

# Aggregated Multicast for Scalable QoS Multicast Provisioning

Mario Gerla<sup>1</sup>, Aiguo Fei<sup>1</sup>, Jun-Hong Cui<sup>1</sup>, and Michalis Faloutsos<sup>2</sup>

<sup>1</sup> Computer Science Department, University of California, Los Angeles, CA90095

<sup>2</sup> Computer Science and Engineering, University of California, Riverside, CA 92521

**Abstract.** IP multicast suffers from scalability problem with the number of concurrently active multicast groups, while scalability of QoS multicast is even further from being solved. In this paper, we propose an approach to reduce multicast forwarding state and provision multicast with QoS guarantees. In our approach, multiple groups are forced to share a single delivery tree. We discuss the advantages and some implementation issues of our approach, and conclude that it is feasible and promising. We then describe how to use our approach to provision scalable QoS multicast. Finally, we define metrics to quantify state reduction and use simulations to show how our scheme achieves state reduction. These initial simulation results suggest that our method can reduce multicast state significantly.

## 1 Introduction

Multicast state scalability is the problem we address in this work. Multicast is a mechanism to efficiently support multi-point communications. IP multicast utilizes a tree delivery structure, on which data packets are duplicated only at fork nodes and are forwarded only once over each link. This approach makes IP multicast resource-efficient in delivering data to a group of members simultaneously and can scale well to support very large multicast groups. However, even after approximately 20 years of multicast research and engineering effort, IP multicast is still far from being as common-place as the Internet itself.

Multicast state scalability is among the technical difficulties that delay its deployment. A multicast distribution tree requires all tree nodes to maintain per-group (or even per-group/source) forwarding state, which grows at least linearly with the number of “passing-by” groups. As multicast gains widespread use and the number of concurrently active groups grows, more and more forwarding state entries will be needed. More forwarding entries translates into more memory requirement, and may also lead to slower forwarding process since every packet forwarding involves an address look-up. In QoS multicast, the problem becomes even worse, because not only routes but also resources (eg, bandwidth) for individual multicast group are needed to maintain. This perhaps is the main scalability problem with IP multicast and QoS multicast provisioning when the number of simultaneous on-going multicast sessions is very large.

Recently, much research effort has focused on the problem of multicast state scalability. Some schemes attempt to reduce forwarding state by tunneling[14] or by forwarding state aggregation[10, 13]. Thaler and Handley analyze the aggregatability of

forwarding state in [13] using an input/output filter model of multicast forwarding. Radoslavov et al. propose algorithms to aggregate forwarding state and study the bandwidth-memory tradeoff with simulation in [10]. Both these works attempt to aggregate routing state after this has been allocated to groups. Second, some other architectures aim to completely eliminate multicast state at routers [6, 11] using network-transparent multicast, which pushes the complexity to the end-points.

Though most research papers on QoS multicast are focusing on solving a theoretical constrained multicast routing problem, there have been efforts to bring QoS into existing IP multicast architecture, such as RSVP [15], QoS MIC [2], QoS extension to CBT [7], and PIM-SM QoS extension [3]. But all these schemes are using per-flow state, keeping track of routes and resources information for each individual group, which suffers scalability problem as mentioned above.

In this paper, we propose a novel scheme to reduce multicast state and provision scalable QoS multicast, which we call aggregated multicast. Our difference with previous approaches is that we force multiple multicast groups to share one distribution tree, which we call an *aggregated tree*. This way the total number of trees in the network may be significantly reduced and thus forwarding state: core routers only need to keep state per aggregated tree instead of per group. In this paper we examine several design and implementation issues of our scheme and describe how to use aggregated multicast scheme to provision multicast with QoS guarantees. We will also present results from our initial simulation experiments in which our scheme achieves significant state reduction in the worst case scenario where group members have no spatial locality at all.

The rest of this paper is organized as follows. Section 2 introduces the concept of aggregated multicast approach and discusses some implementation related issues. Section 3 talks about QoS provisioning on the aggregated tree. Section 4 proposes metrics to quantify multicast state reduction in aggregated multicast and presents simulation results. Section 5 gives a short summary of our work.

