Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks

Jun-Hong Cui[1], Jinkyu Kim[1], Aiguo Fei[1], Michalis Faloutsos[2], Mario Gerla[1]
[1] Computer Science Department, University of California, Los Angeles, CA 90095
[2] Computer Science & Engineering, University of California, Riverside, CA 92521

Abstract— **IP multicast suffers from scalability problems as the number of concurrent active multicast groups increases, since it requires a router to keep a forwarding state for every multicast tree passing through it. In QoS multicast provisioning, the problem is exacerbated, since not only the forwarding state but also the resource requirement of a multicast group must be kept at the router. To provide scalable QoS multicast support, in this paper, we propose a novel architecture, called Aggregated QoS Multicast (AQoSM). Using the concept of aggregated multicast [7], AQoSM can support QoS multicast scalably and efficiently in Diff-Serv-Supported MPLS networks. In this paper, we develop the framework for the architecture and provide a feasibility check from an implementation point of view. The architecture is flexible and can be customized to the needs and the existing protocols of a domain. Our simulations indicate that the architecture performs well in several common scenarios. It achieves smaller blocking of users with strong QoS requirements because of its load balancing capability. It also achieves up to 85% reduction in state with a modest 10% of bandwidth overhead.**

I. Introduction

Today, many non real time applications such as news, software distribution, etc., can be effectively supported by some alternate techniques (to network level multicasting) such as web caching and application level multicast. Real time (but non-interactive) applications such as video on demand can take advantage of the same alternate techniques (e.g. web caching). In contrast, if we consider true interactive, real time applications such as video conferencing, distributed network games, distributed virtual collaborations (with real time visualization and remote experiment steering), distance lectures with student participation, we realize that alternate techniques such as web caching would severely affect time responsiveness. As a result, it is important to provide efficient QoS multicast support, since it will be a prominent offering in the gamut of future Internet services.

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 QoSMIC [6], QMRP [5], RIMQoS [9], QoS extension to CBT [10], and PIM-SM QoS extension [2]. But all these schemes use per-flow state. Today people are backing away from micro-flow based QoS architecture, namely the Integrated Services architecture (IntServ) [4]. The reason behind this choice is simple: requiring per-flow reservation and data packet handling, Integrated Services architecture has scalability problems at network core routers. Instead of this, the recent trend is to use aggregated flow based solutions, namely, the Differentiated Services architecture (Diff-Serv) [3] and the Multiple Protocol Label Switching (MPLS) technology [13]. 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 separate states for individual multicast groups which pass though it.

To provide scalable and efficient QoS multicast support, in this paper, we propose a novel architecture, called Aggregated QoS Multicast (AQoSM), which is designed based on the aggregated multicast scheme developed in [7]. Using the concept of aggregated multicast, AQoSM can support QoS multicast scalably and efficiently in Diff-Serv-Supported MPLS networks. QoS multicast provisioning is a multifaceted problem, involving routing, admission control, resource management and many other issues. In this paper, we provide efficient and practical solutions for these critical issues. Our analysis and simulation study will show that AQoSM is efficient, scalable, feasible and implementable based on MPLS and Diff-Serv techniques.

The rest of this paper is organized as follows. Section II gives a short review of aggregated multicast. Section III presents the AQoSM architecture in detail. Then Section IV evaluates the performance of AQoSM through simulations. Finally Section V offers an overall summary of our contributions.

This material is based upon work supported by the National Science Foundation under Grant No. 9805436 and CAREER Grant No. 9985195, and CISCO/CORE fund No. 99-10060.

