AQoSM: Scalable QoS Multicast Provisioning in Diff-Serv Networks ⋆

Jun-Hong Cui [a] , Li Lao [b] , Michalis Faloutsos [c] , Mario Gerla [b]

[a] *Computer Science & Engineering Department, University of Connecticut, Storrs*

[b] *Computer Science Department, University of California, Los Angeles*

[c] *Computer Science & Engineering, University of California, Riverside*

Abstract

The deployment of IP multicast support is impeded by several factors among which are the state scalability problem, the cumbersome management and routing, and the difficulty of supporting QoS. In this paper, we propose an architecture, called Aggregated QoS Multicast (AQoSM), to provide scalable and efficient QoS multicast in Diff-Serv networks. The key idea of AQoSM is to separate the concept of groups from the concept of distribution tree by "mapping" many groups to one distribution tree. In this way, multicast groups can now be routed and rerouted very quickly by assigning different labels (e.g., tree IDs) to the packets. Therefore, we can have load-balancing and dynamic rerouting to meet QoS requirements. In addition, the aggregation of groups on fewer trees leads to routing state reduction and less tree management overhead. Thus, AQoSM enables multicast to be seamlessly integrated into Diff-Serv without violating the design principle of Diff-Serv of keeping network core "QoS stateless" and without sacrificing the efficiency of multicast. Finally, efficient resource utilization and strong QoS support can be achieved through statistical multiplexing at the level of aggregated trees. We design a detailed MPLS-based AQoSM protocol with efficient admission control and MPLS multicast tree management. By simulation studies, we show that our protocol achieves significant multicast state reduction (up to 82%) and tree maintenance overhead reduction (up to 86%) with modest (12%) bandwidth overhead. It also reduces the blocking ratio of user requests with strong QoS requirements due to its load balancing and statistical multiplexing capabilities.

Key words: multicast, QoS, state scalability, Diff-Serv, MPLS

⋆ This material is based upon work supported by the National Science Foundation under Grant No. CNS-0435515, CNS-0435230, ANIR-9985195 (CAREER award), and IDM-0208950.

Email addresses: jcui@cse.uconn.edu (Jun-Hong Cui), llao@cs.ucla.edu (Li

1 Introduction

Real-time multicast applications need mechanisms to support (if not guarantee) QoS. This is especially compelling in the case of commercial applications: customers will not pay for a service unless it is reliable and offered at a satisfactory level of quality. From the perspective of network service providers, QoS multicast provisioning requires three mechanisms: a) discovery of available resources, b) maintenance of the available resources, and c) quick recovery from failures. These requirements translate to the following functions: a) QoS-aware routing, b) call admission and load-balancing, and c) fault-tolerant routing.

The current multicast architecture does not handle QoS efficiently due to the following issues. First, multicast routing state does not scale well: routers need to keep state per group and in some protocols per group/source. Large numbers of groups result in large amount of state to be maintained at routers, which translates into large memory requirements and slow packet forwarding. Second, routing is cumbersome. A tree is associated with a single group. Creating and maintaining a multicast tree per group is time and resource consuming, especially if we account for QoS constraints. As a result, a congested link or a node failure leads to tearing down and rebuilding large parts of the tree. Finally, load-balancing and rerouting a tree is not an option.

Though most research papers on QoS multicast focus on solving a theoretical QoS-constrained multicast routing problem, there have been several more pragmatic efforts to bring QoS into the existing IP multicast architecture, such as YAM [9], QoSMIC [18], QMRP [10], RIMQoS [22], QoS extension to CBT [25], and PIM-SM QoS extension [5]. However, we find that the problem can only be addressed by an architecture that will address all three QoS aspects listed earlier, namely QoS aware routing, admission control, and fault tolerance. The proposed protocols address only one of the requirements, usually the QoS aware routing. Furthermore, they all use per-flow state and continue to be plagued from the state scalability problem. Today people are backing away from micro-flow based QoS architecture, for example, the Integrated Services architecture (IntServ) [8], and are moving towards the aggregated-flow based solutions, such as Differentiated Services (Diff-Serv) [6] architecture and the Multiprotocol Label Switching (MPLS) technology [39]. The argument behind this choice is simple: the per-flow reservation and data packet handling required by IntServ lead to scalability problems at network core routers. Incorporating the per-flow state requirement and traffic management of multicast in a per-class architecture, such as a Diff-Serv or MPLS network, does not solve the state scalability problem, since each router still needs to maintain sepa-

Lao), michalis@cs.ucr.edu (Michalis Faloutsos), gerla@cs.ucla.edu (Mario Gerla).

rate states for individual multicast groups that pass through it. To improve multicast state scalability, several mechanisms have been proposed such as [50], [44], [12], [49], [38] [7], etc. Though promising, these efforts are typically not concerned with QoS. Recently, there are also some works which target at scalable QoS multicast provisioning [28] [48] [45]. However, they either do not scale to large groups or sacrifice the efficiency of multicast.

In this paper, we propose an architecture, called Aggregated QoS Multicast (AQoSM), to provide scalable QoS multicast that addresses the issue of QoS and routing in a unified and comprehensive way. Our architecture intends to be employed in a Diff-Serv-supported transit domain and its use is transparent outside the domain or to the application layer. AQoSM uses the concept of aggregated multicast [21], in which the key innovation is the decoupling of group and distribution tree concepts. Many groups can be multiplexed on a single tree. More importantly, a group can be switched easily between distribution trees. This simple feature leads to a proliferation of new properties and advantages. First, the creation and management of trees become more efficient. We can create trees on-demand and route a group very quickly. Second, group *rerouting* becomes a viable option: it is a matter of assigning different labels (i.e., tree IDs) to its packets at the entrance points. This opens new possibilities for load-balancing and fault tolerance: we can now start to look at sophisticated load-balancing and failure recovery schemes. This way, we can adapt to changes in the QoS requirements, in the network load, and in the group membership. From the scalability point of view, the major benefit is that our architecture reduces the multicast state by mapping multiple groups to one tree. Finally, the admission control can be carried out on the level of aggregated trees instead of individual links, and thus is resource efficient due to statistical multiplexing of multiple groups on a single tree.

As we have discussed, QoS multicast provisioning is a multifaceted problem, involving routing, admission control, resource management and many other issues. Our goal is to provide efficient and practical solutions for those issues. Based on our proposed AQoSM architecture, we develop a protocol using MPLS technique. Our analysis and simulation study will show that the developed AQoSM protocol is efficient, scalable, feasible, and implementable. With the increasing demand for interactive, real-time applications, AQoSM is a promising solution for scalable, real-time QoS multicast services in the Internet.

The rest of this paper is organized as follows. Section 2 gives some background information on aggregated multicast, Diff-Serv and MPLS. Section 3 describes the architecture of AQoSM and discusses some related issues. Section 4 presents an MPLS-based AQoSM protocol (MAQoSMP) design in details. Then we study the performance of MAQoSMP through simulations in Section 5. Finally we review some related work in Section 6 and conclude with

a brief summary of our contributions in the end.

2 Background

2.1 Aggregated Multicast

Conventional IP multicast routing protocols confront a severe scalability problem when there are large numbers of multicast groups ongoing in the networks. This is mainly due to **state explosion** and **control explosion** issues. First, each router needs to maintain separate states for individual groups (or group/sources). Large numbers of groups mean large amount of state to be maintained at routers, which translates into large memory requirement and slow packet forwarding. Second, conventional IP multicast protocols establish and maintain a multicast tree per-group (or group/source). Large numbers of groups mean large numbers of trees to set up and maintain. Consequently, the number of corresponding tree setup and maintenance control messages will become huge and explode. In backbone networks, this "state scalability" problem will be exacerbated, since there are potentially enormous multicast groups crossing backbone domains. A backbone domain is typically a concentration point of the global network, and its performance greatly influences the global network's performance.

Aggregated multicast [21] is a scheme proposed to improve multicast state scalability within a transit domain (especially the backbone domain) by exploiting inter-group tree sharing. It is inspired by a key observation: within a backbone domain, the number of edge routers are limited; therefore, when there are a large number of simultaneously active multicast groups, many groups are likely to have significant overlaps in their multicast delivery trees. As a result, these multicast flows can be aggregated into one flow at ingress router(s) by packet encapsulation techniques (e.g., IP encapsulation or MPLS [1]), and delivered to egress routers via a single multicast tree, which we call *aggregated tree*. In this way, core routers, i.e., the routers in the middle of the network, only need to keep forwarding state per aggregated tree rather than per group. Of course, edge routers at the boundaries of the network need additional information for classifying the multicast packets. It also solves the control explosion issue, since the multicast tree setup and maintenance control messages are significantly reduced. In essence, this scheme is similar to Diff-Serv in that both of them eliminate "per-flow" state in the core routers by coupling multiple flows into one equivalent "class".

[1] MPLS is more efficient than IP encapsulation, since it only inserts a "thin" label between link and IP layer in packets.

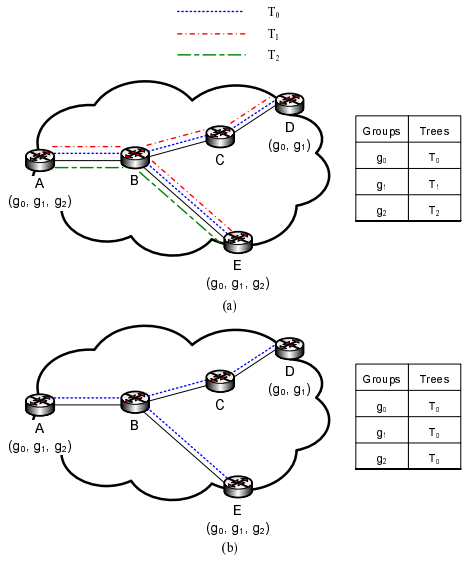


Fig. 1. Comparison of traditional multicast and aggregated multicast. (a) In traditional multicast, one multicast tree is needed for each group. (b) In aggregated multicast, one aggregated tree can be shared by multiple groups.

In aggregated multicast, the critical step is group-tree matching, that is, to match groups to appropriate trees. As shown in Fig. 1, multicast groups g_0 and g_1 have members A, D, and E, and g_2 has members A and E. These three groups can be aggregated onto the same multicast tree T_0, which spans the leaf routers A, D, and E. For g_0 and g_1, every tree leaf of T_0 corresponds to a group member, so T_0 is a **perfect** match for g_0 and g_1. In contrast, group g_2 does not have member at the tree leaf D, thus T_0 is a **leaky** match for g_2. Leaky match allows us to further reduce multicast state at the cost of bandwidth waste, since some data are delivered to nodes that are not members of the group.

2.2 Diff-Serv and MPLS

Differentiated Service architecture (Diff-Serv) [6] is proposed for scalable service differentiation in the Internet. In a Diff-Serv domain, packets crossing a link and requiring the same behavior (e.g., scheduling treatment and drop probability) constitute a Behavior Aggregate (BA). At the ingress nodes, the packets are classified and marked with a Diff-Serv Code Point (DSCP) according to their Behavior Aggregate. At each transit node, the DSCP is used to determine the behavior for each packet.