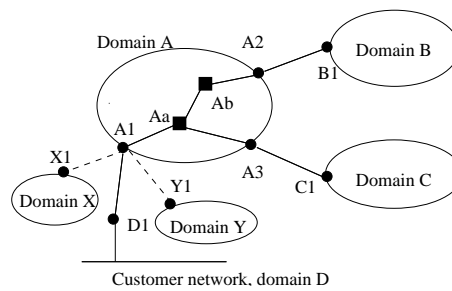
## 2 Aggregated Multicast

Aggregated multicast is targeted as an intra-domain multicast provisioning mechanism in the transport network. For example, it can be used by an ISP (Internet Service Provider) to provide multi-point data delivery service for its customers and peering neighbors in its wide-area or regional backbone network (which can be just a single domain). The key idea of aggregated multicast is that, instead of constructing a tree for each individual multicast session in the core network (backbone), one can have multiple multicast sessions share a single aggregated tree to reduce multicast state and, correspondingly, tree maintenance overhead at network core.

### 2.1 Concept

Fig. 1 illustrates a hierarchical inter-domain network peering. Domain A is a regional or national ISP's backbone network, and domain D, X, and Y are customer networks of domain A at a certain location (say, Los Angeles). Domain B and C can be other

customer networks (say, in New York) or some other ISP's networks that peer with A. A multicast session originates at domain D and has members in domain B and C. Routers D1, A1, A2, A3, B1 and C1 form the multicast tree at the inter-domain level while A1, A2, A3, Aa and Ab form an intra-domain sub-tree within domain A (there may be other routers involved in domain B and C). The sub-tree can be a PIM-SM shared tree rooted at an RP (Rendezvous Point) router (say, Aa) or a bi-directional shared CBT (Center-Based Tree) tree centered at Aa or maybe an MOSPF tree. Here we will not go into intra-domain multicast routing protocol details, and just assume that the traffic injected into router A1 by router D1 will be distributed over that intra-domain tree and reaches router A2 and A3.



**Fig. 1.** Domain peering and a cross-domain multicast tree, tree nodes: D1, A1, Aa, Ab, A2, B1, A3, C1, covering group  $G_0(D1, B1, C1)$ .

Consider a second multicast session that originates at domain D and also has members in domain B and C. For this session, a sub-tree with exactly the same set of nodes will be established to carry its traffic within domain A. Now if there is a third multicast session that originates at domain X and it also has members in domain B and C, then router X1 instead of D1 will be involved, but the sub-tree within domain A still involves the same set of nodes: A1, A2, A3, Aa, and Ab. To facilitate our discussions, we make some distinctions among these nodes. We call node A1 a **source node** at which external traffic is injected, and node A2 and A3 **exit nodes** which distribute multicast traffic to other networks, and node Aa and Ab **transit nodes** which transport traffic in between. In a bi-directional inter-domain multicast tree, a node can be both a source node and an exit node. Source nodes and exit nodes together are called **terminal nodes**. Using the terminologies commonly used in DiffServ[4], terminal nodes are often *edge* routers and transit nodes are often *core* routers in a network.

In conventional IP multicast, all the nodes in the above example that are involved within domain A must maintain separate state for each of the three groups individually though their multicast trees are actually of the same “shape”. Alternatively, in an aggregated multicast approach, one can setup a pre-defined tree(or establish on demand) that covers nodes A1, A2 and A3 using a single multicast group address (within domain A). This tree is called an **aggregated tree (AT)** and it is shared by all multicast groups that are covered by it and are assigned to it. We say an aggregated tree  $T$  covers

a group  $G$  if all terminal nodes for  $G$  are member nodes of  $T$ . Data from a specific group is encapsulated at the source node. It is then distributed over the aggregated tree and decapsulated at exist nodes to be further distributed to neighboring networks. This way, transit router Aa and Ab only need to maintain a single forwarding entry for the aggregated tree regardless how many groups are sharing it.

## 2.2 Implementation Considerations

It is not our goal to provide protocol details for aggregated multicast in this paper. However, a high-level overview of how it can be implemented in practice will provide a reality check which helps validate our work and provides some insights regarding its advantages and drawbacks.

First of all, there are various options for distributing multicast traffic of different groups over a shared aggregated tree. Regardless of the implementation, there are two basic requirements: (1)the original group address of data packets must be stored somewhere and can be recovered by exit nodes to determine how to forward these packets in the access network, and; (2)some kind of identification for the aggregated tree which the group is using must be carried in the packet header and is used by transit nodes to forward the packet. One possibility is to use IP encapsulation as said above, which, of course, adds complexity and processing overhead (at terminal nodes). A more efficient solution is MPLS (Multiprotocol Label Switching)[12] in which labels can identify different aggregated trees.