II. Aggregated Multicast

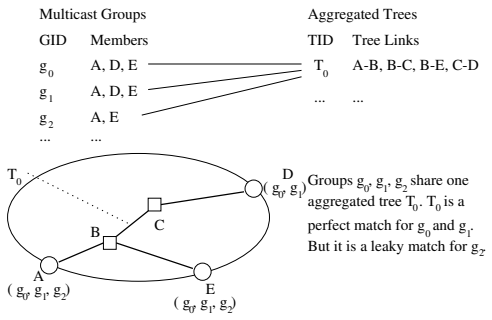

Fig. 1. Illustration of aggregated multicast

Aggregated multicast [7] is a scheme proposed to reduce multicast state. The key idea is to force multicast groups to share a single distribution tree. This enforcement takes place at the border routers of the network. Data packets from different groups are multiplexed on the same distribution tree, called **aggregated tree**. Each data packet of each group is encapsulated and travels on the aggregated tree. This way, routers in the middle of the network, namely core routers, need to keep state only per aggregated tree, which are much less in number than the groups they are servicing. Of course, border routers at the boundaries of the network need to maintain sufficient information to multiplex and demultiplex groups in and from aggregated trees. Note that the focus of the work was on reducing multicast state without discussing explicitly the support of QoS. Fig. 1 illustrates the basic idea of aggregated multicast.

In aggregated multicast, we need to match groups to aggregated trees. The group-tree matching problem hides several subtleties. The set of the group members and the tree leaves are not always identical. A match is a **perfect** for a group, if all the tree leaves have group members. A match may also be a **leaky match**, if there are leaves of the tree that do not have group members. In other words, we send data to parts of the tree that is not

wanted by anyone. A disadvantage of the leaky match is that some bandwidth is wasted to deliver data to nodes that are not members for the group. Namely, we trade off bandwidth for state reduction.

III. AQoSM—The New Architecture for Scalable QoS Multicast Provisioning

We design a new architecture, AQoSM (Aggregated QoS multicast), to support scalable QoS multicast in Diff-Serv-Supported MPLS networks. 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 is targeted at QoS multicast provisioning in a single domain, particularly backbone domains. The domain we discuss in this paper is an MPLS domain which supports Differentiated Services; in other words, it is a Diff-Serv-Supported MPLS domain.

In a nutshell, AQoSM maintains MPLS-trees that serve multiple groups in a Diff-Serv-Supported MPLS domain. 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, e.g. for QoS reasons.

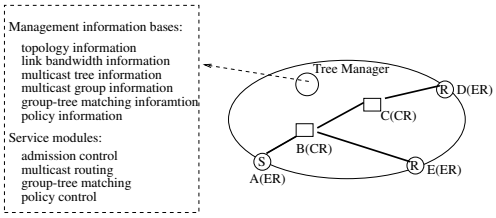

Fig. 2. Illustration of a tree manager in a Diff-Serv-Aware MPLS domain.

We introduce a logical entity called *tree manager*, which is illustrated in Fig. 2, where A, D, and E are edge routers, and B and C are core routers. The tree manager for multicast functions like the Bandwidth Broker for unicast [11]. The tree mananger needs to have information of: the network topology, the available resources, the group membership, and group QoS requirements. The tree manager can be implemented in centralized or distributed ways. For simplicity of presentation, we can think of it as a single node.

The tree manager is responsible for maintaining trees and matching groups to trees. It consists of several service modules, such as admission control, group-tree matching, routing and policy control. The routing module peers with routers to obtain the topology information of the network domain, and is responsible for establishing new trees and detaching obsolete, idle trees. The group-tree matching module needs to keep the information of active groups and established trees and the group-tree matching table, taking the task of matching incoming multicast groups to proper (existing or new) trees. The admission control module maintains link residual bandwidth, and is responsible for admission control. The policy control module preserves a policy information base and helps to do a network policy administration. This paper will mainly focus on the routing, group-tree matching and admission control modules for AQoSM.

Before going into the detailed design issues, we give a "big picture" of AQoSM. After collecting membership and QoS requirement of multicast groups, link state information (or topology information), available bandwidth of links, the tree manager has up-to-date information about the entire network domain and about all the multicast groups. 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 and tries to find a match with an established tree. If no such tree exists, the tree manager computes a new multicast tree according to membership and QoS requirements through 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 MPLS tree is established through a Label Distribution Protocol (LDP). 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. A big picture of AQoSM is shown in Fig. 3.

