponding interface is a child interface. If no response is forthcoming within *upstream\_expire\_time* seconds, (e.g., a router’s upstream neighbor becomes unreachable), the router sends a *quit-notification* message upstream, and flushes all of its downstream branches by sending *flush-tree* messages, allowing them to individually rejoin if necessary.

We adopt the soft state approach and devise the state refresh and update procedure accordingly. The proposed state refresh procedure can be easily integrated with the CBT tree maintenance procedure (i.e., sending of *echo-request* and *echo-reply* messages) thus making the overheads that result from the proposed QoS extension small. Specifically, the state kept at an on-tree router is associated with the same *downstream\_expire\_time* timer. When a timer expires, the corresponding state is removed. An on-tree router initiates a state update process by periodically sending an *echo-request* message and piggybacking its state in the message. Upon receiving an *echo-request* message, a parent router updates its state if needed, resets the associated timer, responds with an *echo-reply* message, and initiates (upstream) a state update process if its state does change.

To deal with the situation in which message loss results in the removal of a valid soft state, we investigate whether or not the correctness of the proposed QoS extension is subject to loss of control messages. To deal with the loss of *echo-request/echo-reply* messages, CBT sets the *upstream\_expire\_time* and *downstream\_expire\_time* timer intervals  $K$  times larger than the *echo\_interval* timer interval so that *echo-request* messages can be lost  $(K - 1)$  consecutive times before the state is (incorrectly) removed. ( $K = 3$  in the current CBT protocol specification.) In the case that *echo-request* messages are lost more than  $K$  consecutive times, we consider the following two cases:

- (C1) If a downstream router does not hear from its upstream router upon the *upstream\_expire\_time* timer expiration, it considers the path to its upstream router as damaged, and sends a *quit-notification* message upstream, and *flush-tree* messages down-

stream to flush the subtree rooted at it. Each downstream group member will then individually find an alternative route to rejoin the tree. This is the approach currently defined in the CBT protocol specification.

- (C2) If an upstream router  $w$  does not hear from its downstream router  $u$  upon the *downstream\_expire\_time* timer expiration, it removes the state and the forwarding cache entry as if the subtree rooted at router  $u$  were “flushed”. Later when a *echo-request* message from the router  $u$  re-appears at router  $w$ , router  $w$ , instead of re-establishing the state and the forwarding cache (as the current CBT specification specifies), flushes the sub-tree. This is because during the period when router  $w$  and its upstream routers believe the sub-tree is “flushed,” if a new *join-request* arrives at one of these on-tree routers, it may be approved when it should actually be denied when the subtree is taken into account. Since consecutive loss of  $K$  *echo-request* messages is usually a good indication of low delivery quality of downstream links, it would actually be better for the downstream group members to re-join the multicast group individually along an alternative route of better quality.

Loss of *join-request*, *join-acknowledgment*, and *quit-notification* messages has been dealt with in the current CBT specification: if any of the *join-request* or *join-acknowledgment* messages is lost, the transient state established at each router on the path from the requesting router to an on-tree router is removed upon timer expiration. The state created on the path along which a *join-acknowledgment* message traverses will eventually time out, due to the lack of *echo-request* messages from downstream. Similarly, if any *quit-notification* message is lost, the valid state maintained at upstream routers will eventually time out, due to the lack of *echo-request* messages from downstream. The same strategy can be directly applied to the QoS extension mechanism to deal with loss of these messages (and *rejection-replay* messages as well). For example, if a *join-acknowledgment* message is lost on its way back to the on-tree router at which the *join-request* initially arrives, some (upstream) router(s) have changed their temporary state into valid soft state, while other (downstream) router(s) remove them. This does not affect the correctness of the proposed extension, because the state at upstream routers will be adjusted upon receipt of the next *echo-request* message. Similarly, when a *leave-update* message is lost on its way upstream, the soft state maintained at upstream routers will be adjusted upon receipt of the next *echo-request* message. The net effect is, however, that during the transient period new join requests may be pessimistically rejected because of tighter (outdated) state kept at some upstream routers. By the same token, *leave-update* messages may be totally eliminated to reduce message overhead at the expense of pessimistic admission control during the transient period.

## 8 What if the QoS Requirements are Heterogeneous

Since CBT is a receiver-initiated multicast routing protocol, it is especially well-suited to support heterogeneous receiver reservation, i.e., each receiver may request different levels of QoS [9]. Note, however, that it does not make sense for each receiver to request different bounds on the inter-destination delay jitter. The eligibility tests and the state update procedure can be modified to support receiver heterogeneity as follows.