To handle aggregated tree management and matching between multicast groups and aggregated trees, a centralized management entity called tree manager is introduced. A tree manager has the knowledge of established aggregated trees in the network and is responsible for establishing new ones when necessary. It collects (inter-domain) group join messages received by border routers and assigns aggregated trees to groups. Once it determines which aggregated tree to use for a group, the tree manager can install corresponding state at the edge nodes involved, or distribute corresponding label bindings if MPLS is used. Aggregated tree construction within the domain can use an existing routing protocol such as PIM-SM, or use a centralized approach like what proposed in centralized multicast[8], or use MPLS signaling protocols extensions proposed in[9] to support the establishment of pre-calculated trees.

The set of aggregated trees to be established can be determined based on traffic pattern from long-term measurements. Let us say, for example, measurements in MCI-Worldcom's national backbone show that there are always many concurrent multicast sessions that involve three routers in Los Angeles, San Francisco and New York. Based on that knowledge, a network operator can instruct the tree manager to setup an aggregated tree covering routers in these three locations. Aggregated trees can also be established, changed (to add/remove nodes) or removed dynamically based on dynamic traffic monitoring. Knowing a set of existing aggregated trees, a tree manager can "match" a specific group, with given group membership (set of terminal nodes), to an aggregated tree that covers the group (i.e., all terminal nodes are member nodes of the tree).

### 2.3 Discussions

A number of benefits of aggregation are apparent. First of all, transit nodes don't need to maintain state for individual groups; instead, they only maintain forwarding state for a potentially much smaller number of aggregated trees. On a backbone network, core nodes are the busiest and often they are transit nodes for many "passing-by" multicast sessions. Relieving these core nodes from per-micro-flow multicast forwarding enables better scalability with the number of concurrent multicast sessions. In addition, an aggregated tree doesn't go away or come up as individual groups that use it, thus tree maintenance can be a much less frequent process than in conventional multicast. The benefit of control overhead reduction is also very important in helping achieve better scalability.

There are a number of concerns raised by this approach. A prime concern is membership dynamics. The problem occurs when a new edge node is added but it is not covered by the current tree, or when an edge node leaves the group and yet it still receives multicast traffic for this group (ie, bandwidth wastage). These problems can be alleviated by allowing a group to switch dynamically from one tree to another. To avoid the problems caused by membership dynamic changes, an ISP should require a customer to provide a list of group members (i.e., borders routers connecting to customer networks participating in the group) prior to the start of a multicast session and not to change group membership for the life of the multicast session - this is like providing a multi-point "VPN" (virtual private network) service. On the other hand, one may argue that, membership change on the backbone is very infrequent for many applications. For example, an Internet TV station may use an ISP's national backbone to distribute its programming to local regional networks, then to subscribers. There can be frequent membership dynamics at access networks connected to subscribers, but membership of backbone nodes is likely to be fixed or change very slowly if there is a large population of TV viewers. Another example is video-conferencing in which participants are expected to be in the group throughout the session or over a long period of time.

In group to aggregated tree matching, complication arises when there is no **perfect match** or no existing aggregated tree covers a group. A match is a **perfect** or **non-leaky match** for a group if all its leaf nodes are terminal nodes for the group thus traffic will not "leak" to any nodes that do not need to receive it. For example, the aggregated tree with nodes (A1, A2, A3, Aa, Ab) in Fig. 1 is a perfect match for our early multicast group  $G_0$  which has members (D1, B1, C1). A match may also be a **leaky match**. For example, if the above aggregated tree is also used for group  $G_1$  which only involves member nodes (D1, B1), then it is a leaky match since traffic for  $G_1$  will be delivered to node A3 (and will be discarded there since A3 does not have state for that group). A disadvantage of leaky match is that certain bandwidth is wasted to deliver data to nodes that are not involved for the group. Now let's get back to the problem. When no perfect match is found, a leaky match may be used, if it satisfies certain constraint (e.g., bandwidth overhead is within a certain limit). This is often necessary since it is not possible to establish aggregated trees for all possible group combinations. The trade-off is bandwidth overhead vs. the benefit of aggregation. When no existing aggregated tree covers a group, either conventional multicast is used, or a new tree is established or an existing tree is extended (by adding new nodes) to cover that group. Of course, it is

possible to enforce that aggregation is only applied to groups that are covered by a set of aggregated trees established based on long-term traffic pattern and any other group will use conventional multicast.

