DSMCast: A Scalable Approach for DiffServ Multicasting

A. Striegel
Department of Computer Science and Engineering
University of Notre Dame, USA
striegel@cse.nd.edu

G. Manimaran
Dependable Computing and Networking Laboratory
Department of Electrical and Computer Engineering
Iowa State University, USA
gmani@iastate.edu

Abstract—One of the dominant questions facing the Internet today is, how can the network meet the needs of the users and their applications (QoS) while trying to keep such implementations scalable to the billions of users present on the Internet? Two of the emerging technologies for answering this question are Differentiated Services (DiffServ) and multicasting. Although both are complementary technologies, the integration of the two technologies is a non-trivial task due to architectural conflicts between multicasting and DiffServ. In this paper, we first investigate viable approaches for DiffServ multicasting and then propose an efficient approach, called DSMCast, for multicasting across an individual DiffServ domain that is scalable in terms of group size, network size, and number of groups. In addition, we propose a protocol for member join/leave and detail our protocol for both SSM (Source-Specific Multicast) and traditional IP multicasting. Finally, we present simulation studies of the DSMCast architecture comparing it with alternate approaches and offer conclusions regarding our work.

Keywords - Differentiated Services, Multicasting, Dynamic DSCPs, Member Join/Leave, Heterogeneous QoS

I. INTRODUCTION

Recently, there has been a push from business and user communities for next generation applications demanding Quality of Service (QoS). However, the Internet in its current form does not support the notion of Quality of Service (QoS). Rather, the Internet follows the same-service-to-all paradigm in which all packets receive the same QoS. This best-effort service model is inadequate in meeting the growing demands of the next generation applications, most of which demand QoS assurances for effective data delivery and presentations.

In order to meet the basic QoS requirements of applications using the network, there are two primary schools of thought governing how resources should be allocated. The first school of thought is to increase the bandwidth available to users such that the extra capacity of the network allows all users to meet their appropriate QoS. By providing capacity beyond the needs of the users on the network (over-provisioning), the network will not become congested and thus all users will be able to meet their QoS. QoS is provided by default and requires no additional overhead in the routers or by the end-users themselves.

In contrast, the second school of thought is that bandwidth can never be considered unlimited and therefore the limited bandwidth should be appropriately prioritized among users. Although the bandwidth of the Internet backbone has dramatically increased [1], it is still a subject of active debate whether appropriate resource allocation mechanisms are needed to meet the QoS requirements of the majority of users. It is the premise of this paper that for the foreseeable future some form of resource provisioning is necessary to provide QoS across the Internet.

A. Differentiated Services

The Differentiated Services (DiffServ) architecture [2] was proposed by the IETF for providing scalable QoS across the Internet. Rather than addressing the issue of end-to-end QoS as the Integrated Services (IntServ) model [3] had done, DiffServ focuses on per-hop behavior (PHB) [4] and QoS management at the domain/ISP (AS) level. In the DiffServ architecture, intelligence is migrated to the edge of the domain in order to keep the core of the network simple and scalable.

Routers in a DiffServ domain are divided into two categories, core routers (simple and high speed) and edge routers (stateful and intelligent). Core routers do not have per-flow state and differentiate packets according to the marking (DSCP - DiffServ Code Point) of the packet. In contrast, edge routers are stateful entities that are responsible for policing and/or marking all packets according to an SLA (Service Level Agreement) between
the source (other ISP, user, company) and the domain, or between two domains.

B. Multicasting

Although multicasting was not specifically designed with the traditional notion of QoS in mind, the use of multicast can have a significant effect on QoS. Most notably, multicast can have a significant impact on the bandwidth utilization of many of the newly emerging applications that demand such QoS guarantees. For many of these new applications (e.g., video/audio on demand, peer to peer sharing, teleconferencing, distributed gaming), the traditional unicast model is highly inefficient for supporting such applications [5]. By reducing multiple unicast connections to a multicast tree with a minimized bandwidth cost, multicast frees additional bandwidth (and hence QoS) for other applications.

From the perspective of both the end user and the network service provider, multicasting could offer tremendous benefit to both network efficiency and QoS. However, the issue of how to support multicasting in DiffServ, has received relatively little research attention. Although the two concepts of bandwidth conservation (multicast) and scalable QoS management (DiffServ) are complementary, the emphasis on scalability by DiffServ creates architectural conflicts with multicasting that make the integration of the two technologies a non-trivial task [6].