### 8.1 Receiver Heterogeneity in the Case of Additive QoS Parameters

Let  $D_v$  denote the delay bound imposed by a receiver router  $v$ , then the eligibility test (Eqs. (3.1)–(3.2)) is modified as

$$d(u, v) + d_{I \setminus \{i\}}^{max}(*, u) \leq D_v, \quad (8.1)$$

and

$$d(v, u) \leq D_w - d(u, w), \text{ for all group members } w \text{ in } T_s(u). \quad (8.2)$$

Note that Eq. (8.2) can be rewritten as

$$d(v, u) \leq \min_{w \in T_s(u)} D_w - d(u, w). \quad (8.3)$$

The per-interface state kept at each on-tree router  $u$  is (i)  $d_i^{max}(*, u)$  and (ii) the minimum laxity defined as the minimum difference between  $D_w$  and  $d(u, w)$  among all downstream on-tree group members  $w$  reachable on interface  $i$ , i.e.,  $\min_{w \in T_s(u)} (D_w - d(u, w))$ .

### 8.2 Receiver Heterogeneity in the Case of Multiplicative QoS Parameters

Let  $1 - L_v$  denote the upper bound on the packet loss ratio imposed by a receiver router  $v$ , then the eligibility test (Eqs. (4.1)–(4.2)) is modified as

$$s^{min}(*, v) = s(u, v) + s_{I \setminus \{i\}}^{min}(*, u) \geq \log L_v, \quad (8.4)$$

and

$$s(v, u) \geq \log L_w - s(u, w), \text{ for all members } w \text{ in } T_s(u). \quad (8.5)$$

The state kept at each on-tree router  $u$  is (i)  $s_i^{min}(*, u)$ ; and (ii) the maximum difference between  $\log L_w$  and  $s(u, w)$  among all the downstream on-tree group members  $w$  reachable on interface  $i$ , i.e.,  $\max_{w \in T_s(u)} (\log L_w - s(u, w))$ .

## 9 Experiment Results

We have developed a customized Java-integrated network simulation tool, *NetSim<sup>Q</sup>*, to evaluate the QoS

control capability for systems equipped with different admission control, resource reservation, QoS unicast/multicast routing, and message scheduling mechanisms [17]. *NetSim<sup>Q</sup>* allows users to input (1) the network topology, the link capacity and the node buffer size; (2) the traffic characteristics and the QoS requirements of message streams; (3) the underlying message model and the resource reservation protocol used; (4) the message scheduling algorithms used; and (5) the routing protocols used. Different combinations of network configuration can be easily composed and evaluated in a plug-and-play fashion. With all the inputs, the underlying simulation engine then emulates, for each connection (specified by the user or randomly generated by the simulation engine), the connection establishment phase, the data transfer phase, and the connection termination phase, and keeps track of the network states and simulation statistics. We take advantage of the simulation environment provided in *NetSim<sup>Q</sup>*, and evaluate the proposed QoS extension in terms of (i) the percentage of join requests being rejected and (ii) message overhead.

We construct three network configurations in *NetSim<sup>Q</sup>* in which all the network components are kept the same except the routing module. The first two configurations use the CBT protocol and the CBT protocol with the proposed QoS extension, respectively, as the routing module, while the third configuration includes the CBT protocol with a straightforward “brute-force” admission control method in which *join-request* messages are simply forwarded on all the interfaces (i.e., *all* the on-tree routers jointly approve/disapprove *join-requests*). The latter configuration is used to quantify the message overhead introduced by the proposed extension.

**Network topology:** The network topology is randomly generated using the method proposed in [18]: the  $n$  routers of a graph are randomly placed on a Cartesian coordinate grid with unit spacing. The  $(x, y)$  coordinates of each router was selected uniformly from integers in  $[0, m]$ , where  $m$  must be selected such that  $m^2 > n$ . In our simulation, we choose  $m$  equal to  $\sqrt{2n}$ . For each pair of routers, an edge is placed connecting the two routers with probability

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

where  $d(u, v)$  is the distance between router  $u$  and  $v$ ,  $L$  is the maximum possible distance between two routers and  $\theta$  is a random number from  $(0, 1)$ . The parameters  $\alpha$  and  $\beta$  are in the range  $(0, 1]$  and can be used to obtain certain desirable characteristics in the topology. For example, a large  $\beta$  gives nodes with a high average degree, and a large  $\alpha$  favors long connections. With appropriate parameters selected, this method has been shown in [18] to render networks that resemble “real world” networks. We fine tune  $\alpha$  and  $\beta$  to generate networks with desired node degrees. Routers are