Multiprotocol Label Switching (MPLS) [39] emerges as an important traffic engineering technology for the Internet. It uses label switching technique. In an MPLS domain, when a stream of data traverses a common path, a Label Switched Path (LSP) can be established using MPLS signaling protocols. At the ingress Label Switch Router (LSR), each packet is assigned a label and

is transmitted downstream. At each LSR along the LSP, the label is used to forward the packet to the next hop.

Recently, there are several research proposals [20,23] targeted at MPLS support of Differentiated Services, but only for unicast. [20] specifies a solution for supporting the Diff-Serv Behavior Aggregates over an MPLS network, which relies on a combined use of two types of LSPs: E-LSP and L-LSP. The solution allows the MPLS network administrator to select how Diff-Serv Behavior Aggregates (BAs) are mapped onto LSPs so that the Diff-Serv, Traffic Engineering and protection objectives can be best matched within the network. [23] mainly addresses how to provide Diff-Serv Traffic Engineering solution using E-LSP, outlining methods to signal bandwidth requirements for multiple Ordered Aggregates when setting up E-LSPs. [19] identifies the DiffServ-aware MPLS Traffic Engineering (DS-TE) problem, that is, MPLS traffic engineering should be able to optimize resource usage at per Diff-Serv class level, and presents the requirements for the technical solutions. [17,30] propose some solutions to support DS-TE. Trimintzios et al. advocate an architectural framework with integrated management and control planes for supporting end-to-end QoS in Diff-Serv/MPLS networks [51]. [11] proposes a new policy to avoid repeated resource preemption due to flow priority and load balancing in MPLS and applies it in Constraint-Based Routing scheme to improve end-to-end QoS in Diff-Serv-aware MPLS networks. Rouhana and Horlait presented an architecture called DRUM (Diffserv and RSVP/IntServ Use of MPLS), in which an MPLS/DiffServ backbone domain delivers end-to-end service guarantees to boundary customer domains supporting IntServ or Diff-Serv models [40].

In this paper, we will develop an MPLS-based AQoSM protocol, which focuses on dealing with multicast traffic. We will not discuss the interaction of unicast traffic with multicast traffic and a unicast QoS architecture is assumed to be already in place.

3 AQoSM—The New Architecture for Scalable QoS Multicast Provisioning

We design a new architecture, AQoSM (Aggregated QoS Multicast), to support scalable QoS multicast. Our architecture uses the aggregated multicast concept. Aggregated multicast was designed as state-reduction scheme, but here, it becomes a powerful tool to simplify traffic management and QoS provisioning. AQoSM provides all the necessary functionality that is needed for QoS support: a) QoS routing, b) call admission control, and c) reconfiguration. It is targeted at QoS multicast provisioning in a single Diff-Serv domain,

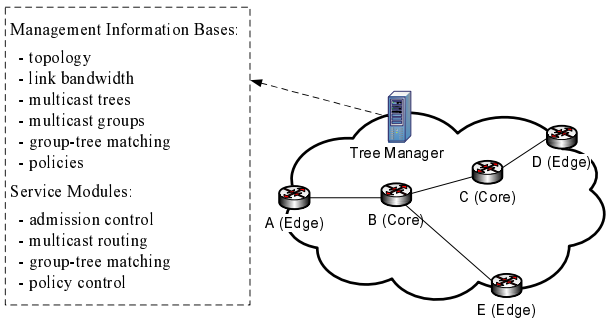


Fig. 2. Illustration of a tree manager in a Diff-Serv domain.

particularly a backbone domain [2]. In this domain, data packets from multiple groups are multiplexed at incoming edge routers, and de-multiplexed at outgoing edge routers.

AQoSM maintains multicast trees with each serving multiple groups. A group is assigned to a tree after careful consideration of: a) the destinations of the group compared to the tree leaves, b) the QoS requirements of the group, and c) available bandwidth on the tree. The advantage is that a group can switch between trees fast. This way, we can reduce the set-up cost for each group, and have groups switch trees when necessary, i.e., for QoS reasons. To clarify, in the context of AQoSM, a (multicast) tree computed and established to transmit multicast packets can also be referred to as an aggregated (multicast) tree, because it is usually shared by several multicast groups. These terms are interchangeable.

Before explaining design issues in more detail, we give a "big picture" of AQoSM first. To manage groups and trees, AQoSM architecture incorporates a logical entity called *tree manager*, which is illustrated in Fig. 2. Its responsibilities include tree maintenance and group-to-tree matching. It consists of several service modules, such as admission control, group-tree matching, routing and policy control, which will be explained in next two paragraphs. The tree manager can be implemented in centralized or distributed ways. For simplicity of presentation, we can think of it as a single node for the time being.

To provide QoS support, tree manager needs to collect up-to-date information of: the network topology, the available resources, the group membership, and their QoS requirements. When it discovers that there is a request for a new multicast group (identified by the edge routers initially involved in it), it calls the group-tree matching module. The group-tree matching module keeps the information of active groups, established trees and the group-tree matching table, and matches incoming multicast groups to proper trees. If no such tree exists, the tree manager computes a new multicast tree according to member-

[2] Usually, there are large numbers of groups in backbone domains, thus the scalability issue is even more challenging.

ship and QoS requirements using the routing module.

After a new tree is computed, the admission control module needs to decide whether adequate resource is available. If not, the incoming multicast request is rejected. Otherwise, the corresponding tree is established in the network [3]. Once a proper multicast tree is found or established, the tree manager distributes the corresponding group-tree matching entry to the member edge routers (source routers and receiver routers) within the group. Source routers take charge of encapsulating, classifying, and marking individual group packets, while receiver routers decapsulate group packets. A member router might act as both source router and receiver router. During the whole process, the policy control, which preserves a policy information base, may be consulted to do a network policy administration. A big picture of AQoSM is shown in Fig. 3, where A, D, and E are edge routers (with A as source router and D and E as receiver routers), and B and C are core routers.

From this brief overview, we can see AQoSM involves many design issues. The remainder of this section describes solutions for each critical issue involved in AQoSM.

3.1 Link State Collection

The tree manager needs to obtain "link state" information from all the routers in the domain in order to find or compute a proper tree for each multicast group. AQoSM is open to many options. If the network is small and very stable, the network administrators could even configure the tree manager manually. Of course, this is not typical. Generally, networks are dynamic with frequent changes to both their topology and traffic load. In the dynamic case, there are two options depending on the unicast routing approach employed in the domain. If a distance vector approach is used in unicast routing, then each router in the domain sends its link state packets directly to the tree manager. Routers can send updated link-state packets when there are some changes, for example, some links or routers go down, or some routers come up. On the other hand, if a link state approach (e.g. OSPF) is employed for unicast routing, the tree manager will benefit from the flooding of link-state packets from all the routers, and thus discover the network topology easily.

[3] How to establish a tree depends on what encapsulation technique is used. If IP encapsulation is employed, then a traditional IP multicast routing protocol can be adopted, such as PIM-SM or CBT. If MPLS service is available, an appropriate multicast Label Distributed Protocol (LDP) needs to come into place. We will show an MPLS-based AQoSM protocol in Section 4.

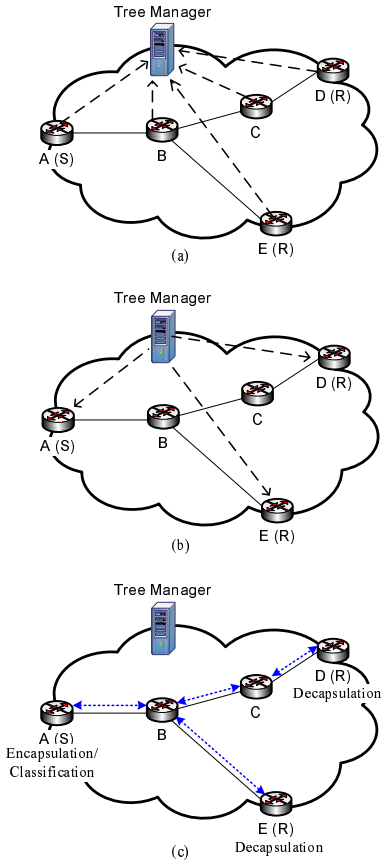


Fig. 3. A big picture of Aggregated QoS Multicast: (a) Membership, QoS requirement, link state, and available bandwidth collection; (b) Group-tree matching entry distribution; (c) Multicast group packets transmitting on established aggregated multicast tree.

3.2 Group Membership Collection

To find a proper tree or establish a new tree for a multicast group, the tree manager needs to know group membership in advance. Similar to link state collection, there are two options for collecting group membership, depending on the unicast routing approach in the domain.

The simplest way is for each edge router to send its group membership directly to the tree manager, indicating what groups it wants to join. Whenever a router's group membership is changed, it sends the updated version to the tree manager. If unicast routing uses a link state approach, then membership can be piggybacked on link-state packets; that is, edge routers add records to link-state packets to indicate which groups they want to listen to. Therefore, after the tree manager receives the link-state packets from all the routers, it can compute a multicast tree for a multicast group. The link state advertising option is very similar to MOSPF [31] except that here only the tree manager computes multicast trees.

3.3 Admission Control

By performing admission control plus classified services from Diff-Serv, AQoSM is able to provide some QoS guarantees to real-time applications. These QoS guarantees can be either deterministic or statistical. Deterministic services provide deterministic support at the cost of low resource utilization, since they reserve resources based on worst-case scenarios. Statistical services greatly improve network resource utilization by statistically multiplexing network traffics, but they only provide probabilistic performance guarantee.

As we know, there are two basic approaches of admission control: parameter-based and measurement-based. The parameter-based approach tries to precisely characterize traffic flow with a set of parameters, such as peak rate, average rate, and token bucket size. The admission control agent then calculates the required resources based on these parameters. It can achieve either deterministic or statistical performance guarantee depending on the admission control policy. On the other hand, in the measurement-based approach (which was first proposed by [26]), the network measures the actual traffic load and uses that for admission control. Since it is probabilistic in nature, it cannot provide tightly guaranteed resource.

Generally, for QoS multicast, admission control can be performed at two different levels: link level and flow (or tree for multicast) level. In the case of link-level admission control, the effective bandwidth for each link is calculated (or measured), and then the residual link bandwidth can be obtained. This method can statistically multiplex all traffics on the same link, but it incurs a large amount of overhead since the effective bandwidth of individual links needs to be updated frequently. For admission control at non-aggregated tree level, the bandwidth requirement is maintained for each delivery tree. The processing overhead is less than the previous method, but still very cumbersome when there are a large number of multicast groups in the network. In AQoSM, we can combine the advantages of these two methods by employing statistical multiplexing for each aggregated tree, which delivers data for multiple groups.

AQoSM is flexible in the sense that it can be seamlessly integrated with either measurement-based or parameter-based admission control mechanism. For example, in the case of measurement-based approach, each router measures its traffic load and sends this information directly to the tree manager. The tree manager calls the admission control module, which will determine available bandwidth and then the acceptance/rejection policy through a group-tree matching algorithm (which will be covered later in this section). On the other hand, a parameter-based approach can also be easily employed. We will present a protocol using parameter-based approach in Section 4.

