Aggregated Multicast: an Approach to Reduce Multicast State

Aiguo Fei, Junhong Cui, Mario Gerla,
Computer Science Department
University of California
Los Angeles, CA 90095

Michalis Faloutsos
Computer Science & Engineering
University of California
Riverside, CA 92521

Abstract—IP multicast suffers from scalability problem with the number of concurrently active multicast groups because it requires a router to keep forwarding state for every multicast tree passing through it and the number of forwarding entries grows with the number of groups. In this paper, we propose an approach to reduce multicast forwarding state. 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 propose metrics to quantify state reduction and analyze the bounds on state reduction of our approach. Finally, we use simulations to verify our analytical bounds and quantify the state reduction. These initial simulation results suggest that our method can reduce multicast state significantly.

I. Introduction

Multicast state scalability is the problem we address in this work. Multicast is a mechanism to efficiently support multipoint 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. In unicasting, address aggregation coupled with hierarchical address allocation has helped to achieve scalability. This can not be done for multicasting easily if not impossible at all, since a multicast address corresponds to a logical group and does not convey any information on the location of its members. 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. This perhaps is the main scalability problem with IP multicast when the number of simultaneous on-going multicast sessions is very large.

Recently, significant research effort has focused on the problem of multicast state scalability. Some schemes attempt to reduce forwarding state by tunneling[12] or by forwarding state aggregation[8, 11]. Thaler and Handley analyze the aggregatability of forwarding state in[11] 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 [8]. 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 [4, 9] using network-transparent multicast, which pushes the complexity to the end-points.

In this paper, we propose a novel scheme to reduce multicast state, 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 study the problem analytically deriving some bounds on the reduction of state. 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 II introduces the concept of aggregated multicast approach and discusses some implementation related issues. Section III proposes metrics to quantify multicast state reduction in aggregated multicast and analyzes the bounds on reduction ratios and presents simulation results. Section IV gives a short summary of our work.

II. 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.

A. 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

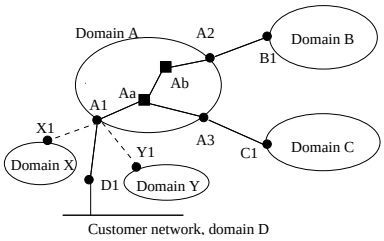

Fig. 1. Domain peering and a cross-domain multicast tree, tree nodes: D1, A1, Aa, Ab, A2, B1, A3, C1, covering group G_0(D1, B1, C1).

that the traffic injected into router A1 by router D1 will be distributed over that intra-domain tree and reaches router A2 and A3.

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[2], 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.

B. Implementation Considerations

It is not our goal to establish an architecture or 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 a couple of possibilities to distribute multicast traffic of different groups over a shared aggregated tree. For any implementation, there are two requirements: (1)original group addresses of data packets must be preserved somewhere and can be recovered by exit nodes to determine how to further forward these packets; (2)some kind of identification for the aggregated tree which the group is using must be carried and transit nodes must forward packets based on that. One possibility is to use IP encapsulation as said above, which, of course, adds complexity and processing overhead (at terminal nodes). Another potentially much better possibility is to use MPLS (Multiprotocol Label Switching)[10] 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 those terminal 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[5], or use MPLS signaling protocols extensions proposed in [6] which allow 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 MCIWorldcom'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, given group membership (set of terminal nodes) information, to an aggregated tree that covers the group (i.e., all terminal nodes are member nodes of the tree).

C. Discussions

A related motivation for aggregated multicast is how to simplify the provisioning of multicast with QoS guarantees in future QoS-enabled networks. Regarding QoS support, per-flow-based traffic management requirement of Integrated Services[3] does not scale. That is why, today providers are backing away from it and are moving towards aggregated flow based Differentiated Services[2]. The intrinsic per-flow nature of multicast may be problematic for DiffServ networks especially in provisioning multicast with guaranteed service quality. Aggregated multicast can simplify and facilitate QoS management for multicast by pre-assignment of resource/bandwidth (or reservation on demand) in a smaller number of shared aggregated trees. A centralized tree manager coupled with admission control and policing at *source* nodes (at network edge) then can do the same for multicast as what bandwidth broker does for point-to-point flows in DiffServ. Aggregated multicast may thus pave the way for ISP's to provide bandwidth guaranteed service of simultaneous multi-point data distribution.