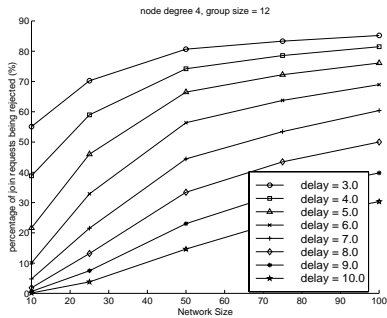


Figure 2: The percentage that a *join-request* is rejected under the proposed extension for small-size networks with small-size multicast groups.

consecutively labeled starting from 0 and group members are randomly picked up. The router with a label equal to the average of the labels of all the group members is (randomly) selected as the core for the multicast group. The delay at each link is normally distributed with mean and standard deviation varying from 1 to 3 units of time, respectively.<sup>5</sup>

**Experiment design:** We consider the case in which the QoS requested is the end-to-end delay constraint. Each simulation run is designed to vary one or more of the following parameters: the network size, the average node degree, the size of multicast groups, and the end-to-end delay constraint. We conduct two sets of experiments. In the first set of simulation runs, we evaluate the proposed QoS extension under small-size networks. The network size varies from  $n = 10$  to 100 routers, the average node degree varies from 3 to 5, the size of each multicast group varies from 3 to 5, the size of each multicast group varies from 4 to 16 members (with each member being both a source and a receiver), and the end-to-end delay requirement varies from 3.0 to 10.0 units of time. For each given set of parameters, 20 different network topologies are (randomly) generated and 10 multicast groups are generated for each network topology. The result is obtained by averaging statistics gathered from the 200 multicast groups generated in each simulation run.

In the second set of simulation runs, we evaluate the proposed extension under large-size networks with large-size multicast groups. The network size is set to  $n = 500, 1000, 1500$  and 2000 routers, respectively, the average node degree varies from 2.5 to 3.5, the size of each multicast group is set to 50, 75 and 100 members, respectively, and the end-to-end delay bound varies from 12.0 to 18.0 units of time. For each given set of parameters, 20 different network topologies are generated and a multicast group is generated for each network topology. The result is obtained by averaging statistics gathered from the 20 multicast groups generated. Note that although only a single multicast group is generated in each simulation run, we vary the mean and variance of the link delay distribution to emulate

<sup>5</sup>The absolute value of the outcome of the random variable with normal distribution is used as the link delay.

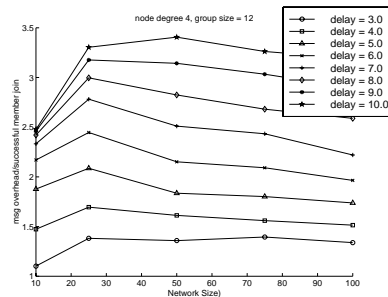


Figure 3: The number of messages generated, in addition to those defined in the CBT protocol, per successful member join under the proposed extension for small-size networks with small-size multicast groups.

the scenario in which background traffic interferes with the multicast group generated.

The performance data collected in each simulation run are (i) the percentage that a *join-request* is rejected under the proposed QoS extension; and (ii) the number of additional messages sent per *join-request* approved under the proposed extension, in addition to those defined in the CBT protocol. Item (i) gives an index on how often the proposed mechanism rejects ineligible new members so as to enforce QoS. A rejected member may send a *join-request* with a lower level of QoS, attempts on an alternative route to join the multicast (which requires modification of how a *join-request* locates a path to an on-tree router in CBT and is a subject of future research), or retry later (with the hope that some on-tree group members that caused the QoS violation have left). Item (ii) measures the message overhead that results from deploying the proposed extension. For benchmarking purpose, we also give the percentage of message overhead increased under the proposed extension as compared to the original CBT protocol and the percentage of message overhead reduced under the proposed mechanism as compared to the brute-force method.

In spite of numerous system parameters involved, the results are found to be quite robust in the sense that the conclusion drawn from the performance curves for a representative set of parameter values (reported below) is valid over a wide range of parameter values. In particular, the conclusions drawn under the case in which the QoS requirement is the end-to-end delay bound are valid under the other cases.