In the tree manager, the multicast routing module is responsible for computing multicast tree based on group user requirements. When a new group comes, if no proper existing tree is found, the tree manager needs to compute a new tree for the group. There might be many types of QoS requirements from the group users, such as bandwidth, delay and delay jitter among group users, etc. For bandwidth requirements, we will present along with the group-tree matching algorithm in Section 3.5. For delay requirements, they can be easily converted to bandwidth requirements. We will discuss heterogeneous QoS requirements in Section 3.6. Then, the main issue here becomes how to compute an efficient multicast tree which satisfies the requirement of delay jitter among group users (if demanded).

Efficient multicast tree computation. As we know, computing a Steiner tree (or optimal tree) for a multicast group is NP-hard. The NP-hardness justifies the use of some approximation algorithms, such as core based tree. Hence AQoSM adopts a PIM-SM/CBT like multicast routing approach. Each aggregated tree is associated with a core. The corresponding cores or RPs (Rendezvous Points) can be properly chosen to achieve load balancing. In this way, the functionality of tree manager (such as routing, admission-control, and group-tree matching) can be implemented in these cores in a distributed fashion.

Multicast tree computation with bounded delay jitter. In the literature, this problem is also referred to as Delay and Delay Variation Bounded Multicasting (DVBM) problem, with the goal of finding a tree (given a source and a set of receivers) that satisfies the QoS requirements on the maximum delay from the source to any of the receivers and on the maximum inter-destination delay variance. The DVBM problem is shown to be NP-complete [24]. And some heuristic algorithms have been proposed [24,42,43]. To leverage the cores, AQoSM employs a CBT-based heuristic algorithm similar to the one in [42]. And the core nodes will serve as a good candidate pool for "central nodes" in the algorithm.

To simplify the multicast address allocation, we borrow the idea of Simple Multicast [37], i.e., each aggregated multicast tree is identified by a combination of the IP address of its core and a multicast D-class address. In this way, the global address assignment problem is eliminated and the aggregated tree address can be totally controlled by the cores.

To further improve the state scalability, the aggregated trees in AQoSM are designed to be bi-directional. The main advantage is that, whenever a bi-directional tree covers the members of a group, it can be used for packet

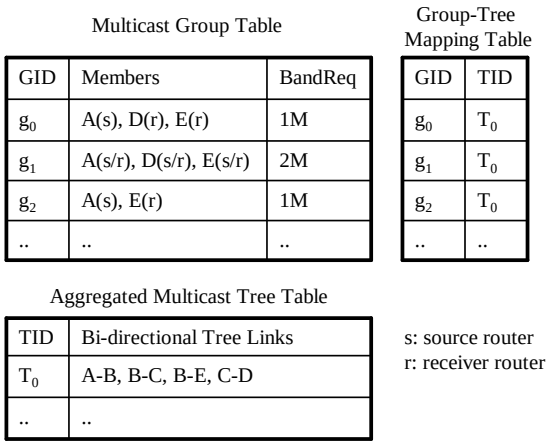


Fig. 4. Group and tree information bases in the tree manager.

delivery for the group, regardless of transmission direction (which is necessary for unidirectional trees, however). Consequently, more groups can share a tree, which leads to more state reduction and less maintenance overhead. This approach is particularly useful for interactive real-time applications such as video conferences and distributed network games, where most members of a group generally act as sources as well as receivers.

However, AQoSM does not exclude using unidirectional trees and other multicast routing protocols. Some QoS-aware routing protocols, such as QoSMIC [18], QMRP [10], and RIMQoS [22], etc. can also be applied.

3.5 QoS-Aware Group-Tree Matching Algorithm

To match a group to a tree, the tree manager needs to maintain established multicast trees, active multicast groups, and a group-tree matching table. One possible organization of the group and tree information is shown in Fig. 4. To simplify our discussion, we only put bandwidth as the QoS requirement of each multicast group in the table, yet other metrics, such as delay jitter, should be also considered (if any). The populated data of the tables in Fig. 4 can refer to Fig. 3. In this example, three multicast groups (group g_0, g_1, and g_2) share one aggregated bi-directional tree T_0.

Recall that a group can be perfectly or leakily matched to an aggregated tree (as explained in Section 2). Leaky match trades off bandwidth consumption for higher aggregation. In order to control the bandwidth waste incurred by leaky match, we define a bandwidth overhead threshold b_t: a group g can use T as its multicast tree only if the bandwidth overhead of T does not exceed the threshold b_t (the methods for calculating bandwidth overhead will be presented later).

On detection of a new multicast group g, the tree manager populates the corresponding entries of multicast group table and executes the QoS-Aware group-tree matching algorithm as shown in Algorithm 1.

Algorithm 1 GTMatch(g)

```
1: CTS(g) ← null //initialize candidate tree set
2: for all T ∈ MTS do
3:    //MTS is the existing multicast tree set
4:    if T covers g AND AC(T, g) succeeds then
5:       //AC is admission control
6:       compute the bandwidth overhead δ_A(T, g)
7:       if δ_A(T, g) < b_t then
8:          CTS(g) ← CTS(g) ∪ T
9:       end if
10:   end if
11: end for
12: if CTS(g) ≠ null then
13:   return a tree in CTS(g) with min. δ_A(T, g)
14: else
15:   compute the native tree T_A(g)
16:   if AC(T_A(g), g) succeeds then
17:      MTS ← MTS ∪ T_A(g)
18:      return T_A(g)
19:   else
20:      return null
21:   end if
22: end if
```

In this algorithm, the tree manager first traverses the list of existing trees MTS and selects a tree T as a candidate tree for group g if it satisfies the following conditions (line 4−10): 1) T covers all the members of g; 2) admission control of using T for g succeeds, i.e., enough bandwidth is available on the tree and g's service class [4]; and 3) the bandwidth overhead $\delta_A(T, g) \leq b_t$. If more than one candidate trees are found, the one with minimum $\delta_A(T, g)$ is used to cover g (line $12 - 13$). Update multicast group table and group-tree matching table.

On the other hand, if no candidate is found, the tree manager computes a "native" tree $T_A(g)$ for g using PIM-SM/CBT like core-based multicast routing algorithm (line 15), based on the multicast group membership, bandwidth requirement (and other requirements such as delay jitter) and available bandwidth of links (as discussed in Section 3.4). Note that, the routing module will choose a good core (or RP) node from all the candidates so that enough

[4] If delay jitter is also the users' concern, jitter check then becomes part of admission control.

bandwidth is available on the links. If this kind of candidate does not exist or not enough allocated bandwidth is available, the multicast group g will be rejected (line 20). Otherwise, $T_A(g)$ is used to cover g and is added to MTS (line $17-18$), and the corresponding tables are updated.

In the above group-tree matching algorithm, we have a missing piece: calculating bandwidth overhead of leaky match. Before we define bandwidth overhead, let us introduce some notations first. A network is modelled as an undirected graph $G(V, E)$. Each edge (i, j) is assigned a positive cost $c_{ij} = c_{ji}$ which represents the cost to transport a unit of data from node i to node j (or from j to i). Given a multicast tree T, total cost to distribute a unit of data over that tree is

$$C(T) = \sum_{(i,j)\in T} c_{ij}. \tag{1}$$

Assume an aggregated tree T is used to cover group g, which has a "native" tree $T_A(g)$, then the percentage bandwidth overhead of using T for g can be defined as

$$\begin{aligned} \delta_A(T, g) &= \frac{C(T) - C(T_A(g))}{C(T_A(g))} \\ &= \frac{C(T)}{C(T_A(g))} - 1, \end{aligned} \tag{2}$$

where $C(T)$ is the total link cost of tree T.

Change in QoS Requirements. When the tree manager detects the bandwidth requirement of group g changes, it simply checks whether the aggregated tree $T(g)$ has enough available bandwidth to accommodate g. If yes, the only thing needed is to update group bandwidth requirement information. If $T(g)$ can not accommodate g, the tree manager has to activate the above group-tree matching algorithm and find or establish another tree for g.

Change in Group Membership. Group membership dynamics might affect the choice of their delivery tree. However, in AQoSM, member join or leave incurs tree switching (i.e., switching a group from its pervious delivery tree to another tree) only when bandwidth overhead threshold is violated or some other QoS constraints are broken. In addition, the tree switching procedure can be done very quickly by changing group-tree matching entries (unless a new tree needs to be established).

3.6 Discussions

Heterogeneous QoS Requirements For high-bandwidth applications such as video-streaming, the inherent heterogeneity of current Internet has made

multicast a challenging problem, since there is no single rate that can fit the demand of receivers with different bandwidth and processing capabilities. A widely adopted solution is layered-multicast [29], in which the group members are divided into a number of homogeneous sub-groups. Even though this approach solves user heterogeneity problem, it exacerbates the state scalability problem, because multiple multicast trees are now needed for each multicast group. AQoSM, on the other hand, can be seamlessly integrated with this approach without significantly increasing the number of aggregated trees to manage since those sub-groups from a large main group usually have very similar tree shape in the backbone domains, and thus they can share one or very few aggregated trees.

Fault Tolerance Recovery from failures is a very important element in any QoS provisioning. AQoSM can provide very efficient and fast failure recovery. By using aggregated multicast, AQoSM significantly reduces the number of trees needed to maintain. Correspondingly, the recovery latency and cost of the trees will be largely reduced. If a node or link fails, only a few affected aggregated trees need to be rerouted, while all the groups sharing these trees will be restored all at once. For added reliability and small switch-over latency, a pre-planned approach can also be applied: backup trees are computed first when aggregated trees are set up; upon failure, backup trees are simply activated. Further studies on multicast fault tolerance could be found in [13].

3.7 Summary

To summarize, AQoSM seamlessly integrates multicast service into Diff-Serv without violating the design principles of Diff-Serv and without sacrificing the efficiency of multicast. In fact, it incorporates two types of traffic aggregation: multicast flow aggregation based on the shapes of multicast trees and QoS flow aggregation based on packet forwarding treatments. The first type of aggregation significantly reduces the multicast state information in core routers, and the second type only requires the core routers to maintain minimum QoS state information for packet forwarding behavior. These two types of traffic aggregation are in agreement with the "Diff-Serv mentality", where we want to keep the core simple and fast, and put as much intelligence as possible to the boundaries of the network.

AQoSM provides the three elements that are essential in any QoS provisioning: a) discovery of resources through QoS-aware routing, b) protection of resources through admission control, and c) recovery from failures through rapid reconfiguration. We have presented effective solutions for each major component of AQoSM. A key property of our architecture is its ability to reroute a multicast group quickly: in most cases, we just need to change the

"label" (i.e., tree ID) information at the ingress and egress routers. This property creates a cascade of advantages: 1) we separate the creation of a routing tree from data distribution; 2) we can adapt to dynamic group membership; 3) we can adapt to the changing QoS requirements of the group; 4) we can recover quickly from failures.

In conclusion, AQoSM is a promising architecture that supports QoS group communications in a scalable and efficient way.

4 An MPLS-based AQoSM Protocol and Implementation

The AQoSM architecture is devised with a variety of options. In this section, we present a detailed protocol, called MPLS-based AQoSM protocol (MAQoSMP), in which we implement AQoSM using MPLS technique. Our design goal is to achieve high state scalability and high resource utilization while satisfying QoS requirements of multicast groups with low overhead.