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 net-

work, 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 that one may have with this approach. First, it relies on a centralized tree manager for tree management, which can be overloaded and be a single point of failure. There are two ways of looking at this problem. First, a variety of systems, such as clusters and processor groups, should be able to provide the processing power and speed required – considering how amazingly many web servers scale up; at the same time, a distributed system can remedy the single point of failure problem. On the other hand, such reliance is common in real life. Most core routers at prominent service providers' backbone networks can not fail without serious consequences. One may also argue that the problem of requiring a tree manager is no worse than that of requiring a few RP routers in PIM-SM or a bandwidth broker in DiffServ[2]. This discussion has its root in debate regarding centralized vs. decentralized multicast. In practices, however, centralized approach is indeed adopted in multicast in ATM networks[7], and centralized tree calculation is one possibility proposed in an MPLS multicast traffic engineering proposal[6].

Another concern with this approach is membership dynamics. The problem aries when a new exit node is added but it is not covered by the current tree, or when an exit node leaves the group and it may cause too much bandwidth overhead if the current tree is still used for this size-reduced group. This can be solved by allowing a group to switch from one tree to another. While this process might be slower than in conventional multicast and causes additional complication especially if bandwidth guarantee (QoS) is involved, it is clearly doable. To avoid membership dynamics completely, an ISP may require a customer to provide a list of group members prior to the start of a multicast session and not to change group membership in the future – this is like 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 tradeoff 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.

III. State Reduction: Analysis and Simulation

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 [8, 11]. We aggregate multiple multicast groups into a single tree to reduce the number 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.

A. Analysis

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.

Furthermore, We also introduce 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.

Now consider state reduction achievable with a single aggregated tree. Consider an aggregated tree T with $m = |T|$ nodes and k_0 leaf nodes ($k_0 \leq m$), and assume it is used as a **perfect match** for a total of n groups each has $k \geq k_0$ terminal nodes. Assume the "native" multicast tree for a group G using this aggregated tree is T_0. Since all T's leaf nodes are terminal nodes for G, it is reasonable to assume $T = T_0$ if both trees are constructed using, say, CBT (or PIM-SM) with the same core (or RP) router. If source-trees (say, rooted at a source node) are used, they can be different, but we can assume $|T| = |T_0| = m$ as an approximation for our discussion. Now consider multicast state entries installed at all routers involved using two different approaches. Using native multicast, the number of entries is $N_0 = n \times m$ (i.e., each group needs a total of m entries at m routers). Using aggregated multicast, the number of entries if $N_a = m + n \times k$: the first term is the number of entries for the aggregated tree and the second term is the number of entries installed at terminal nodes. We get

$$r_{as} = 1 - \frac{m + n \times k}{n \times m} = 1 - \frac{k}{m} - \frac{1}{n}. \quad (3)$$

When n is large enough,

$$r_{as} \sim 1 - \frac{k}{m} \leq 1 - \frac{k_0}{m}, \quad (4)$$

which provides an upper bound on r_{as}. To achieve $r_{as} > 0$, we need $n > \frac{1}{1-k/m}$; i.e., to "squeeze" that many of groups into an AT. The percentage of reducible state entries that can be eliminated by aggregated multicast is:

$$r_{rs} = 1 - \frac{N_a - n \times k}{N_0 - n \times k} = 1 - \frac{m}{n(m-k)} = 1 - \frac{1}{n(1 - \frac{k}{m})}. \quad (5)$$

When $k = m$ (i.e., all tree nodes are terminal nodes, an unlikely event), no state is reducible and extra state entries are actually introduced by using an aggregated tree (i.e., m entries). When $k < m$, this ratio approaches 100% when n is large enough; i.e., a large number of reducible state entries are reduced to only m entries required by the aggregated tree. If not all multicast groups using aggregated tree T have k terminals or leaky matches exist, it gives us an approximation for state reduction if we replace k with the average number of terminal nodes $\bar{k}$ in the above equations. Apparently the lower the terminal node ratio($\frac{k}{m}$), the higher state reduction ratio can be achieved.