### 3 Provision Multicast with QoS Guarantees

One motivation for aggregated multicast is to provision multicast services with QoS guarantees in future QoS-enabled networks. This problem has not attracted much attention yet within IETF since the first priority so far has been to provide IP data (typically, best effort) multicast services. We note however that real time, interactive multicast applications will be in the future at least as important as (if not more than) data, or more generally, non real time multicast applications. There is an interesting reason why the support of QoS oriented multicast for interactive applications will become very important in the future Internet. Today, many non real time applications such as news, software distribution, etc, can be effectively supported by alternate techniques (to network level multicasting) such as web caching and application level multicast. In fact, these alternate techniques are often invoked to bypass the problems posed by IP level multicast. Real time (but, non interactive) applications such as video on demand can take advantage of the same alternate techniques (eg, 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. Moreover, interactive applications cannot be effectively supported by multiple unicast connections, since they typically requires many to many communications (it would be extremely costly to provision  $N \times N$  connections, each with guaranteed bandwidth allocation!). As a result, it is important to address the scalability of QoS multicast, since it will be a prominent offering in the gamut of future Internet services.

As we already did for data multicast, the main technique we will be proposing in order to reduce router processing O/H and enhance scalability of QoS multicast is tree "aggregation". To this effect, we wish to note that Internet community has already embraced "flow aggregation" as the philosophy for scalable QoS provisioning. In fact, today people are backing away from the micro-flow based QoS architecture, namely the Integrated Services architecture[5], and are moving towards aggregated flow based architecture - the Differentiated Services architecture[4] - at least in the network core. The argument backing the aggregated approach is simple: the per-flow reservation and data packet handling required by Integrated Services simply do not scale to large networks.

As of now, however, the success of the Diff Serv concept as observed in the QoS unicast applications has not materialized yet in the QoS multicast world. In fact, over the past few years, several meritorious QoS multicast schemes have been proposed, all still inspired by the Int Serv model, and all dealing with individual flows.

The main criticism one can move to such schemes is poor scalability. In earlier sections we argued on the O/H required to set up and maintain routes for individual best effort multicast groups. The problem becomes much more complex if one must

allocate and maintain not only routes but also resources (eg, bandwidth) for individual groups. The scalability problem, however, has not so far deterred the investigation of Int Serv QoS multicast solutions. The main reason was the lack of incentive: given that conventional multicast routing requires per flow state, then, why should we seek non per flow state QoS multicast solutions.

The emergence of aggregated, scalable techniques like the one presented in the previous sections of this paper will clearly change the situation. If multicast routing has become scalable, then QoS provisioning to multicast applications should also be scalable. In fact, the aggregated multicast solutions presented in the previous sections will be the starting point for scalable, real time QoS multicast services in the Internet.

### **3.1 QoS Provisioning on the Aggregated Tree**

To understand how several multicast groups can be aggregated and managed with QoS support, one may go back to the MPLS (Multi Protocol Label Switching) concept mentioned in the previous section. In essence, the aggregated multicast scheme can be viewed as the extension of MPLS from the path to the tree. MPLS plays a key role in unicast QoS support. It “pins down” the path, allowing the use of arbitrary alternate paths not available from the common routing tables. It tunnels several sessions on the same path, by encapsulating the IP packets in an MPLS envelope. It enables QoS provisioning and grooming on a per MPLS path basis (as opposed to a per flow basis). As a consequence, CAC on individual sessions is carried out at the edge node (or Border Gateway) only, with minimal latency and without engaging the intermediate nodes along the MPLS path. It enables “measurement based” resource tracking and Call Acceptance Control. Namely, the edge node need not keep track of the exact number of IP telephony calls (say) currently multiplexed on the MPLS path; it simply monitors MPLS utilization (using a proper window average) and determines available bandwidth and acceptance/rejection policy. Finally, the MPLS mechanism allows very flexible sharing of bandwidth across all flows multiplexed on the same path. The MPLS approach is consistent with the Diff Serv principles of flow aggregation. No per flow resource allocation or signaling is required at the intermediate core routers.

The proposed aggregated multicast approach will extend all the above described MPLS features to the Aggregate Tree. In particular, QoS provisioning will be done in the background, and may be coordinated between the bandwidth broker of the domain in question and the shared tree managers (assuming one resource manager per shared tree).

### **3.2 QoS Aggregate Tree Implementation and Operation**

In the following we outline a straw-man implementation of the QoS Aggregated Multicast Tree. This implementation relies on and expands upon the basic AT implementation described in the previous section.

(a) When an AT (Aggregated Tree) is initialized, and is earmarked for the support of a particular QoS application (eg, video conference), it receives a bandwidth allocation commensurate to the traffic predictions for that application among that particular set of

destinations. The AT is a permanent tree in that it is long-lived and expected to carry a large number of sessions simultaneously.