4.1 Overview

In MAQoSMP, the tree manager is implemented in a distributed fashion. We distribute the functionalities of tree manager into the core nodes within the backbone domain [5]. The set of possible cores are advertised using the bootstrap mechanism [16].

When an edge router receives a join message for a group g, it classifies this multicast flow into a Diff-Serv behavior aggregate based on the QoS service requested. To map this multicast group onto an aggregated tree, it determines a core using a hash function (which we call *group-to-core hash function*). This core is referred to as the default core c_0 for the group g. Upon receipt of a request from g relayed by the corresponding edge router, c_0 will find or compute a proper aggregated tree for group g by conducting group-tree matching algorithm and admission control as described in the previous sections.

When a multicast packet arrives at the ingress router, it is assigned a DSCP based on the behavior aggregate its multicast group belongs to, and labelled with the corresponding aggregated tree information. It should be noted that multicast tree aggregation and Diff-Serv flow classification are two independent mechanisms. Multicast flows belonging to different behavior aggregates

[5] We differentiate core routers and core nodes. Core nodes are "cores" of multicast trees, while core routers are "internal" routers compared with edge routers. Core nodes can reside in both core routers and edge routers.

can be mapped onto the same multicast tree, and vice versa. Thus, we can obtain maximum gain of tree aggregation without modifying Diff-Serv model. As the packet is transmitted inside the Diff-Serv domain, the intermediate routers look up the next hops for the aggregated tree, duplicate the packet, and forward it according to the per-hop behavior (PHB) specification for its DSCP. At the egress routers, the packet is restored and delivered towards the destinations.

In the following sections, we discuss the details of some design issues and the protocol implementation.

4.2 Admission Control and Statistical Multiplexing

The admission control policy is crucial to ensure QoS guarantees and achieve high resource utilization. Our protocol adopts a parameter-based approach, because it is able to provide strongly guaranteed and differentiated services to multiple forwarding classes by applying different admission control policies. Furthermore, it is easier and causes less (measurement) overhead compared to a measurement-based approach. On the other hand, however, how to calculate the "real" resource required by users, especially for VBR (Variable Bit Rate) traffic, is extremely important for a parameter-based approach since it significantly affects the resource utilization.

In MAQoSMP, admission control is done at aggregated multicast tree level, i.e., for each aggregated tree, effective bandwidth is computed. Since many groups share one aggregated tree, a statistical multiplexing model can be applied according to different types of traffic (Markovian traffic [15] or self-similar traffic [32]). In this way, we can combine the advantages of link-level admission control and flow-level admission control by employing statistical multiplexing for each aggregated tree: the computational overhead is reduced and the resource utilization is improved.

Based on statistical multiplexing models, such as [15] and [32], the service requirement of small loss probabilities can be translated into effective bandwidth. Using this method, the admission control module in each core is able to calculate the effective bandwidth of the aggregated trees it manages, based on the flow parameters (e.g., arrival rates) and QoS requirements (such as loss ratio) of their groups. Then the core nodes can exchange this information periodically among themselves to obtain a global picture of link bandwidth usage, based on which the admission control module will make acceptance/rejection decisions. In this way, tree manager can calculate and maintain aggregated effective bandwidth for each delivery tree, rather than for each link; therefore, our method is very flexible and efficient.

4.3 Tree Management Using MPLS Technique

In MAQoSMP, aggregated multicast trees are managed using MPLS technique. In other words, MAQoSMP manages MPLS aggregated multicast tree (or MPLS tree for short). In this approach, multicast packets are assigned MPLS-like labels at the ingress routers, then transmitted along the established MPLS tree, and the labels are removed at the egress routers. The implication of this approach is that the creation and management of trees becomes very efficient. In addition, group rerouting becomes very quick and simple: it is a matter of assigning different labels on its packets at the entrance points. This way, we can easily adapt to changes in the QoS requirements and group member dynamics.

To efficiently establish MPLS trees, we devise a label distribution protocol for bi-directional trees (which are adopted by MAQoSMP to improve state scalability). It is important to note that even though there exist solutions to distribute labels for unidirectional multicast trees [33], no research work has been found for label distribution of bi-directional trees in the literature. We propose a distributed solution. It extends the existing unidirectional MPLS tree setup schemes [33]: root-initiated or leaf-initiated. The idea is as follows: a bi-directional tree can be decomposed into n unidirectional trees (n is the number of the leaf routers in the bi-directional tree), each of which has a "root" router. Thus, the tree manager can send the n unidirectional tree objects to the corresponding "root" routers. Then each "root" router uses root-initiated unidirectional MPLS tree setup approach. Leaf-initiated approach can be used similarly. More details about root-initiated approach and leaf-initiated approach can be found in [33]. To tear down an existing bi-directional multicast tree, tree manager only notifies the leaf routers of this tree, and each leaf router sends label withdraw message to its upstream LSRs. Note that there are alternative approaches to establish MPLS trees. For example, MPLS trees can be established using Multipoint-to-Point LSPs [41], and multicast traffic flows on the same tree can be routed on the corresponding LSPs. In this way, the number of required LSPs and labels can be greatly reduced compared with Point-to-Point LSPs. The mapping of aggregated trees to Multipoint-to-point LSPs is worth further investigation and thus not presented here.

4.4 Considering Dynamic Cores

In the default group-tree matching algorithm, for each group g we only consider the trees within its default core c_0. We design a new functionality: core switch, where a group's core can be changed dynamically, for the following reasons. First, core switch can achieve better aggregation and lower rejected

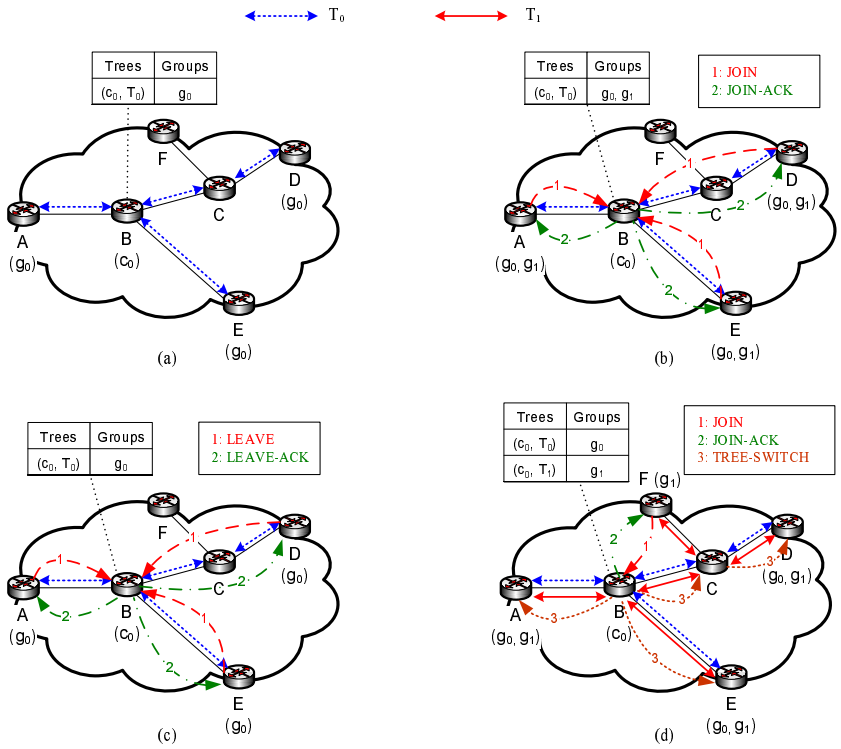


Fig. 5. (a) Initial state: group g_0 uses tree (c_0, T_0); (b) Member join: group g_1 starts with members A, D, and E, and groups g_0 and g_1 share the tree (c_0, T_0); (c) Member leave: group g_1 terminates; (d) Tree switch: based on (b), a new member F joins group g_1, and g_1 switches from (c_0, T_0) to (c_0, T_1).

join requests: if there are no existing trees (including "native" tree) to cover a group g in the default core, an existing (or new) tree in another core can be used for the data delivery of group g. In this way, more groups will share a single delivery tree and fewer groups may be rejected. Second, from the perspective of the global network, the backbone domain is likely to be unevenly loaded. The construction of one poorly unbalanced group may result in the rejection of several future groups. Therefore, it is important to balance the link loads in the backbone domain. Here, core switch allows the tree manager to select a core c' with most balanced candidate tree. As a result, link loads are more balanced and potentially fewer groups may be rejected. Of course, these benefits come at the cost of additional communication between cores. We will describe the implementation of core switch mechanism in Section 4.5.5.

4.5 Protocol Details

In MAQoSMP, we define the following control messages: *JOIN*, *JOIN-ACK*, *LEAVE*, *LEAVE-ACK*, *TREE-SWITCH*, *CORE-SWITCH-REQ*, *CORE-SWITCH-ACK*, and *CORE-CHANGE*. These control messages are mainly transport level messages between edge routers and cores.

4.5.1 Cores and Edge Routers

The responsibilities of cores include QoS-aware multicast routing, group to tree matching, admission control, and MPLS tree management. At the edge of the network domain, source edge routers perform traffic conditioning, marking, and labelling for incoming packets and the receiver routers, upon receiving packets, drop MPLS labels and further deliver packets to other domains or customer networks.

4.5.2 Member Join

When an edge router r receives a request to join a group g from outside domains, it first uses the group-to-core hash function to get g's default core c_0, and then sends a message $JOIN(g)$ to c_0. c_0 triggers its tree manager module to find or establish an appropriate aggregated tree (e.g. (c', T), since an aggregated tree is identified by a combination of the core's IP address and a class D address). It should be noted that this join message might activate tree switch or core switch if the existing tree could not cover group g (the details will be discussed in the following subsections). Then the corresponding group-tree matching entry is sent back to r through a message $JOIN\text{-}ACK(g, (c', T))$. r adds this entry to its group-tree matching table for the purpose of assigning MPLS labels to incoming packets, and employs the distributed bi-directional MPLS tree setup procedure if this tree has not been constructed.

4.5.3 Member Leave

Similarly, when an edge router r wants to leave a group g, it sends a $LEAVE(g)$ message to its core c'. On receiving of the $LEAVE$ message, c' manipulates the group-tree matching algorithm, which might also cause tree switch or core switch. As the tree manager finds that all members in a group leave, it first sends $LEAVE\text{-}ACK(g, (c', T))$ message to notify the leaf routers of the tree, and then updates its own tables. If the tree is now obsolete, that is, when all groups mapped onto a MPLS tree terminate, the leaf routers remove label forwarding entries and propagate label withdraw messages to upstream routers to destroy the aggregated tree.

4.5.4 Tree Switch

In MAQoSMP, membership dynamics are efficiently supported though tree matching, tree switching and core switching. Tree switch procedure (in the same core) is triggered when the membership of a group g is changed and its original aggregated tree (c_0, T) is not able to cover the group with reasonable overhead or satisfy QoS requirement. In this case, c_0 finds or establishes

an appropriate tree for g, say, (c_0, T'), and then it utilizes a multicast message *TREE-SWITCH*$(g, (c_0, T'))$ to notify all the members of group g to join (c_0, T') and leave (c_0, T). The member routers may trigger MPLS tree establishment or release if needed.