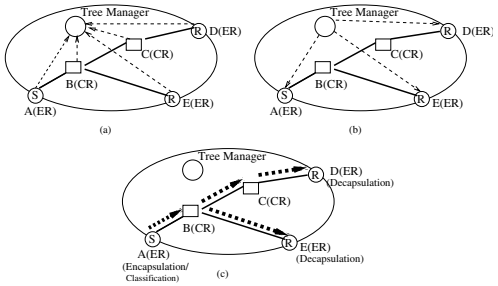

Fig. 3. A big picture of Aggregated QoS Multicast: (a) Membership, QoS requirement, link state, and available bandwidth collection; (b) Group-tree matching entry distribution; (c) Multicast group packets transmitting on established MPLS aggregated multicast tree.

From its brief overview, we can see AQoSM involves many design issues: link state collection, group membership collection, admission control, multicast routing, QoS-aware group-tree matching, MPLS tree management, etc. Some of the issues, such as link state collection, group membership collection, and admission control are not unique to AQoSM, and existing technologies can be applied. For example, link state collection can be done based on the unicast routing approach, and group membership can be sent directly to the tree manager or be piggybacked on link-state packets if unicast routing uses a link state approach. For admission control, either parameter-based or measurement-based approach can be used, while the latter one is a better choice for Differentiated Services since it is probabilistic in nature, and it cannot provide tight guaranteed resource. The remainder of this section mainly describes detailed solutions for the new issues involved in AQoSM.

A. Multicast Routing

When a new group comes, if the tree manager can not find a proper existing tree, it then needs to compute a new tree for the group through the multicast routing module. AQoSM adopts a PIM-SM/CBT like routing algorithm to compute a bi-directional tree for a group. The corresponding RP or core node can be properly chosen to achieve load balancing. Note that we use bi-directional trees instead of unidirectional trees in AQoSM. The main advantage is that, whenever a tree covers the members of a group, it can be used for packet delivering for the group, without

being checked for transmission direction (which is necessary for unidirectional trees). In this way, more groups can share a single tree, which means more state reduction and fewer aggregated trees. However, AQoSM does not exclude using unidirectional trees, and other QoS multicasting protocols, such as QoSMIC [6], QMRP [5], and RIMQoS [9] can also be applied.

B. QoS-Aware Group-Tree Matching Algorithm

To match a group to a tree, the tree manager needs to maintain tables for established multicast trees, active multicast groups, and group-tree matching entries. Before stepping into the group-tree matching algorithm, we introduce some notations and definitions.

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

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

If every link is assumed to have equal cost 1, tree cost is simply $C(T) = |T| - 1$, where $|T|$ denotes the number of nodes in T. This assumption holds in this paper. Let MTS (Multicast Tree Set) denote the current set of multicast trees established in the network (or all the trees in the tree table). A "native" multicast tree (e.g. using PIM-SM/CBT like core based tree routing algorithm, denoted by **A**) which satisfies the membership and QoS requirement of a multicast group g is denoted by $T_A(g)$, while $T(g)$ defines the aggregated tree which g uses to transmit data.

As mentioned in Section II, it is possible that $T(g)$ does not have a perfect match with group g, which means that some of the leaf nodes of $T(g)$ are not the member nodes of g, and then packets reach some destinations that are not interested in receiving them. Thus, there is bandwidth overhead. Assume an aggregated tree T_0 is used by groups g_i, $1 \le i \le n$, each of which has a "native" tree $T_A(g_i)$, then the average percentage bandwidth overhead for T_0 can be defined as

$$\begin{aligned}\delta_A(T_0) &= \frac{\sum_{i=1}^{n} B(g_i) \times (C(T_0) - C(T_A(g_i)))}{\sum_{i=1}^{n} B(g_i) \times C(T_A(g_i))} \\ &= \frac{C(T_0) \times \sum_{i=1}^{n} B(g_i)}{\sum_{i=1}^{n} B(g_i) \times C(T_A(g_i))} - 1,\end{aligned} \tag{2}$$

where $B(g)$ is the bandwidth requirement of group g.