Fig. 2–5 give the results of the first set of simulation runs. As shown in Fig. 2, the percentage of *join-requests* being rejected increases as the end-to-end delay requirement becomes tighter or as the network size increases. For most of the combinations of network sizes and QoS requirements, a non-negligible percentage of *join-requests* do not pass the eligibility tests. This may imply that the shortest-path-to-core approach which a *join-request* uses to locate a route to join the multicast tree in CBT may not be desirable when QoS is figured in. An off-tree route search approach similar to that of

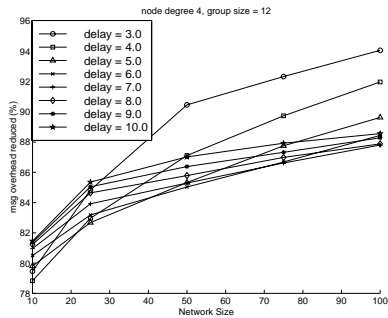


Figure 4: The percentage of message overhead reduced under the proposed extension as compared to the brute-force method for small-size networks with small-size multicast groups.

YAM [19] or QoSMIC [20] may be deployed to find a better QoS-satisfying route.

Fig. 3 and 5 show that the message overhead introduced under the proposed extension does not increase with network size, and ranges from 2% to 90% more than that of the original CBT protocols. The percentage of message overhead introduced by the proposed extension falls below 50% when the network size exceeds 100. This serves as an evidence that the proposed extension is scalable. Moreover, as shown in Fig. 4, the proposed mechanism incurs 55%–85% less of control messages as compared to the “brute-force” method. Another interesting finding is that the message overhead increases with the end-to-end delay requirement (Fig. 3). This is because the overhead to approve a *join-request* is generally much more than that to reject one.

Fig. 6–9 gives the results of the second set of simulation runs. Similar to the first set of experiments, a significant percentage (averagely 55%) of *join-requests* cannot be approved if they simply follow the shortest path to the core to join the multicast tree. Again as shown in Fig. 7 and 9, the message overhead does not increase with the network size in large-scale networks, suggesting that the proposed extension is scalable. Moreover, the proposed extension incurs 80-97% less of messages as compared to the “brute-force” method, and introduces 25-90% more messages than the original CBT protocol.

## 10 Conclusion

This paper presents a set of QoS enhancements to allow QoS deployment in the Core Based Tree (CBT) Protocol [1, 2], with the minimum possible impact to the existing infrastructure. Different levels of QoS in the form of additive QoS (e.g., end-to-end delay bound), multiplicative QoS (e.g., maximum packet loss ratio), and/or concave (e.g., minimum bandwidth available) can be supported.

Major components of the QoS extension include (i) a set of simple eligibility tests to verify whether or not a new member can join a multicast tree at adequate QoS, while not violating the existing QoS to the other on-

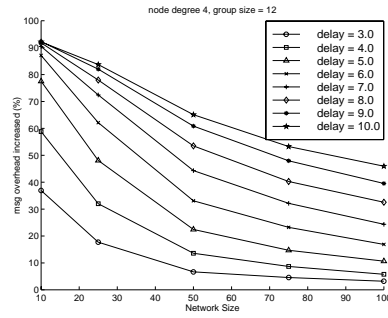


Figure 5: The percentage of message overhead increased under the proposed extension as compared to the CBT protocol for small-size networks with small-size multicast groups.

tree members; and (ii) a state update/refresh procedure that is based on soft state and can be readily integrated with the tree maintenance mechanism that already exists in CBT, e.g., sending of *echo-requests* and *echo-replies* in CBT. Experiment results show that the proposed extension is scalable because it only introduces 2-90% (25-90%) more messages than the CBT protocol in small-size (large-size) networks. The proposed work can be included as an (optional) QoS extension to CBT and any other bi-directional multicast routing protocol with an explicit member join procedure and a soft state refresh procedure, such as Simple Multicast [10].

## References

- [1] A. Ballardie, B. Cain and Z. Zhang, “Core based trees (CBT version 3) multicast routing - protocol specification,” draft-ietf-idmr-cbt-spec-v3-00.txt, Internet-draft, Inter-domain multicast routing, March 1998.
- [2] A. Ballardie, “Core based trees (CBT) multicast routing architecture,” RFC 2201, ftp://ds.internic.net/rfc/rfc2201.txt.
- [3] Christophe Diot, Walid Dabbous, and Jon Crowcroft, “Multipoint communication: A survey of protocols, functions, and mechanisms” *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, April 1997.
- [4] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 7, pp. 1228–1234, September 1996.
- [5] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, “Real-time communication in packet-switched networks,” *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 122–139, January 1994.
- [6] H. Zhang, “Service Disciplines for guaranteed performance service in packet-switching networks,” *Proceedings of the IEEE*, Vol. 83, No. 10, October 1995.
- [7] G. N. Rouskas and I. Baldine, “Multicast routing with end-to-end delay and delay variation constraints,” *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 1, pp. 1–9, April 1997.
- [8] W. Fenner, “Internet Group Management Protocol: version 2 (IGMPv2),” draft-ietf-idmr-igmp-v2-08.txt, Internet-draft, Inter-domain multicast routing, 1998.
- [9] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource ReSerVation protocol RSVP - version