Examples of member join, member leave, and tree switch are illustrated in Fig. 5.

4.5.5 Core Switch

As explained earlier, core switch mechanism helps to improve tree aggregation and balance the link loads in the backbone domain. For example, when membership has been changed for a group g and the original tree in the default core c_0 is no longer appropriate, the default core c_0 queries other cores by sending out a multicast message *CORE-SWITCH-REQ*$(g, member\ list, QoS\ requirements)$ though a pre-defined bi-directional multicast tree, which connects all cores. Upon receiving the *CORE-SWITCH-REQ* message, each core activates its tree manager module to check if there is a good tree to cover group g. If there is, a unicast message *CORE-SWITCH*$(g, (c', T'), bandwidth\ overhead)$ is sent back to c_0. From the received messages, c_0 chooses the one with smallest bandwidth overhead, say, (c'', T'') as the delivery tree for g. Similar to tree switch, this will trigger a unicast message *CORE-SWITCH-ACK*$(g, (c'', T''))$ back to the new core c'' and a *TREE-SWITCH*$(g, (c'', T''))$ message through the multicast tree (c_0, T) to notify all the group members of g to switch to the new tree. At the same time, c_0 needs to record the new core c'' for further requests for group g. For instance, when a new member joins group g, it sends a *JOIN* message to the default core c_0 and c_0 needs to forward this request to c''. Finally, c_0 will send a message *CORE-CHANGE*(g, c'') back to the new member (if core switch has been caused by a member join). When the load balancing option is turned on, the most balanced multicast tree can be selected in a similar fashion. Fig. 6 illustrates the core switch procedure through an example of member leave.

4.6 Discussions

In summary, MAQoSMP provides scalable and efficient QoS multicast support in MPLS/Diff-Serv networks. It is worth pointing out that this benefit is achieved at the cost of additional overhead in comparison with traditional multicast schemes, e.g., computation overhead for executing group tree matching algorithm and bandwidth overhead due to leaky match between groups and trees. When members join/leave a group, tree manager tries to map this group to an appropriate tree. A simple and straightforward approach requires scan-

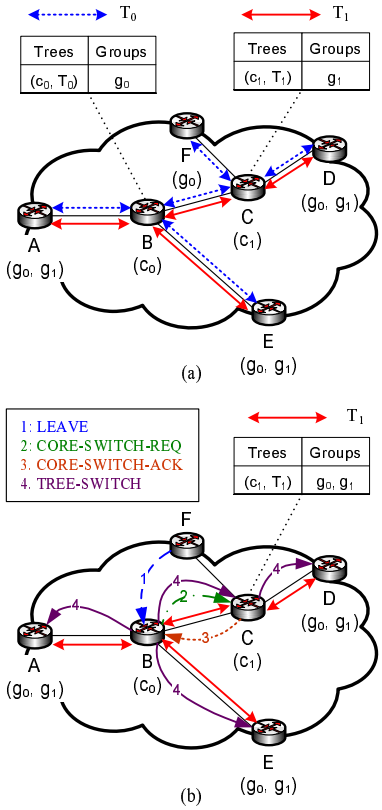


Fig. 6. (a) Before core switch: group g_0 uses tree (c_0, T_0) while group g_1 uses tree (c_1, T_1); (b) Core switch procedure: when member F leaves group g_0, g_0 switches from core c_0 to c_1 and shares the tree (c_1, T_1) with group g_1.

ning all existing trees once, and thus the computation complexity is mainly determined by the number of multicast trees [6]. Additionally, some bandwidth is wasted when groups are mapped onto trees in a "leaky" fashion. By using a bandwidth waste threshold b_t, we are able to control the amount of bandwidth overhead. Note that when b_t is increased, fewer trees will be established, and thus the computation overhead is reduced; on the other hand, more bandwidth will be wasted. Therefore, there is a trade-off of computation overhead vs. bandwidth overhead when b_t is varied. We will evaluate the overhead in the following section.

[6] There are some obvious approaches to improve the algorithm performance. For example, existing multicast trees can be maintained in a sorted list according to their cost. Depending on the cost of the native tree of a group $C(T_A(g))$ and the bandwidth overhead threshold b_t, only a fraction of trees with cost within the range $[C(T_A(g)), (b_t + 1) \times C(T_A(g))]$ needs to be scanned.

5 Performance Evaluation

In this section, we conduct simulations in NS-2 [2] and evaluate the performance of MAQoSMP, especially on the aspects of scalability, overhead, load balancing and statistical multiplexing.

5.1 Performance Metrics

In our simulations, we use the following metrics to quantify the performance of MAQoSMP.

Number of MPLS Trees is the average number of MPLS trees maintained in the tree manager. This metric is an indirect measurement for the multicast tree maintenance overhead and the computation overhead for group tree matching. The more multicast trees, the more memory required and the more processing and computation overhead involved in the tree manager.

Number of Label Forwarding Entries is the average number of label forwarding entries installed in all the routers (including the core routers and edge routers). This metric reflects the memory requirement and forwarding processing overhead in the routers. The fewer label forwarding entries, the less memory required and the faster labels forwarded.

Request Rejection Ratio is defined as

$$RR_{ratio}(t) = \frac{N_R(t)}{N_A(t)}. \tag{3}$$

where $N_A(t)$ denotes the number of group requests arriving in time period t after steady state is reached and $N_R(t)$ denotes the number of group requests which are rejected.

Tree Setup Ratio is defined as

$$TS_{ratio}(t) = \frac{N_A(t) - N_M(t) - N_R(t)}{N_A(t)}. \tag{4}$$

where $N_A(t)$ and $N_R(t)$ are defined as above. $N_M(t)$ denotes the number of group requests which can be matched to some existing trees. $TS_{ratio}(t)$ gives a measurement of tree setup overhead: the higher $TS_{ratio}(t)$ is, the higher MPLS tree setup rate.

Real Bandwidth Waste Ratio is the percentage of bandwidth wasted due to leaky match between groups and trees. It quantifies the bandwidth overhead

of MAQoSMP.

Delay and Loss Ratio measure average end-to-end performance of the multicast trees. Delay is the amount of time to deliver a packet from a source to a receiver, which includes propagation, transmission and queueing delay. Loss ratio is defined as the percentage of data packets lost due to buffer overflow. Low delay and small loss ratio are especially desirable for real-time applications such as video conferencing and network games.

5.2 Simulation Environments

The network used for the simulation results presented here is abstracted from a real network topology, Abilene backbone [1], which has 12 core routers. Since there are no edge routers in the backbone, we attach an additional node as an edge router to each core router.

In our simulation, we use our group model developed in [14], **GEM** (GEneralized Membership Model), to generate group members. GEM is a realistic model constructed from multicast traffic measurements on MBONE and net games, and it aims to re-generate multicast traffic consistent with the real traffic. It defines a metric called *Group Participation Probability*, which characterizes the probability that each router participates in multicast groups: for two nodes i and j with participation probability p_i and p_j, let N_i be the number of groups that have i as a member and N_j be the number of groups that have j as a member, then it is easy to prove that, on average, $\frac{N_i}{N_j} = \frac{p_i}{p_j}$.

Following this model, we assign participation probability to each router in the target network. Core routers will not be members for any multicast group and are assigned probability 0. Any other edge router is assigned a probability of 0.2 or 0.8 according to the real-time traffic of its corresponding core router. For a router, more traffic means more participation in the network communication, thus there is higher probability for it to join a multicast group. As to bandwidth capacity, we take the real values for outgoing links of all core routers, while for links from edge routers to core routers, we assume they have infinite capacity which will not affect the admission control.

In our simulation experiments, multicast session requests arrive as a Poisson process with arrival rate λ. Sessions' life time has an exponential distribution with average μ. At steady state, the average number of sessions is $\bar{N} = \lambda \times \mu$. We define three types of multicast groups: low bandwidth (10K), medium bandwidth (100K), and high bandwidth (1M). Of all the incoming groups, 50% are low, 30% are medium, and 20% are high [7]. These groups belong to

[7] These bandwidth requirements are typical for several common applications on

Table 1
A High Level Comparison of Simulated MAQoSMP and Native QoS-aware PIM-SM/CBT MPLS Multicast (Native PIM-SM/CBT)

Name	*Multicast Routing*	*MPLS Tree*	*Group-Tree Matching*	*QoS-aware*
MAQoSMP	PIM-SM/CBT like	Yes	Yes	Yes
PIM-SM/CBT	PIM-SM/CBT like	Yes	No	Yes

three Diff-Serv classes, i.e., 10% are EF (Expedited Forwarding), 40% are AF (Assured Forwarding), and 50% are BE (Best Effort). Accordingly, a certain amount of bandwidth is reserved for each traffic class on all links, and separate queues are used to isolate flows of different classes. Performance data is collected every 10 simulation seconds as "snapshots" after steady state is reached (e.g. at $T = 10\mu$). The results shown below are the average values of multiple "snapshots". Note that we have varied simulation parameters (e.g., uniform vs. non-uniform percentage of groups with different bandwidth requirements and Per Hop Behaviors) in simulations, and the results show very similar trend.

5.3 Results and Analysis

We design experiments to compare MAQoSMP vs. native QoS-aware PIM-SM/CBT MPLS multicast (native PIM-SM/CBT for shorthand), where an MPLS tree is simply constructed using PIM-SM/CBT protocol for each multicast group. A high level comparison of simulated MAQoSMP and native PIM-SM/CBT is shown in Table I. In our experiments, MAQoSMP employs bi-directional trees, and each member of a group can be a source and a receiver. Once a multicast session starts up, its core node (or RP) is randomly chosen from the 12 core routers in the network. For MAQoSMP, the algorithm specified in Section 3.5 is used to match a group to a tree. The corresponding routing algorithm is PIM-SM/CBT like routing algorithm which is also used for native PIM-SM/CBT. In MAQoSMP, if the tree computed based on the original core can not accommodate the group, a new RP will be selected among the other RP candidates until a good tree is found or the group is rejected because of no enough bandwidth available. In native PIM-SM/CBT, if the original core cannot find an appropriate tree with sufficient bandwidth for the group, the group is simply rejected.

---

the Internet, e.g., text message applications (10K), voice/audio or low-quality video applications (100K), and high-quality video applications (1M). Similar settings for the percentage of bandwidth requirements have been used in previous works such as [38].

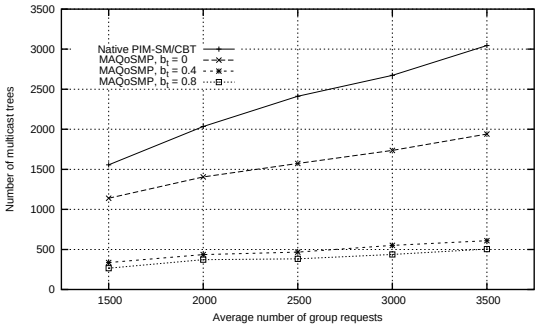


Fig. 7. Number of MPLS Trees vs. number of group requests.