(b) A measurement based bandwidth management scheme assures that the ratio (average traffic load)/(allocated bandwidth) is adequate for the application (accounting for both traffic statistical characteristics and application QoS requirements) and for the current load. Note that different applications may require different safety margins depending on their statistical characteristics and their delay and packet loss constraints. If necessary, traffic statistics of an application can be “learned” by observing the flows at the edge nodes. The AT bandwidth manager can be centralized or distributed. In a distributed implementation, each edge node has a bandwidth agent (BA). The BA continuously monitors the above mentioned ratio and acquires/releases bandwidth as appropriate. For example, if more bandwidth is needed, the BA uses the existing intra domain tools (eg, Q-OSPF) to determine if more bandwidth is available on the path from an edge node to the Core or Rendezvous Point router (assuming a CBT approach). The peripheral BAs exchange information about bandwidth available and come up with a consistent bandwidth allocation decision (note that it would not help if one BA allocated 10 Mbps and another allocated only 5Mbps!). The BA allocation decisions may be supervised by the domain bandwidth broker, to ensure fair resource allocation across ATs supporting different applications.

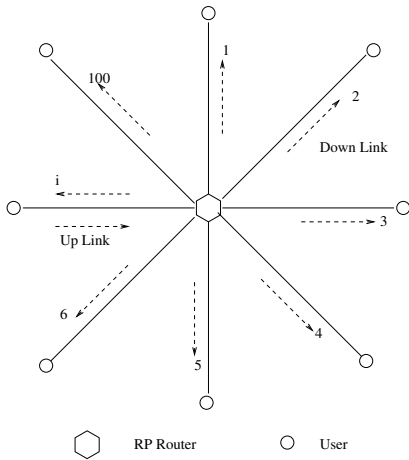
(c) Call Acceptance Control is decided at the edge node, based on “measured” available bandwidth on the AT. This eliminates the very high latency typically experienced in conventional “per flow” QoS multicast approaches.

(d) Users dynamically join/leave an existing multicast AT in a totally transparent way and with zero latency - no bandwidth needs to be allocated or released. This is a dramatic improvement with respect to the node processing O/H and latency required by per flow QoS schemes.

### 3.3 Statistical Allocation Advantage

Typically, the use of the AT implies a “wastage” of resources since the streams are delivered to more destinations than strictly necessary. This wastage is traded off with the reduction in processing O/H and the ease of path and resource maintenance. There are situations, however, when the Aggregation approach can in fact lead to bandwidth allocation savings. We outline one such example below.

Consider a video conference with a maximum of 100 simultaneous participants placed at different locations. The multicast tree is for simplicity a star with direct, point to point links from user to RP router (see Fig. 2). Assume that 1 Mbps “equivalent” bandwidth is required by each session. Typically, at any given time only the video and audio of the person currently speaking is multicast to the group. In the traditional IntServ, “per flow” reservation approach, a full duplex 1 Mbps allocation is required (on the link from user to RV router) when a user joins the group. Thus, the “total” bandwidth allocation (counting the unidirectional bandwidth in each direction of the link) is  $2S$  Mbps, where  $S$  is the number of current participants. If we use the AT scheme, the total allocated bandwidth is 101 Mbps, regardless of the number of simultaneous participants. This is because the AT bandwidth agent BA measures bandwidth usage, that is 1Mbps on each link in the direction RP to edge node, and 1 Mbps summed over



**Fig. 2.** Videoconference multicast group.

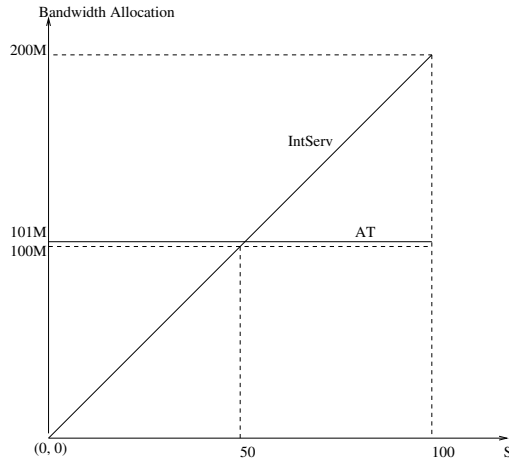
all uplinks from edge node to RP (assuming that only one member is transmitting at a time). If we plot the allocated bandwidth as a function of  $S$  (see Fig. 3), we note that the Int Serv scheme requires “more” bandwidth than the AT scheme for  $S > 50$ . This is an unexpected result, which tells us that because of the more flexible, statistical allocation in the Aggregated Tree, we stand to save instead of waste bandwidth! The careful reader will notice that the Aggregate Tree saving leads to an allocation that is “asymmetric”; this asymmetry can be compensated across different trees, which typically have different roots and different topology layouts. Moreover, even the IntServ scheme could take advantage of the statistical sharing of uplink bandwidth among various transmitters. But, this would require checking the bandwidth allocation and state of several different multicast groups at each intermediate node!