When the tree manager detects a new multicast group g, it populates the corresponding entries of multicast group table and does the following QoS-Aware group-tree matching algorithm (Let b_t be the given bandwidth overhead threshold):
(1) Compute a "native" multicast tree $T_A(g)$ for g based on the multicast group membership, bandwidth requirement and available bandwidth of links. Note that the routing module will choose a good RP or core node from all the candidates so that enough bandwidth is available on the links. If this kind of candidate does not exist or not enough allocated bandwidth is available for g's service class, the multicast group g may be rejected;
(2) For each tree T in MTS, if T covers g and enough bandwidth is available on the tree and g's service class, compute $\delta_A(T)$. If $\delta_A(T) < b_t$, then T is considered as a candidate to cover g;
(3) Among all candidates, choose the one such that $C(T)$ is minimum and denote it as T_m; T_m is used to cover g. Update multicast group table and group-tree matching table;
(4) If no candidate found in step (2), $T_A(g)$ is used to cover g and is added to MTS and the corresponding tables are updated. Of course, if no $T_A(g)$ computed in step (1), this multicast group is denied.

In addition, whenever 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.

C. MPLS Tree Management

After a new multicast tree is computed, its corresponding MPLS tree needs to be established. Note that AQoSM employs bi-directional trees. Although there exist solutions to distribute labels for unidirectional multicast trees [12], no research work has been found for bi-directional trees' label distribution in the literature.

We have got two kinds of solutions for bi-directional MPLS tree setup: one is centralized, and the other is distributed. In the centralized solution, the tree manager generates all the MPLS labels for the bi-directional tree and distributes them to the corresponding routers directly. Then the routers will create label forwarding entries for the tree. An alternative is the distributed approach. This approach extends the existing unidirectional MPLS tree setup schemes [12]: root-initiated or leaf-initiated. The idea is very simple: a bi-directional tree can be viewed as a combination of n unidirectional trees, where n is the number of the leaf routers in the bi-directional tree. Each unidirectional tree has a leaf router of the bi-directional tree as its "root". Since the whole bi-directional tree is available, it is not difficult to create unidirectional tree objects. 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 scheme. In this method, we use the root-initiated scheme. Similarly, we can apply the leaf-initiated scheme also. More details about the root-initiated and leaf-initiated unidirectional MPLS tree setup schemes can be found in [12].

When an MPLS tree becomes idle, the tree manager might need to destroy the MPLS tree and delete the corresponding entry in the tree table. Depending on what kind of approach is used for MPLS tree setup, the tree manager sends label withdraw messages to all the in-tree routers of the aggregated multicast tree if the centralized approach is employed; or, if we adopt the distributed approach, the tree manager only notifies the leaf routers of the bi-directional multicast tree, and each leaf router sends label withdraw message to its upstream Label Switch Routers.

D. Summary

AQoSM employs aggregated multicast for QoS multicast provisioning in Diff-Serv-Supported MPLS domains. The simple idea of separating the concept of groups from the concept of tree distribution opens a world of new possibilities. First, groups can now be routed and rerouted very quickly. We just need to "label" the packets differently. The implications are astounding. We can have load-balancing and dynamic rerouting to meet QoS requirements. Second, the aggregation of groups on few trees leads to several other advantages. It provides routing state reduction and efficient resource utilization through statistical multiplexing.

The scalability of our architecture stems from the following reasons. First, we need to maintain fewer trees. Second, the routing state at core routers is reduced significantly, thus memory resource are saved and forwarding processes are facilitated greatly.

Third, QoS routing decisions are pushed to the boundaries of the network. This is in agreement with the "Diff-Serv mentality", where we want to keep the core simple and fast, while put as much intelligence as possible to the boundaries of the network.

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

IV. Performance Evaluation

In this section, we provide simulation results to evaluate the performance of AQoSM, specially on the aspects of scalability and load balancing.