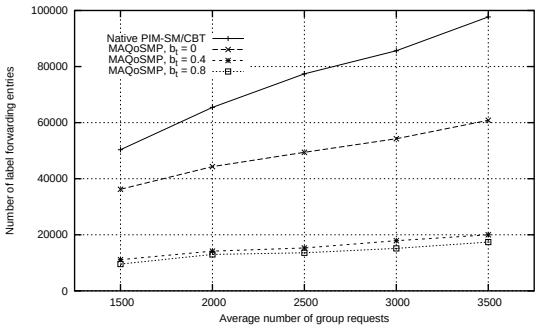


Fig. 8. Number of Label Forwarding Entries vs. number of group requests.

5.3.1 Multicast State Scalability

In our experiments, we vary the bandwidth overhead threshold b_t, an upper bound on the real bandwidth waste, from 0 to 0.8 for MAQoSMP. Fig. 7 shows the results for **Number of MPLS Trees** vs. the average number of group requests. We can see that MAQoSMP "scales" with the number of concurrent groups: for native PIM-SM/CBT, the number of MPLS trees grows almost linearly with the number of groups; for MAQoSMP, as the number of groups becomes bigger, the number of trees also increases, but the increase is much less pronounced than that of native PIM-SM/CBT, even for perfect match ($b_t = 0$). When there are 3500 group requests, the number of trees is only 505 instead of 3046 for $b_t = 0.8$, indicating that much less tree maintenance overhead and computation overhead are involved in the tree manager. Also the "increase" decreases as there are more groups, which means that as groups are pumped into the network, more groups can share a single MPLS tree.

Fig. 8 plots the change of **Number of Label Forwarding Entries** with the number of group requests. It has a similar trend as the metric **Number of MPLS Trees**. The number of label forwarding entries is reduced from 97676 to 17378 (above 80% reduction) for $b_t = 0.8$ when 3500 groups come. Thus, we can conclude that, in MAQoSMP, the label maintenance and forwarding process overhead are significantly reduced.

In Fig. 9, we demonstrate the effect of the number of active groups on **Tree**

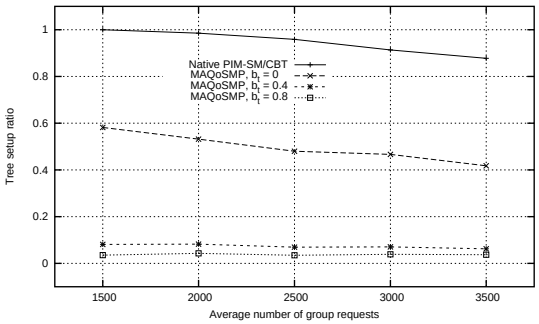


Fig. 9. Tree Setup Ratio vs. number of group requests.

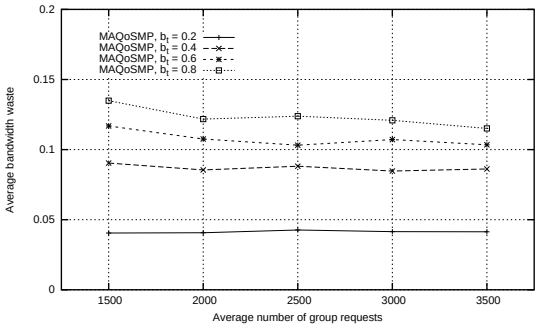


Fig. 10. Real bandwidth waste vs. number of group requests.

Setup Ratio. From the figure, we can see that the tree setup ratio decreases with the number of groups, which is consistent with the previous analysis: more groups share a single MPLS tree when the number of groups is bigger, and thus fewer trees need to be set up. Comparing MAQoSMP and native PIM-SM/CBT, tree setup ratio is much smaller in MAQoSMP, which means the tree setup overhead is dramatically reduced.

From Fig. 7, Fig. 8, and Fig. 9, a general observation is that, when bandwidth overhead threshold is increased, that is, more bandwidth is wasted, Number of MPLS Trees, Number of Label Forwarding Entries, and Tree Setup Ratio decrease, which translates into less tree and label management overhead and group-tree matching computation overhead. Therefore, there is a trade-off between management overhead reduction and bandwidth waste. The balance depends on the network administration policy.

Recall that bandwidth waste threshold is an upper bound on the real bandwidth waste, we want to ask a question: is this metric a loose or tight upper bound? Fig. 10 provides the answer for our simulation scenario. When $b_t = 0.4$ and 0.8, the corresponding Real Bandwidth Waste Ratios are approximately 0.085 and 0.12, respectively. This means that the performance improvement shown in the previous figures is achieved with very limited bandwidth waste. Furthermore, this result indicates that the bandwidth waste threshold needs to be determined empirically in order to control real bandwidth waste.

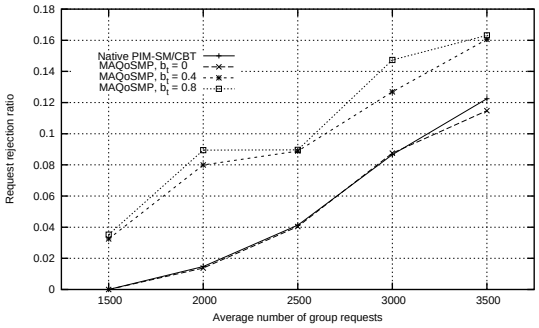


Fig. 11. Request Rejection Ratio vs. number of group requests.

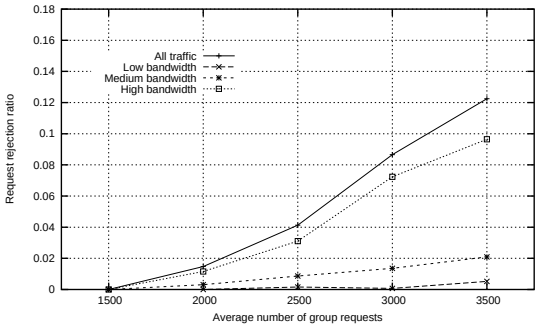
Fig. 11 depicts how the aggregation affects **Request Rejection Ratio**. The figure shows that the request rejection ratio is influenced by the aggregation under leaky match cases. Intuitively, leaky match causes some bandwidth waste, thus it should have some effects on admission control: the more bandwidth waste, the bigger request rejection ratio. This trend is clearly reflected in the figure.

To further illustrate how MAQoSMP affects groups with different bandwidth requirements and QoS classes, we plot the request rejection ratio for each type of groups in Fig. 12 and 13 respectively. Fig. 12 shows rejection ratio for low, medium and high bandwidth groups. From Fig. 12(a) and (b), it is clear that MAQoSMP achieves similar rejection ratio as PIM-SM/CBT for different types of bandwidth requests when $b_t = 0$. In addition, as shown in Fig. 12(c), as b_t increases, the relative percentage of rejected requests among the three types of groups are fairly static, despite the fact that the overall rejection ratio is higher due to leaky match. We make similar observations for groups belonging to different PHB classes from Fig. 13: the percentage of rejected requests among BE, AF and EF groups is approximately equal to the percentage of join requests from these groups, irrespective of how b_t varies. Therefore, we conclude that, in comparison with traditional multicast protocols, MAQoSMP does not favor or disfavor any particular type of groups with certain bandwidth requirements and QoS classes.

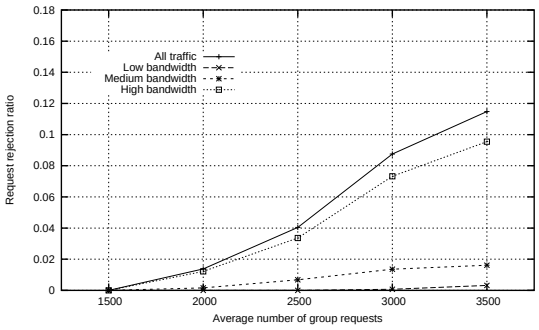
5.3.2 Load Balancing (LB)

In MAQoSMP, core switch mechanism enables a group to find a tree with good performance, such as higher state aggregation and more balanced load. In this subsection, we evaluate the benefits of load balancing in MAQoSMP. We use the Coefficient of Variation (which is equal to the standard deviation divided by the mean) of link loads caused by a multicast tree to measure the load balancing property of this tree: lower Coefficient of Variation means the tree is more balanced.

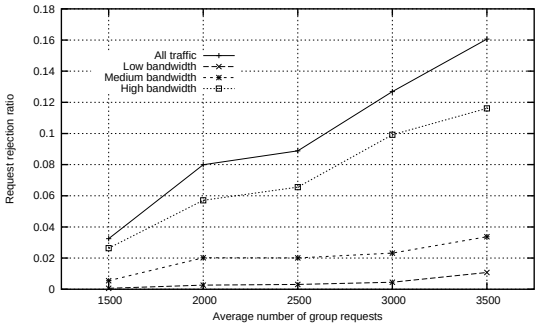
To illustrate the benefit of load balancing, we compare the performance of



(a)



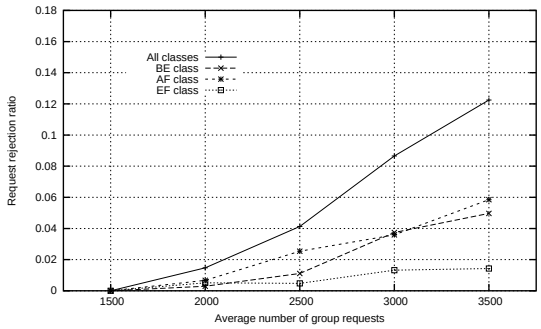
(b)



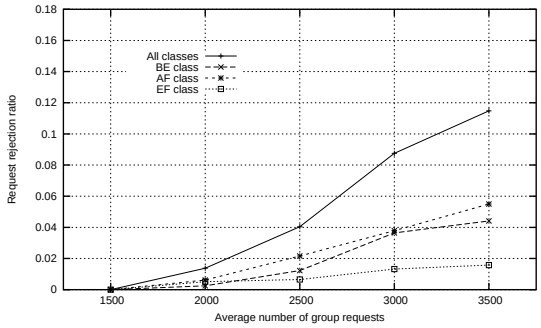
(c)

Fig. 12. Request rejection ratio. (a) PIM-SM/CBT; (b) MAQoSMP with $b_t = 0$; (c) MAQoSMP with $b_t = 0.4$.

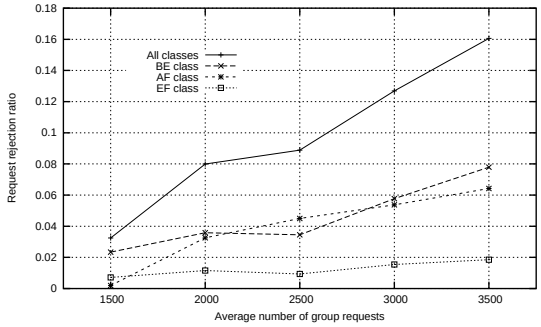
native PIM-SM/CBT, MAQoSMP without load balancing, and MAQoSMP with load balancing. The results are plotted in Fig. 14 and Fig. 15, which show the Correlation of Variation of link loads and rejection ratio, respectively. Obviously, as shown in Fig. 14, the link loads are more evenly distributed when load balancing option is turned on. Correspondingly, load balancing allows more groups to be admitted, especially for high bandwidth waste threshold (Fig. 15). For example, when $b_t = 0.4$, approximately $1 - 3\%$ more groups can be accommodated with load balancing feature, albeit at the cost of higher control overhead.