The above analysis tells us that, while that the overall state reduction ratio which reflects state reduction at all multicast routers is bounded by the percentage of terminal nodes, the reducible state reduction ratio which reflects state reduction at transit or core routers can approach 100% if we can "squeeze" enough groups into an aggregated tree – transit nodes only need state for aggregated trees, the number of which is much smaller than the number of groups.

B. Simulation

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.

First we introduce two concepts. 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:

$$\begin{aligned} \delta_A(T) &= \frac{n \times C(T) - \sum_{i=1}^{n} C(T_0(G_i))}{\sum_{i=1}^{n} C(T_0(G_i))} \\ &= \frac{n \times C(T)}{\sum_{i=1}^{n} C(T_0(G_i))} - 1, \end{aligned} \quad (6)$$

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

$$AD = \frac{N_g}{N_t}. \quad (7)$$

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.).

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

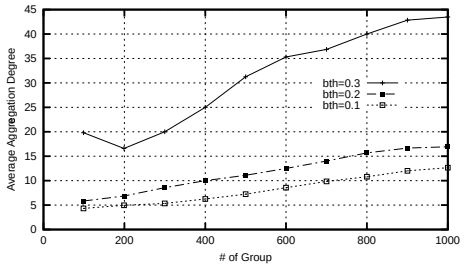

Fig. 2. Average aggregation degree vs. number of groups.

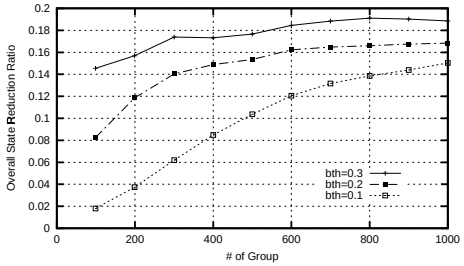

Fig. 3. Overall state reduction ratio vs. number of groups.

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. 2 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. 3 and Fig. 4 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.

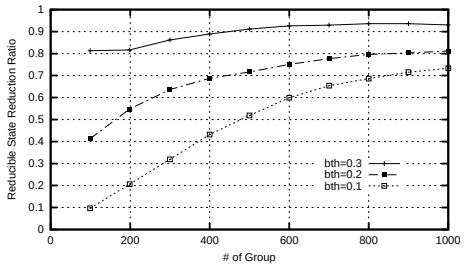

Fig. 4. Reducible state reduction ratio vs. number of groups.

IV. Conclusions and Future Work

In this paper, we propose a novel approach to resolve the problem of multicast state scalability. The key idea of aggregated multicast is to force groups into sharing a single delivery tree, instead of creating multiple trees and then trying to aggregate the state.

Our work could be summarized in the following points:

- Aggregated multicast is an unconventional yet feasible and promising approach.
- We derive analytical bounds for the amount of state reduction we can get from aggregated multicast.
- Initial simulations show promising results. In simulation experiments with a 30% bandwidth overhead control, aggregated multicast can reduce 90% or more of the reducible state and roughly 16% of the total multicast state for large numbers of groups. Recall that we assumed no group-member locality, which is the worst case scenario.
- Our analytical and experimental results reinforce each other exhibiting similar trends.

In addition, our scheme provides a mechanism for simplified QoS multicast provisioning and a mechanism for multicast traffic engineering. We find that this by itself could be a sufficient reason for aggregated multicast.

FUTURE WORK. We are in the process of conducting more simulations under different environments. We are particularly interested in assessing the effect of member locality in the performance of our approach. Initial observations suggest that it will have significant positive impact.

References

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

[2] S. Blake, D. Black, and et al. An architecture for differentiated services. *IETF RFC 2475*, 1998.

[3] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. *IETF RFC 1633*, 1994.

[4] P. Francis. Yoid: extending the internet multicast architecture. *http://www.aciri.org/yoid/docs/index.html*.

[5] S. Keshav and S. Paul. Centralized multicast. *Proceedings of IEEE ICNP*, 1999.

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

[7] Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*. Addison-Wesley Publishing Company, 2nd edition, October 1999.

[8] 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.

[9] Y. Chu S. Rao and H. Zhang. A case for end system multicast. *Proceedings of ACM Sigmetrics*, June 2000.

[10] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *IETF RFC 3031*, 2001.

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

[12] J. Tian and G. Neufeld. Forwarding state reduction for sparese mode multicast communications. *Proceedings of IEEE INFOCOM*, March 1998.