A. Performance Metrics

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

Number of MPLS Trees is the average number of MPLS trees maintained in the tree manager. This metric is a direct measurement for the multicast tree maintenance overhead. The more multicast trees, the more memory required and the more processing 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 it is, the higher MPLS tree setup rate.

B. Results and Analysis

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.

To generate multicast groups more realistically, in our simulation, we use one of the group models developed in [8]: the random node-weighted model. In this model, each node is assigned a weight, which is the probability of the node to participate in multicast sessions. In the target network, core routers will not be members for any multicast group and thus are assigned weight 0. Any other edge router is assigned a weight 0.2 or 0.8 according to the real-time traffic of its corresponding core router. The rationale behind this is that, for a router, more traffic means more participation in the network communication, thus it has higher probability 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 group request rejection ratio.

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. Performance data is collected at steady state(e.g. after $T = 10\mu$).

We design experiments to compare AQoSM 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 AQoSM and native PIM-SM/CBT is shown in Table I. In our experiments, AQoSM 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 AQoSM, the algorithm specified in Section III-B is used to match a group to a tree. The corresponding routing algorithm A is PIM-SM/CBT like routing algorithm which is also used for native PIM-SM/CBT. In both AQoSM and native PIM-SM/CBT, if the tree computed based on the original core cannot 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 no enough bandwidth is available. In this way, better load balancing will be achieved.

In our experiments, we vary the bandwidth overhead threshold (represented as **bth**) from 0 to 0.3 for AQoSM. Fig. 4 shows the results for **Number of MPLS Trees** vs the number of concurrent active groups. We can see that AQoSM "scales" with the average number of concurrent groups: for native PIM-SM/CBT, the number of MPLS trees grows almost linearly with the number of groups; for AQoSM, as the number of groups becomes bigger, the number of trees also increases, but the increase is much less than that of native PIM-SM/CBT (even for perfect match ($bth = 0$), the number of trees is only 880 instead of 3500 when there are 3500 groups), which means much less tree maintenance overhead involved in the tree manager. Moreover, the "increase" decreases as there are more groups, which means that as more groups are pumped into the network, more groups can share a single MPLS tree.

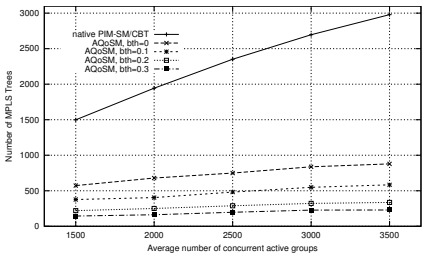

Fig. 4. Number of MPLS Trees vs number of active groups.

Fig. 5 plots the change of **Number of Label Forwarding Entries** with the number of concurrent active groups. It has a similar trend to the metric **Number of MPLS Trees**. The number of label forwarding entries is reduced from 118900 to 31600 (above 75% reduction) even for perfect match when 3500 groups come. Thus, we can conclude that, in AQoSM, the label maintenance and forwarding process overhead are significantly reduced.

In Fig. 6, we demonstrate the effect of the number of concur-

TABLE I
A HIGH LEVEL COMPARISON OF SIMULATED AQOSM AND NATIVE QOS-AWARE PIM-SM/CBT MPLS MULTICAST (NATIVE PIM-SM/CBT)

Name	*Multicast Routing*	*Tree Type*	*MPLS Tree?*	*Group-Tree Matching?*	*QoS-aware?*
AQoSM	PIM-SM/CBT like routing	Bi-directional	Yes	Yes	Yes
Native PIM-SM/CBT	PIM-SM/CBT like routing	Bi-directional	Yes	No	Yes

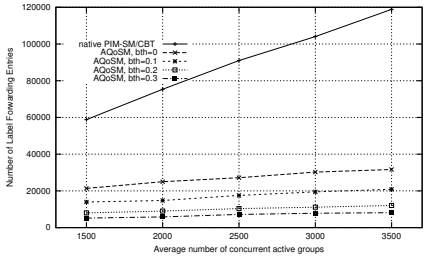

Fig. 5. Number of Label Forwarding Entries vs number of active groups.