(a)



(b)



(c)

Fig. 13. Request rejection ratio. (a) PIM-SM/CBT; (b) MAQoSMP with $b_t = 0$; (c) MAQoSMP with $b_t = 0.4$.

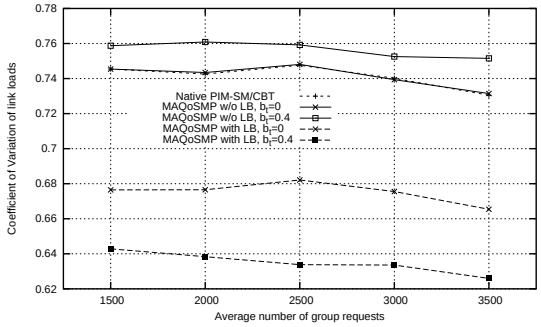


Fig. 14. Coefficient of Variation of link loads vs. number of group requests.

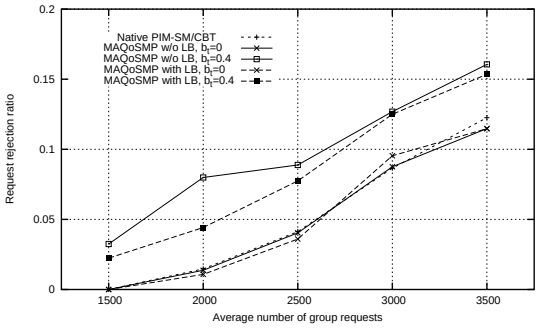


Fig. 15. Request Rejection Ratio vs. number of groups group requests.

5.3.3 Statistical Multiplexing (SM)

Besides multicast state scalability and load balancing, another important feature of MAQoSMP is to use statistical multiplexing technique to reduce the amount of bandwidth to be reserved for multicast groups while still satisfying certain QoS requirements, such as packet loss ratio. When statistical multiplexing is used, the tree manager computes the effective bandwidth for the incoming group, and decides whether to accept it based on residual link capacities. Otherwise, the tree manager will simply use the peak rate of the group for admission control.

We compare the performance of MAQoSMP with and without statistical multiplexing with respect to three metrics: loss ratio, delay, and request rejection ratio. In this series of simulation experiments, the average number of groups in steady state is 100 groups, and the buffer size for each group is increased from 1 to 10000 packets. The loss ratio is fixed to 0.001%. Each source has two states "on" and "off", which are modelled by a two-state Markov Chain. When the source is in "on" state, it transmits at a constant peak rate, whereas in "off" state, it pauses data transmission. The duration of on/off state is exponentially distributed. In our simulations, the mean burst period lasts for 0.1 sec, and the mean idle period is 0.9 sec.

As shown in Fig. 16, statistical multiplexing saves bandwidth reserved for each group and reduces group request rejection ratio significantly. For example, 32% more groups can be admitted with buffer size of 10000 packets per group when statistical multiplexing is used. We also observe that with statistical multiplexing, more groups can be admitted as buffer size increases. The reason for this observation is simple: large buffers can accommodate more bursty flows since overflow is less likely to occur, and thus the effective bandwidth of the same group decreases. On the other hand, because peak rate is always used in admission control when the traffic is not multiplexed, buffer size does not affect the rejection ratio.

Even though statistical multiplexing can accept more groups, one of our concerns is that it may degrade the QoS performance of the multicast groups.

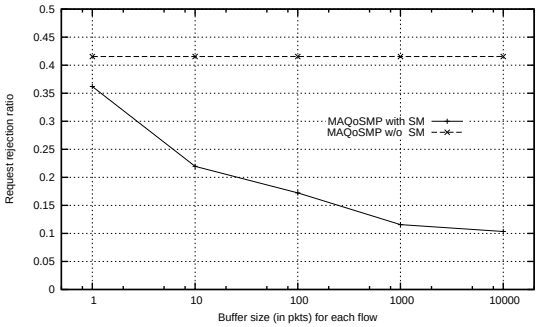


Fig. 16. Request Rejection Ratio vs. buffer size for each group.

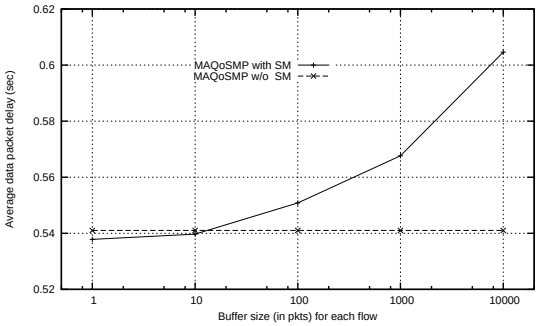


Fig. 17. Average data packet delay vs. buffer size for each group.

Fortunately, experiment results reveal that the loss ratio remains at 0 for all cases in our simulation scenarios. We also plot the average data packet delay vs. buffer size in Fig. 17. It is evident that when buffer is smaller than 100 packets for each group, the packet delays of MAQoSMP with and without statistical multiplexing are comparable. With larger buffers, multiplexed traffic tends to have longer queueing delays, since in this case, more groups are admitted and the traffic intensity increases in the network. However, the increase in packet delays is limited: in the worst case when buffer size is as large as 10000 packets per group, the delay grows from 0.538 to 0.605 second. From another point of view, this statistical multiplexing feature provides a trade-off between higher request acceptance and degraded QoS performance, which gives the network administrator additional flexibility to engineer the parameters to maximize ISP's benefits while satisfying users' QoS requirements.

Through our simulation studies, we can see that MAQoSMP achieves scalability by: (1) tree maintenance and management overhead reduction; (2) label maintenance and forwarding process overhead reduction. MAQoSMP also achieves load balancing by using core switch mechanism and higher resource utilization with statistical multiplexing technique.

6 Related Work

In this section, we provide a brief overview on related work. We first present some representative QoS multicast protocols for the Internet, and then review multicast schemes with QoS support in MPLS or Diff-Serv domains.

6.1 QoS Multicast Support

There is a large body of works on QoS multicasting in the Internet. In this section, we briefly review some representative protocols. Interested readers are referred to two comprehensive surveys [47,52] and references therein for details.

YAM (Yet Another Multicast protocol) [9] and QoSMIC (Quality of Service sensitive Multicast Internet protoCol) [18] rely on flooding techniques to find multiple paths, from which a best path in terms of QoS is selected. YAM builds shared trees through a spanning join mechanism: a receiver searches on-tree nodes through expanding ring search and finds the best path as a multicast tree branch. To restrict the flooding scope of YAM, QoSMIC adopts a two-phase searching scheme. The first phase is local search, which is similar to the spanning join in YAM. When no on-tree node is found in the local search, the second phase, tree search, is invoked. The new member contacts a designated manager router, which causes a set of on-tree routers to propose themselves as candidates. After receiving bid messages from these candidate routers, the new router chooses the best path to graft onto the multicast tree.

YAM and QoSMIC incurs high communication overhead by using flooding techniques. To alleviate this problem, a QoS-aware Multicast Routing Protocol (QMRP) is proposed in [10]. It starts with a single-path mode, attempting to use the default unicast path between the new receiver and the nearest on-tree router as the tree branch. If this path does not satisfy the QoS requirements, the protocol switches to a multi-path mode, in which multiple tree branches are searched and the best one is chosen.

RIMQoS (Receiver-Initiated QoS multicast protocol) [22] targets at building source-rooted multicast trees with multiple QoS constraints. It assumes an underlying QoS unicast routing protocol is available. In this protocol, a node joins the tree via a low-cost path computed by the unicast routing protocol and may later switch to a higher-cost path that meets the QoS requirements.

There are also some other multicast protocols supporting QoS, such as QMBF (QoS-aware Multicast Protocol using Bounded Flooding) [27] and QoSM^{2}P (Quality of Service Multipath Multicast Protocol) [3]. In spite of the large

number of existing studies on QoS multicasting in the Internet, most work focuses on how to establish multicast trees to improve QoS metrics or to satisfy given QoS constraints, without touching the multicast state scalability problem.

6.2 Multicast in MPLS/Diff-Serv Domains

To enable QoS multicasting and traffic engineering in the Internet, some proposals about supporting multicast in MPLS networks have been proposed [33,34], but their focus is to construct a multicast traffic engineering tree by building multicast trees immediately on L2 or mapping L3 trees onto L2. Edge Router Multicasting (ERM) [53] converts a multicast flow into multiple unicast flows by limiting branching nodes of multicast trees to only the edge routers. It simplifies the multicast LSP setup and avoids L3 forwarding, while trading off tree cost and end-to-end delay.

Diff-Serv has been proposed as a promising architecture for providing scalable QoS support. As explained earlier, there are some conflicts between multicast and Diff-Serv, such as the requirements of stateful core vs. stateless core. To tackle these problems, many research efforts have been devoted to alleviating multicast state scalability.

Z. Li et al. presented QoS-aware Multicasting in Diff-Serv domain (QMD) [28] to reduce the multicast state. In this scheme, the edge routers are required to process control requests and maintain control state, and a limited number of core routers maintain multicast state. The multicast trees are constructed such that the number of stateful core routers is minimized and the QoS requirements are met. In QUASIMODO [4], only a fraction of routers are multicast-capable and multicast packets are tunnelled between multicast routers. Obviously, this approach sacrifices the bandwidth efficiency of native multicast.

To improve the scalability of Diff-Serv-aware multicasting in a further step, some approaches are proposed to completely eliminate multicast state in core routers. Similar to Xcast [7], DiffServ Multicast (DSMcast) [48] proposes to encapsulate multicast tree information in packet header. It improves the scalability in terms of number of groups while suffering from the same bandwidth and processing overhead as Xcast. This model is further extended to provide heterogeneous QoS to multicast groups by including DS fields in packet header to allow the DSCP to dynamically adapt to the needs of downstream receivers as the packets traverse the network [46]. Edge-Based Multicasting (EBM) [45] offers an alternative approach to keep core routers stateless. In this protocol, an entity called Multicast Broker (MB) constructs multicast trees by cluster-

ing egress routers with similar QoS requirements and then linking the clusters. Multicast packets are tunnelled from edge to edge and are only replicated at edge routers.

It is important to note that the above-mentioned schemes attempt to reduce or even eliminate multicast state inside core routers for individual multicast groups, whereas our work strives for improving multicast state scalability by exploiting inter-group state aggregation. Therefore, our approach is orthogonal to the existing schemes, and can be combined with some of them (e.g., QMD).

In addition to these schemes, some other work targets at different problems of multicasting in Diff-Serv domains. For instance, DiffServ-Aware Multicasting (DAM) [54] aims to solve the Neglected Reserved Sub-tree (NRS) problem and to accommodate heterogeneous QoS requirements. Call Admission Multicast Protocol (CAMP) proposed in [36,35] performs measurement-based admission control for dynamic groups with bandwidth guarantees.