C. Analyzing DiffServ & Multicasting

Similar to DiffServ, multicasting relies on routers in the network to provide its underlying functionality. Whereas DiffServ partitions the network routers into two distinct types (core and edge routers), multicasting is either supported in routers or not supported. Ideally in multicast, all routers on the end-to-end path support multicasting in order to maximize efficiency (i.e., resource sharing). The proliferation of multicasting allows for the minimal cost multicast tree whereby the multicast packets are replicated based on per-group state information only where absolutely necessary.

In addition, whereas DiffServ relies on only edge routers possessing intelligence and state information, multicasting relies on per-group state throughout the entire network. Thus, when trying to integrate the two technologies, one is faced with two conflicting principles, core statelessness for scalability versus per-group state information for efficiency. The natural question is, which principle is more important, state information (storage and router complexity) or maximal network efficiency (bandwidth)? It is our belief that state information is vastly more costly than bandwidth. Whereas one could argue that if one can increase bandwidth capacity, one can also increase the state storage capacity, it is the maintenance of such state information (router complexity) that is the problem. Thus, therein lies the multicast scalability problem.

For instance, for each multicast group flowing across the domain, each router must store the state information as well as manage the state information (timers, etc.). The maintenance of the group information is further compounded by the potential for a huge number of unique multicast groups. For example, consider if an individual web server employed SSM (Source Specific Multicasting [7]) and placed each web object as an individual group. Although this may seem far-fetched at first, such a scheme is not difficult to envision [8]. Next, consider if one hundred, one thousand, or even all of the web servers on the Internet employed such a scheme. If the traditional multicasting approach were employed, the routers in the Internet would need to support and maintain state information for millions or even billions of multicast groups.

D. DiffServ & Multicast Conflicts

Although the problem of scalability in multicasting is not necessarily unique to DiffServ, the scalability problems of multicasting run counter to the fundamental principle of scalability that DiffServ was designed to address in the first place. The primary conflicts between DiffServ and multicasting are summarized below:

- **Scalability of per-group state:** The per-group state information of multicasting is not scalable to the high speeds of the core of the network. Thus, any solutions for DiffServ multicasting must address this fundamental issue.
- **Sender versus receiver-driven QoS:** Whereas DiffServ provides a sender-driven QoS (packets are marked at the sender or ingress), multicasting is a receiver-driven service. As a result, appropriate mechanisms must be developed that bridge the gap between the sender-driven QoS of DiffServ and the receiver-driven QoS of multicasting.
- **Resource management:** Unlike unicast connections, a multicast packet may replicate into one or more packets in the network. This problem is compounded by the fact that the ingress node (entrance to the DiffServ domain) may not necessarily know the exact makeup of the multicast tree. Thus, appropriate signaling or management mechanisms
must be introduced in order to manage the resource impact of multicasting on the network [9].

E. Motivation

Thus, we pose a simple question: Why not follow the DiffServ approach and minimize the number of routers that must intelligently process and maintain such multicast information? If bandwidth can be thought of as significantly cheaper than maintaining state information at all routers, can one leverage the DiffServ model to provide a better multicast?

These questions provide the motivation for our paper. In our paper, we propose a novel approach for DiffServ multicasting, DiffServ Multicast (DSMCAST), which leverages the unique aspects of the DiffServ architecture to exploit the intelligence of edge routers and maintain core statelessness. Rather than employing a multicast everywhere approach, DSMCAST reduces the problem to edge-to-edge transport across a single DiffServ domain. By migrating the per-group state information from the router to the packet, core routers are allowed to remove the complexity of multicast routing and multicast state maintenance while still providing significant bandwidth savings versus separate unicast connections. The migration of state information to the edge has implications for not only core scalability but also for resource management and heterogeneous QoS.