### 3.4 Extensions of the Basic QoS AT Scheme

Within a domain, the ISP will offer several “permanent” ATs to choose from. The multicast group manager will periodically query the AT layout database for information regarding all the installed ATs, and will dynamically select the tree that best matches its current members configuration. As the membership grows, the groups can simply switch from one tree to another. Note that by virtue of the “soft state” operation induced by the measurement scheme, this switch-over is totally transparent. It does not require any bandwidth reallocation.

In some cases, the addition of a couple of users in locations not served by the current tree may not warrant the switch-over to a new tree. The new users can then be easily accommodated by connecting them to the nearest edge node/router.

In some applications (eg, battlefield communications, distributed visualization and control, etc.) it is important to provide fault tolerant multicast. For example, consider the control of a space launch carried out from different ground stations interconnected by an Internet multicast tree. This control scenario may require the exchange of real



**Fig. 3.** Total bandwidth allocation as a function of active conference participants.

time, interactive data and streams. One elegant way to provide fault tolerance is the use of separate, possibly node disjoint multicast trees. For added reliability and zero switch-over latency, the duplicate data could be sent on both trees simultaneously.

Another scenario in which the Aggregate Tree concept is beneficial is mobile hand-off. Consider for example a video conference participant driving between Los Angeles and San Diego. Assume that both Los Angeles and San Diego are leaves of the AT tree to which the multicast group of the mobile user has subscribed. With our proposed scheme, as the user is “handed off” from the Los Angeles to the San Diego edge router, he finds a path with resources already allocated to him. There is minimal disruption of communications. This “soft handoff” is achieved with no overhead in the core network.

## 4 Simulation Studies for State Reduction

In this section we attempt to quantify multicast state reduction that can be achieved using aggregated multicast. It is worth pointing out that our approach of multicast “aggregation” is completely different from multicast “state aggregation” approaches in [10, 13]. We aggregate multiple multicast groups into a single tree to reduce the number of multicast forwarding entries, while their approach is to aggregate multiple multicast forwarding entries into a single entry to reduce the number of entries. It is possible to further reduce multicast state using their approaches in an aggregate multicast environment. Here we study state reduction achieved by “group aggregation” before any “state aggregation” is applied.

First, we introduce two state reduction metrics. Without losing generality, we assume a router needs one state entry per multicast address in its forwarding table. Here we care about the **total number** of state entries that are installed at **all** routers involved to support a multicast group in a network. In conventional multicast, the total number of entries for a group equals the number of nodes  $|T|$  in its multicast tree  $T$  (or subtree within a domain, to be more specific) – i.e., each tree node needs one entry for

this group. In aggregated multicast, there are two types of state entries: entries for the shared aggregated trees and group-specific entries at terminal nodes. The number of entries installed for an aggregated tree  $T$  equals the number of tree nodes  $|T|$  and these state entries are considered to be **shared by all groups** using  $T$ . The number of group-specific entries for a group equals the number of its terminal nodes because only these nodes need group-specific state.

Thus, we come up with the concept of **irreducible state** and **reducible state**: group-specific state at terminal nodes is **irreducible**. All terminal nodes need such state information to determine how to forward multicast packets received, no matter in conventional multicast or in aggregated multicast. For example, in our early example illustrated by Fig. 1, node A1 always needs to maintain state for group  $G_0$  so it knows it should forward packets for that group received from D1 to the interface connecting to Aa and forward packets for that group received from Aa to the interface connecting to node D1 (and not X1 or Y1), assuming a bi-directional inter-domain tree.

Let  $N_a$  be the total number of state entries to carry  $n$  multicast groups using aggregated multicast,  $N_0$  be the total number of state entries to carry the same  $n$  multicast groups using conventional multicast. We introduce the term **overall state reduction ratio** – i.e., total state reduction achieved at all routers involved in multicast, intuitively defined as

$$r_{as} = 1 - \frac{N_a}{N_0}. \quad (1)$$

Let  $N_i$  be the total number of irreducible state entries all these group need (i.e., sum of the number of terminal nodes in all groups), **reducible state reduction ratio** is defined as

$$r_{rs} = 1 - \frac{N_a - N_i}{N_0 - N_i}, \quad (2)$$

which reflects state reduction achieved at transit or core routers.