7 Conclusions

In this paper, we propose and develop a multicast architecture to support QoS in a scalable way. The main innovation is that we separate the logical entity of a group from that of a distribution tree. Many groups can be multiplexed on a single tree by appropriate labelling of the packets in an MPLS fashion. Also, a group can use different distribution trees during its lifetime. This logical separation has two main advantages: a) it facilitates the management of trees and of QoS provision, and b) it enables fast re-routing of groups. As a result, our architecture can provide load balancing, adaptability to changing conditions, and fault-tolerance. In addition, our architecture reduces the multicast state at core routers.

An alternative way to look at our architecture is through the trade-offs it enables.

- A level of indirection (group to tree mapping) versus more flexibility (fast group rerouting).
- Requirement of additional functions and information versus ability to provide scalable and efficient QoS support.
- Wasted bandwidth (used by packets travelling towards non-receivers) versus reduction of routing state at core routers.
- High resource utilization (achieved by statistical multiplexing) versus limited QoS performance degradation.

It is important to note that having these trade-offs we can operate at the ap-

propriate point in the possible spectrum. Thus the administrator of a domain can customize the architecture to the operational needs.

We conduct simulations to quantify the claims of the architecture. We compare our scheme with a native QoS-aware PIM-SM/CBT protocol. The results can be summarized in the following points:

- The number of Label Forwarding entries is reduced significantly with our approach by a factor of up to five.
- The overhead of setting up a tree is better amortized as the number of groups increases.
- Our protocol can achieve high resource utilization by accommodating more users than traditional multicast. The advantage comes from the ability to statistically multiplex different multicast flows on the same aggregated tree and explore many trees for a given group for the purpose of resource discovery and load balancing.

Future Work. We would like to study the effect of various scenario parameters on the performance of the architecture. We want to consider parameters such as the topology, the distribution of sources and receivers, and the locality of preference. We also want to investigate the design issues of incorporating measurement-based admission control approaches in our AQoSM architecture.

References

[1] Abilene network topology. *http://www.ucaid.edu/abilene/*.

[2] The Network Simulator - ns-2. *http://www.isi.edu/nsnam/ns/*.

[3] P. Baccichet, E. Pagani, and G. P. Rossi. Quality of service multipath multicast protocol. In *Proceedings of Networked Group Communication (NGC)*, Oct. 2002.

[4] G. Bianchi, N. Blefari-Melazzi, G. Bonafede, and E. Tintinelli. QUASIMODO: Quality of service-aware multicasting over DiffServ and overlay networks. *IEEE Network*, 17(1):38–45, Jan. 2003.

[5] S. Biswas, R. Izmailov, and B. Rajagopalan. A QoS-aware routing framework for PIM-SM based IP-multicast. *Internet draft: draft-biswas-pim-sm-qos-00.txt*, June 1999.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for Differentiated Services. *IETF RFC 2475*, Dec. 1998.

[7] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens. Explicit multicast (Xcast) basic specification. *Internet draft: draft-ooms-xcast-basic-spec-01.txt*, Mar. 2001.

[8] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet architecture: an overview. *IETF RFC 1633*, 1994.

[9] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pages 5–11, Jan. 1997.

[10] S. Chen, K. Nahrstedt, and Y. Shavitt. A qos-aware multicast routing protocol. *Proceedings of IEEE INFOCOM*, Mar. 2000.

[11] J.-F. Chiu, Z.-P. Huang, C.-W. Lo, W.-S. Hwang, and C.-K. Shieh. Supporting end-to-end QoS in DiffServ/MPLS networks. In *Proceedings of International Conference on Telecommunications*, pages 261–266, Feb. 2003.

[12] L. H. M. Costa, S. Fdida, and O. C. M. Duarte. Hop-by-hop multicast routing protocol. *Proceddings of SIGCOMM'01*, Aug. 2001.

[13] J.-H. Cui, M. Faloutsos, and M. Gerla. An architecture for scalable, efficient and fast fault-tolerant multicast provisioning. *IEEE Network special issue on Protecction, Restoration, and Disaster Recovery*, 18(2):26–34, Mar. 2004.

[14] J.-H. Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta. Measuring and modelling the group membership in the Internet. In *Proceedings of the ACMSIGCOMM/USENIX Internet Measurement Conference (IMC 2003)*, Oct. 2003.

[15] A. I. Elwalid and D. Mitra. Effective bandwidth of general Markovian traffic sources and admission control of high speed networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, June 1993.

[16] D. Estrin, M. Handley, A. Helmy, P. Huang, and D. Thaler. A dynamic bootstrap mechanism for rendezvous-based multicast routing. *Proceddings of IEEE INFOCOM'99*, 1999.

[17] F. L. F. et al. Protocol extensions for support of Differentiated-Service-aware MPLS traffic engineering. *Internet draft: draft-ietf-tewg-diff-te-proto-07.txt*, Mar. 2004.

[18] M. Faloutsos, A. Banerjea, and R. Pankaj. QoSMIC: Quality of Service sensitive Multicast Internet protoCol. *ACM SIGCOMM'98, Vancouver, British Columbia*, Sept. 1998.

[19] F. L. Faucheur and W. Lai. Requirements for support of Differentiated Services-aware MPLS traffic engineering. *IETF RFC 3564*, July 2003.

[20] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) support of Differentiated Services. *IETF RFC 3270*, May 2002.

[21] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium(GI2001)*, Nov. 2001.

[22] A. Fei and M. Gerla. Receiver-initiated multicasting with multiple QoS constraints. *Proceedings of IEEE INFOCOM*, Mar. 2000.

[23] S. Ganti, N. Seddigh, and B. Nandy. MPLS support of Differentiated Services using E-LSP. *Internet draft: draft-ietf-mpls-diff-ext-00.txt*, Apr. 2001.

[24] R. N. George and B. Ilia. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selection Areas in Communications*, 15(3):346–356, Apr. 1997.

[25] J. Hou, H.-Y. Tyan, B. Wang, and Y.-M. Chen. QoS extension to CBT. *Internet draft: draft-hou-cbt-qos-00.txt*, Feb. 1999.

[26] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *Proceedings of ACM SIGCOMM'95*, Sept. 1995.

[27] Z. Li and P. Mohapatra. Qos-aware multicast protocol using bounded flooding (QMBF) technique. In *Proceedings of IEEE International Conference on Communications (ICC)*, Apr. 2002.

[28] Z. Li and P. Mohapatra. QoS-aware multicasting in DiffServ domains. In *Proceedings of IEEE Global Internet Symposium*, Nov. 2002.

[29] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *In Proceedings of ACM SIGCOMM'96*, pages 117–130, 1996.

[30] M. Moh, B. Wei, and J. H. Zhu. Supporting differentiated services with per-class traffic engineering in MPLS. In *Proceedings of International Conference on Computer Communications and Networks*, pages 354–360, Oct. 2001.

[31] J. Moy. Multicast routing extensions to OSPF. *IETF RFC 1584*, Mar. 1994.

[32] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16(2):387–396, 1994.

[33] D. Ooms, R. Hoebeke, P. Cheval, and L. Wu. MPLS multicast traffic engineering. *Internet draft: draft-ooms-mpls-multicast-te-00.txt*, 2001.

[34] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul, and F. Ansari. Overview of IP multicast in a multi-protocol label switching (MPLS) environment. *IETF RFC 3353*, 2004.

[35] E. Pagani and G. P. Rossi. Measurement-based admission control for dynamic multicast groups in Diff-Serv networks. In *Proceedings of NETWORKING*, May 2002.

[36] E. Pagani, G. P. Rossi, and D. Maggiorini. A multicast transport service with bandwidth guarantees for Diff-Serv networks. In *Proceedings of International Workshop on QoS in MultiService IP Networks (QoS-IP)*, Jan. 2001.

[37] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green. Simple Multicast: A design for simple, low-overhead multicast. *Internet draft: draft-perlman-simple-multicast-03.txt*, Oct. 1999.

[38] P. I. Radoslavov, D. Estrin, and R. Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical report, USC Dept. of CS 99-697 (Second Revision), July 1999.

[39] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching architecture. *IETF RFC 3031*, 2001.

[40] N. Rouhana and E. Horlait. Differentiated Services and Integrated Services use of MPLS. In *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, pages 194–199, July 2000.

[41] H. Saito, Y. Miyao, and M. Yoshida. Traffic engineering using multiple multipoint-to-point LSPs. In *Proceedings of IEEE INFOCOM*, pages 894–901, Mar. 2000.

[42] K. Sanjiv and R. Srivatsan. Improved multicast routing with delay and delay variation constraint. *Global Telecommunications Conference (GLOBECOM'00)*, 1:476–480, 2000.

[43] P.-R. Sheu and S.-T. Chen. A fast and efficient heuristic algorithm for the delay and delay variation bound multicast tree problem. *In Proceedings of the 15th International Conference on Information Networking (ICOIN'01)*, pages 611–618, Feb. 2001.

[44] I. Stoica, T. Ng, and H. Zhang. REUNITE: A recursive unicast approach to multicast. In *Proceedings of IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000.

[45] A. Striegel, A. Bouabdallah, H. Bettahar, and G. Manimaran. EBM: a new approach for scalable DiffServ multicasting. In *Proceedings of International Workshop on Networked Group Communication (NGC)*, Sept. 2003.

[46] A. Striegel and G. Manimaran. Dynamic DSCPs for heterogeneous QoS in DiffServ multicasting. In *Proceedings of IEEE GLOBECOM*, Nov. 2002.

[47] A. Striegel and G. Manimaran. A survey of QoS multicasting issues. *IEEE Communications Magazine*, 40(6):82–87, June 2002.

[48] A. Striegel and G. Manimaran. DSMCast: a scalable approach for DiffServ multicasting. *Computer Networks*, 44(6):713–735, Apr. 2004.

[49] D. Thaler and M. Handley. On the aggregatability of multicast forwarding state. *Proceedings of IEEE INFOCOM*, Mar. 2000.

[50] J. Tian and G. Neufeld. Forwarding state reduction for sparse mode multicast communications. *Proceedings of IEEE INFOCOM*, Mar. 1998.

[51] P. Trimintzios, I. Andrikopoulos, G. Pavlou, P. Flegkas, D. Griffin, P. Georgatsos, D. Goderis, Y. TJoens, L. Georgiadis, C. Jacquenet, and R. Egan. A management and control architecture for providing IP Differentiated Services in MPLS-based networks. *IEEE Communications Magazine*, 39(5):80–88, May 2001.

[52] B. Wang and J. C. Hou. Multicast routing and its QoS extension: Problems, algorithms, and protocols. *IEEE Network*, 14(1):22–36, Jan. 2000.

[53] B. Yang and P. Mohapatra. Edge router multicasting with MPLS traffic engineering. In *Proceedings of IEEE International Conference on Networks (ICON)*, Aug. 2002.

[54] B. Yang and P. Mohapatra. Multicasting in differentiated service domains. In *Proceedings of IEEE GLOBECOM*, Nov. 2002.