The rest of the paper is organized as follows. The next section, Section II, categorizes the existing solutions to the problem. Next, Section III outlines the basic concepts of the DSMCAST architecture regarding how the multicast transport service is delivered (packet header structure and tree construction). Section IV details the join/leave protocol in both the SSM and the traditional IP multicast environments. Following that, Section V examines DSMCAST as it relates to other protocols and existing models. Then, Section VI analyzes the performance of DSMCAST through both theoretical and simulated results. Finally, Section VIII concludes the paper by offering several concluding remarks regarding DSMCAST.

II. DIFFSERV MULTICAST APPROACHES

In order to offer support for DiffServ multicasting, solutions can be divided into three main classes, state-based, edge-based, and encapsulation-based. The approaches are summarized in Table I and discussed in more detail below.

A. State-Based Approach (Traditional IP Multicast)

In this approach, the per-group state information for each of the multicast groups is maintained in the edge routers as well as the core routers of the DS domain. When a new egress point (router where the packet exits the domain) wishes to join or leave a multicast group, the corresponding core and edge routers propagate upstream their state information for the multicast group that is affected. For a join request, resource reservation is handled at each router (core or edge) until the request reaches the multicast tree. This approach is the equivalent of simply applying the traditional IP multicast model without distinction to the edge or core routers of the DiffServ domain. Examples of this approach in existing literature include the draft work by the IETF [9], DAM [11], and QUASIMODO (adaptations to the PIM-SM model) [12].

To start, this approach contradicts the fundamental concept of stateless core routers in the DiffServ architecture. This approach has scalability problems because of the complexity of per-group state maintenance at all core routers. In addition, the state-based approach also pushes the complexity of multicasting protocols onto the core (control messages, resource reservation), thus violating the concept of simple core routers. The state-based approach is viable only if the number of unique multicast group stays extremely small such as currently seen today [13].

B. Edge-Based Approach

The second approach is to restrict the location of multicast capable routers within the DiffServ domain. Rather than allowing all routers to be multicast capable, only the edge routers of a DiffServ domain are allowed to be multicast capable. This approach is simple to implement and is highly scalable but incurs performance degradation in terms of bandwidth consumption versus the state-based approach. Apart from its simplicity, it is highly suitable for sparse groups where replication of packets at the core routers is less likely to occur compared to that of the dense groups. Examples of this approach include the current tunneling mechanism of MBone [5], EBM [14], and QDM [15].

In such a solution, core routers are entirely multicast unaware, thus requiring zero support for multicasting (replication or control messages). Thus, there is no additional implementation necessary in the core routers and the per-group statelessness of the edge-based approach complies with the goals of the DiffServ architecture.

C. Encapsulation-Based Approach

The third approach is to embed the multicast information within the packet itself. In the encapsulation-
<table>
<thead>
<tr>
<th>Type</th>
<th>Replication Allowed</th>
<th>Replication Information</th>
<th>Strengths (+)</th>
<th>Weaknesses (-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-based</td>
<td>Everywhere</td>
<td>Per-group state in routing table</td>
<td>+ Most efficient B/W usage</td>
<td>- Scalability</td>
</tr>
<tr>
<td>Edge-based</td>
<td>Edge routers</td>
<td>None, Tunneling [10]</td>
<td>+ Simple</td>
<td>- Dense groups</td>
</tr>
<tr>
<td>Encapsulation-based</td>
<td>Everywhere</td>
<td>Included in packet</td>
<td>+ Scalable</td>
<td>- Processing cost</td>
</tr>
</tbody>
</table>

**TABLE I**

**Summary of DiffServ multicast approaches**

Based on the encapsulation-based approach, the multicast tree for the domain is encapsulated within the multicast packet. Similar to how core routers rely on the marking of the packet to determine the PHB for unicast QoS, the encapsulation-based approach adopts a similar methodology whereby the necessary multicast information is embedded inside the packet. Hence, this approach does not require per-group state information and fits extremely well within the DiffServ architecture. Examples of this approach include the DSMCast model proposed in the paper and SGM [16] (compared in greater detail in Section V-C).

However, encapsulation does introduce several additional costs/overheads. First, encapsulation consumes additional bandwidth for each packet in the form of encoding the multicast tree and thus may not scale well with the group size. This scalability problem is alleviated by encoding only the tree for the domain, not the entire end-to-end multicast tree. As a result, the size of the tree is dependent upon only the number of egress points for the domain, not the number of actual receivers in the multicast group. The second tradeoff that occurs with encapsulation is the additional CPU cost incurred due to header processing that may be entirely alleviated through hardware support.