Further more, we define another metric about aggregation overhead. Assume an aggregated tree  $T$  is used by groups  $G_i, 1 \leq i \leq n$ , each of which has a “native” tree  $T_0(G_i)$ , the **average aggregation overhead** for  $T$  is defined as:

$$\begin{aligned} \delta_A(T) &= \frac{n \times C(T) - \sum_{i=1}^n C(T_0(G_i))}{\sum_{i=1}^n C(T_0(G_i))} \\ &= \frac{n \times C(T)}{\sum_{i=1}^n C(T_0(G_i))} - 1, \end{aligned} \quad (3)$$

where  $C(T)$  is the cost of tree  $T$  (total cost of all  $T$ 's links). Intuitively,  $\delta_A(T)$  reflects the amount of extra bandwidth wasted to carry multicast traffic using the shared aggregated tree  $T$ , in percentage. Let  $N_g$  be the total number of multicast groups and  $N_t$  be the total number of aggregated trees used to support these groups, **average aggregation degree** – i.e., the average number of groups an aggregated tree “matches”, is defined as

$$AD = \frac{N_g}{N_t}. \quad (4)$$

The larger this number, the larger the number of groups that are aggregated into an aggregated tree, and correspondingly the more the state reduction. This number also

reflects control overhead reduction: more groups an aggregated tree supports, fewer number of trees are needed and thus less control overhead to manage these trees (fewer refresh messages, etc.).

Next we will present simulation results from a dynamic matching experiment allowing leaky matches. In this experiment, we use the Abilene[1] network core topology as our simulation network, which has eleven nodes located in eleven metropolitan areas. Distance between two locations is used as the routing metric (cost), which could result in different routes than the real ones; however, routes from UCLA to a number of universities (known to be connected to Internet 2) discovered by traceroute are consistent with what we expect from the Abilene core topology using distance as routing metric.

We randomly generate multicast groups and use the following strategy to match them with aggregated trees and establish more aggregated trees when necessary. In generating groups, every node can be a terminal node (i.e., we don't single out any node to be core node that is not directly accessible to neighboring networks); in simulation results to be presented, group size is uniformly distributed from 2 to 10. When a group  $G$  is generated, first a source-based "native" multicast tree  $T_0$  (with a member randomly picked as the source) is computed. An aggregated tree  $T$  (from a set of existing ones, initially empty) is selected for  $G$  if the following two conditions are met: (1)  $T$  covers  $G$ ; and (2) after adding  $G$ ,  $\delta_A(T) \leq b_{th}$ ; where  $b_{th}$  is a fixed threshold to control  $\delta_A(T)$ . When multiple trees satisfy these conditions, a min-cost one is chosen. If no existing tree satisfies these conditions, either (1) an existing tree  $T$  is extended (by adding necessary nodes) to cover  $G$  if the extended tree  $T'$  can satisfy the following condition: after adding  $G$ ,  $\delta_A(T') \leq b_{th}$ ; or (2) the native tree for  $G$  is added as a new aggregated tree. Constraints above guarantee that bandwidth overhead is under a certain threshold.

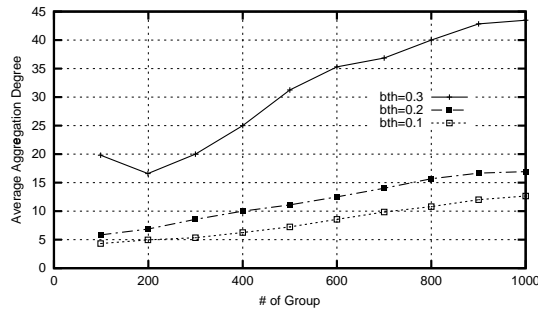


Fig. 4. Average aggregation degree vs. number of groups.

Fig. 4 plots the simulation result of average aggregation degree vs. number of groups added for different bandwidth overhead thresholds. As the result shows, as more groups are added (i.e., more concurrently active groups), the average aggregation degree increases: we can "squeeze" more groups into an aggregated tree, in average. Bandwidth overhead threshold affects aggregation degree in a "positive" way: as we lift the control threshold, more aggregation can be achieved – as we are willing to "sacrifice" more bandwidth for aggregation, we are getting more aggregation. Fig. 5 and Fig. 6 plot the

results for overall state reduction ratio and reducible state reduction ratio defined in Eq. 1 and 2, and demonstrate the same trend regarding the number of groups and bandwidth overhead threshold as aggregation degree. The results show that, though overall state reduction has its limit, reducible state is significantly reduced (e.g., over 80% for a 20% bandwidth overhead threshold). This also confirms our early analysis.