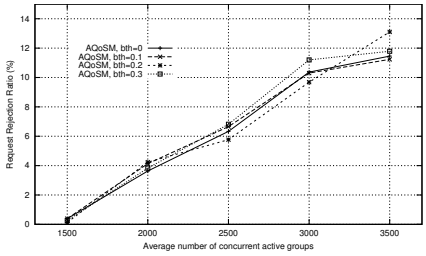

Fig. 7. Request Rejection Ratio with different bandwidth overhead thresholds.

rent active groups on **Tree 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 group share a single MPLS tree when the number of groups is bigger, and thus less trees need to set up. In addition, the tree setup ratio is much smaller in AQoSM compared with native PIM-SM/CBT, which means the tree setup overhead is dramatically reduced.

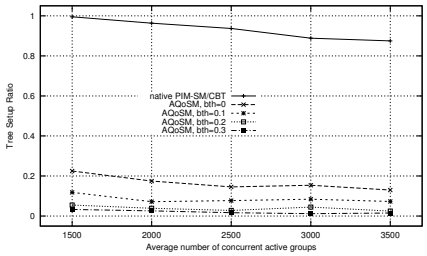

Fig. 6. Tree Setup Ratio vs number of active groups.

From Fig. 4, Fig. 5, and Fig. 6, 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 less tree and label management overhead. Therefore, there is a trade-off between management overhead reduction and bandwidth waste. The balance depends on the network administration policy.

Fig. 7 investigates how the aggregation affects **Request Rejection Ratio**. The figure shows that the request rejection ratio is not influenced by the aggregation even under leaky match cases. Apparently, leaky match causes some bandwidth waste, thus it should have some effects on admission control: the more bandwidth waste, the bigger request rejection ratio. However, in AQoSM, though some links are congested, it is still possible for a group to find a good tree since the RP of the group can be dynamically changed. In other words, we achieve load balancing in AQoSM.

V. CONCLUSIONS

In this paper, we propose and develop a multicast architecture to support QoS scalably and efficiently in Diff-Serv-Supported MPLS networks. 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 and adaptability to changing conditions. In addition, our architecture reduces the multicast state at core routers.

We conduct simulations to quantify the claims of the architecture. We compare our scheme with a native 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.
- The overhead of setting up and maintaining a tree is better amortized as the number of groups increases.
- Our architecture can accommodate more users than with a traditional multicast. The advantage comes from the ability to explore many trees for a given group, which is not supported in classic multicast protocols.

REFERENCES

[1] Abilene network topology. *http://www.ucaid.edu/abilene/*.
[2] 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.
[3] S. Blake, D. Black, and et al. An architecture for Differentiated Services. *IETF RFC 2475*, 1998.
[4] R. Braden, D. Clark, and S. Shenker. Integrated Services in the internet architecture: an overview. *IETF RFC 1633*, 1994.
[5] S. Chen, K. Nahrstedt, and Y. Shavitt. A qos-aware multicast routing protocol. *Proceedings of IEEE INFOCOM*, Mar. 2000.
[6] M. Faloutsos, A. Banerjea, and R. Pankaj. QoSMIC: Quality of Service sensitive Multicast Internet protoCol. *ACM SIGCOMM'98, Vancouver, British Columbia*, Sept. 1998.
[7] 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.
[8] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast with inter-group tree sharing. *Proceedings of NGC2001*, Nov. 2001.
[9] A. Fei and M. Gerla. Receiver-initiated multicasting with multiple QoS constraints. *Proceedings of IEEE INFOCOM*, Mar. 2000.
[10] 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.
[11] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. *RFC 2638*, July 1999.
[12] D. Ooms, R. Hoebeke, P. Cheval, and L. Wu. MPLS multicast traffic engineering. *Internet draft: draft-ooms-mpls-multicast-te-00.txt*, 2001.
[13] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching architecture. *IETF RFC 3031*, 2001.