### III. DSMCast - An Approach for DiffServ Multicast

Unlike the state-based approach of traditional IP multicasting, the encapsulation-based approach can potentially satisfy the three conflicts between DiffServ and multicasting. Thus, although an encapsulation-based approach does incur several performance penalties, it is the premise of this paper that the benefits of such an approach (core statelessness and bandwidth savings versus separate unicasts) far outweigh the penalties (additional bandwidth).

However, the use of encapsulation and removal of core state introduce several challenges that must be addressed. To start, the first challenge is to develop an efficient packet header that conveys the multicast tree and is optimized for fast hardware processing. Next, the header must support options to accommodate heterogeneous egress points with minimal bandwidth overhead. Furthermore, the construction and maintenance of the tree and the packet header must also be addressed. Finally, a new domain-wise join/leave protocol must be designed to overcome the statelessness of core routers. Additionally, although the general trend in multicast is towards SSM [7], a complete architecture should also be able to offer support for traditional IP multicast as well as SSM. Hence, it is the motivation of this paper to address these challenges in the DSMCast architecture.

In short, the DSMCast architecture can be subdivided into four main components, the transport mechanism, the extensions for heterogeneous QoS, the tree construction mechanism, and the egress (member) join/leave protocol. Figure 1 shows the DSMCast architecture in relation to the global Internet.

#### A. Transport Mechanism

The most basic function of the DSMCast architecture is to provide a core stateless transport mechanism for multicast traffic across the domain. To accomplish this goal, all replication/routing for multicast packets is controlled by the DSMCast Tree Encapsulation Header (TEH). When a multicast packet arrives at the ingress
In the case where IP options are present, most routers divert the packet to a slow path (software) for processing.

Alternatively, such support may be offered in software at the cost of significant performance.

Additionally, the IP checksum of the packet may need to be updated as well which is similar to the modifications of the IP checksum field for the TTL.
to convey the configuration of the TEH. The settings for the options field are listed in Table II. The next portion of the TEH is the replication/routing information. Depending upon the setting in the options field, each entry may be either 23 bits (3 bytes) in the 5/2 case (5 interface replication bits (i.e. eth0, eth1, .., eth4), 2 QoS transformation bits per interface) or 32 bits (4 bytes) long in the 6/3 case. Each entry consists of a unique identifier (8 bits), the replication information (5 or 6 bits), and the QoS transformation code (if included).

By default, DSMCast assumes that the unique identifier will be the last 8 bits of the lowest IP address of the router. Alternatively, the network administrator may assign a unique ID instead to each router. The re-use of the IP address allows for easy integration from the OSPF/IS-IS topology information and should be sufficiently unique for an individual router in the domain. In the event that a longer unique address is required, an extension bit can be enabled to double the unique ID space to 16 bits. If a unique ID separate from the IP address is used, a complete list of mapping between the IP address and unique ID must be given to all routers in the domain.

The replication field denotes the interfaces on which the packet should be replicated. The order of the interfaces is sorted by the unique ID of the neighbor on the other side of the link corresponding with the interface. The maximum number of interfaces is denoted by the Extended Replication bit of the options field (5 or 6 interfaces, 12 or more with use of the Extended ID bit) [18].

During operation, a core router will locate its entry in the DSMCast header through the ID field. The Rep field will denote how the packet should be replicated (1=Replicate, 0=No Replication). If present, the QoS transformation code denotes how to change the DSCP (and hence the PHB). A complete example of the TEH is given in [18].

C. Variable QoS Extensions

In order to support heterogeneous QoS within the multicast tree, the DS field must be allowed to change as the multicast packet is replicated. In such a heterogeneous QoS tree, the “best” PHB is propagated up the tree towards the ingress router. As a result, the PHB can only “worsen” as the packet passes down the multicast tree. The Variable QoS option adds a QoS Transformation field for each bit of the replication field. In the QoS transformation field, a code is given that determines how the current PHB should be changed. The actual transformation process can be configured as desired by the network administrator. Table III lists an example of an encoding using 3 bits. For example, a bit field of 111 would change an AF10 packet by reducing the DSCP by two loss and two delay classes to AF32. The intricacies associated with heterogeneous QoS management have been investigated in greater detail in one of our current works [19].