In interpreting the implications of the above simulation results, we should be aware of their limitations: the network topology is fairly small and it is adopted from a logic topology and not really a backbone network with all routers at presence. Nevertheless, it should give us some feelings about the “trend”. Another fine point is that, this simulation represents a worst-case scenario since all groups are randomly generated and has no correlation or pattern. In practice, certain multicast group membership pattern (locality, etc.) may be discovered from measurements and can help to realize more efficient aggregation.

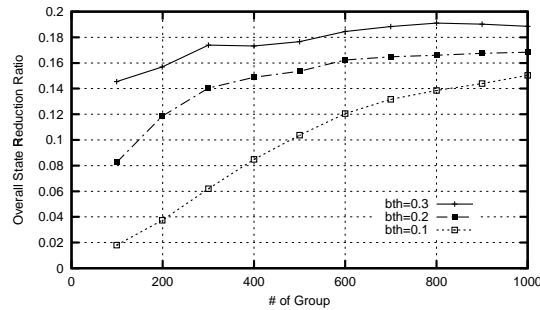


Fig. 5. Overall state reduction ratio vs. number of groups.

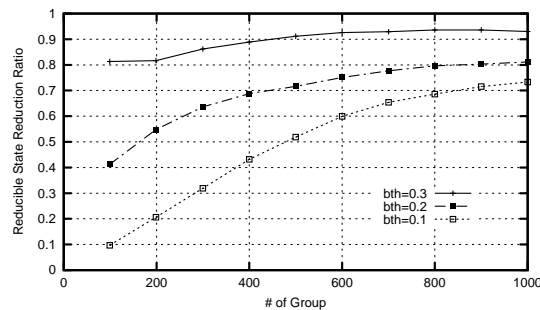


Fig. 6. Reducible state reduction ratio vs. number of groups.

## 5 Conclusions

In this paper, we proposed a novel approach, aggregated multicast, to provision QoS multicast within intra-domain. The key idea of aggregated multicast is to force groups into sharing a single delivery tree. This way per-flow state is eliminated from network core and is only required at edge routers.

Our work could be summarized in the following points:

- Aggregated multicast is an unconventional yet feasible and promising approach.
- We discussed how this approach can be used to provision multicast with QoS guarantees.
- We proposed metrics to quantify multicast state reduction in aggregated multicast and our initial simulation shows promising results.

## References

1. Abilene network topology. <http://www.ucaid.edu/abilene/>.
2. A. Banerjea, M. Faloutsos, and E. Crawley. Qosmic: a quality of service sensitive multicast internet protocol. *Internet draft: draft-banerjea-qosmic-00.txt*, October 1998.
3. 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.
4. S. Blake, D. Black, and et al. An architecture for differentiated services. *IETF RFC 2475*, 1998.
5. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. *IETF RFC 1633*, 1994.
6. P. Francis. Yoid: extending the internet multicast architecture. <http://www.aciri.org/yoid/docs/index.html>.
7. J. Hou, H.-Y. Tyan, B. Wang, and Y.-M. Chen. Qos extension to cbt. *Internet draft: draft-hou-cbt-qos-00.txt*, February 1999.
8. S. Keshav and S. Paul. Centralized multicast. *Proceedings of IEEE ICNP*, 1999.
9. D. Ooms, R. Hoebeker, P. Cheval, and L. Wu. MPLS multicast traffic engineering. *Internet draft: draft-ooms-mpls-multicast-te-00.txt*, 2001.
10. P. I. Radoslavov, D. Estrin, and R. Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical report, USC Dept. of CS Technical Report 99-697 (Second Revision), July 1999.
11. Y. Chu S. Rao and H. Zhang. A case for end system multicast. *Proceedings of ACM Sigmetrics*, June 2000.
12. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *IETF RFC 3031*, 2001.
13. D. Thaler and M. Handley. On the aggregatability of multicast forwarding state. *Proceedings of IEEE INFOCOM*, March 2000.
14. J. Tian and G. Neufeld. Forwarding state reduction for sparse mode multicast communications. *Proceedings of IEEE INFOCOM*, March 1998.
15. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: a new resource reservation protocol. *IEEE Network*, September 1993.