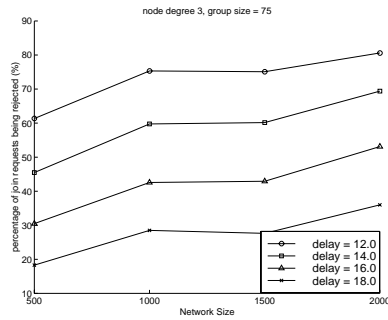


Figure 6: The percentage of message overhead reduced under the proposed extension as compared to the brute-force method for large-scale networks with large-size multicast groups.

1 function specification,” draft-ietf-rsvp-spec-15.ps, Internet draft, May 1997.

- [10] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, and T. Maufer, “Simple multicast: a design for simple, low-overhead multicast,” <http://www.ietf.org/internet-drafts/draft-perlman-simple-multicast-01.txt>, December 1998.
- [11] T. Pusateri, “Distance vector multicast routing protocol,” <ftp://ds.internic.net/internet-drafts/draft-ietf-idmr-dvmrp-v3-04.txt>, Internet draft, February 1997.
- [12] J. Moy, “Multicast extensions to OSPF,” RFC 1584, March 1994.
- [13] Debra Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy, “Protocol independent multicast (PIM) sparse mode/dense mode,” <ftp://netweb.usc.edu/pim>. Working drafts, 1996.
- [14] D. Estrin, D. Farinacci, S. Deering, D. Thaler, A. Helmy, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, “Protocol independent multicast - sparse mode (PIM-SM): protocol specification,” <http://netweb.usc.edu/pim>, RFC 2362 and Internet draft, 1998.
- [15] H.-Y. Tyan, J. Hou, B. Wang, and Y.-M. Chen, “QoS extension to CBT,” Technical report, Dept. of Electrical Engineering, The Ohio State University, Columbus, OH. Available at <http://eewww.eng.ohio-state.edu/drcl/pubs.html>.
- [16] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, “QoS Routing Mechanisms and OSPF Extensions”, draft-guerin-qos-routing-ospf-04.txt.
- [17] Hung-Ying Tyan, Bin Wang, Yi Ye, and Chao-Ju Hou. NetSim<sup>Q</sup>: A Java-integrated network simulation tool for QoS control in point-to-point high speed networks. *3rd NASA Research and Education Network Workshop*, Moffett Field, CA, August 1998.
- [18] B. Waxman. SWSL: routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6:1617–1622, December 1988.
- [19] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pp. 5–11, January 1997.
- [20] A. Banerjee and M. Faloutsos and R. Pankaj. Designing QoS-MIC: A Quality of Service Sensitive Multicast Internet Protocol. *Internet Draft*, 1998.

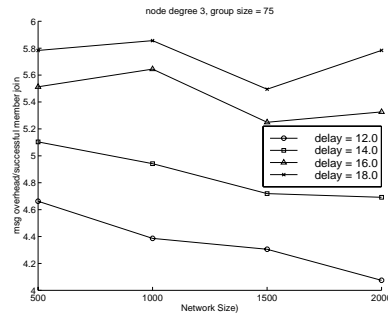


Figure 7: The percentage of message overhead reduced under the proposed extension as compared to the brute-force method for large-scale networks with large-size multicast groups.

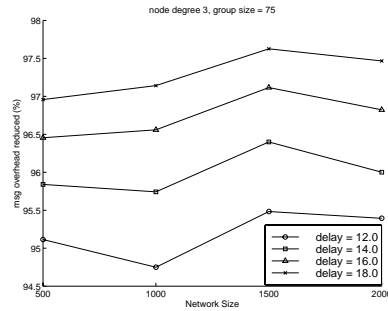


Figure 8: The percentage of message overhead reduced under the proposed extension as compared to the brute-force method for large-scale networks with large-size multicast groups.

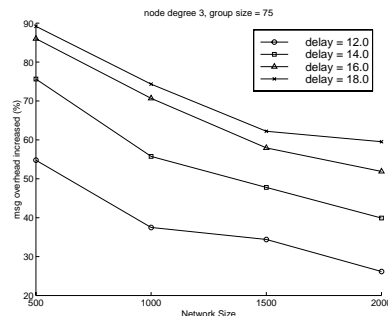


Figure 9: The percentage of message overhead reduced under the proposed extension as compared to the brute-force method for large-scale networks with large-size multicast groups.