D. Tree Construction

The TEH is constructed and maintained by the ingress router of the multicast group. An example of an individual entry in the TEH is given in Figure 3 based on the example domain in Figure 4. It is assumed that each edge router has complete topological knowledge of the entire DS domain (core and edge routers). This information

---

**TABLE II**

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Abbreviation</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>V</td>
<td>Variable QoS</td>
<td>0 - No variable QoS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - Variable QoS present (heterogeneous QoS)</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>Extended Replication</td>
<td>0 - 5/2 bits per entry (24 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Rep/QoS) bits</td>
<td>1 - 6/3 bits per entry (32 bits)</td>
</tr>
<tr>
<td>5</td>
<td>I</td>
<td>Extended ID</td>
<td>0 - Use an 8 bit ID (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - Use a 16 bit ID</td>
</tr>
<tr>
<td>4,3</td>
<td>QoS</td>
<td>QoS Setting</td>
<td>Switch for QoS transformation methods</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>Clear Rep</td>
<td>0 - Leave replication bit present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - Clear replication bits on replication</td>
</tr>
<tr>
<td>1,0</td>
<td>Rsv</td>
<td>Reserved</td>
<td>Unused</td>
</tr>
</tbody>
</table>

---

4The notion of better/worse PHBs depends upon many factors (current network state, rate-limiting, etc.) and is best determined by the network administrator.
is easily available from common intra-domain protocols such as OSPF [20] or IS-IS [21] or by configuration of the network administrator. Note that this information is only the topology of the local DS domain, not the topology of the global Internet nor the resource allocations on each of link of the domain.

As a result of its knowledge of the DS domain, the ingress router can appropriately construct a multicast distribution tree for a specific multicast group and update the tree as egress routers join and leave the multicast group. The tree size is strictly limited by the number of egress routers that wish to receiver traffic and is independent of the number of actual receivers on the global multicast tree. The tree construction algorithm is left to the decision of the network administrator and may be optimized either for cost (shared tree [22]) or QoS (shortest path tree [23]). Although the routing of the tree may be based on network dynamics (due to the fact that the tree is route-pinned inside the packet), such methods are left as a topic for future research.

In addition, it is assumed that there is an underlying scheme for detecting faults or reconfigurations in the underlying domain topology. The issues with fault detection/recovery for a distributed set of edge nodes and stateless core nodes are beyond the scope of this paper but are analyzed in more detail in [24]. While the route-pinned packet does potentially make DSMCast briefly more susceptible to faults, it does offer several significant benefits. First, the route-pinned nature of the tree allows the ingress router to know the maximum resource consumption by a given multicast tree. Since the ingress router must explicitly specify each link for replication and the PHB associated with each link, the resource consumption is available at the router that does the policing in the first place. Second, admission control for the tree is dramatically simplified. Since the ingress router knows the exact makeup of the tree, it can negotiate for the resources of the tree on a whole rather than leaving the tree to be negotiated piece-wise by individual routers along the tree. Whereas resource management in the state-based approach can be complex and difficult, resource management is significantly easier with DSMCast as a result of only the ingress router maintaining resources for the multicast tree.

In addition to fault tolerance, another potential cause for concern lies with the size of the TEH due to the number of routers in the domain. This argument can be
countered in several respects. To start, the size of the header is directly related to the number of core routers in the multicast tree, not the number of egress points nor the number of receivers in the multicast group. Furthermore, the multicast tree is the distribution tree for transport across a single domain, not the global multicast tree. If necessary, the tree may be subdivided into multiple trees to reduce the size of the DSMCast TEH.

IV. EGRESS ROUTER JOIN/LEAVE

The next logical step after providing the underlying transport service is to solve the problem of how join/leave messages are processed. Whereas the state-based approach simply relied on individual nodes propagating join/leave messages towards the multicast tree, the removal of state information and control packet processing from the core leaves a void that multicast control messages cannot cross without intervention.

The first router that will receive a join or leave message must be an edge router due to the DiffServ architecture. At that point, it is up to the edge router to determine where to forward the control packet. In the case of a leave packet, the task is fairly simple as the edge router is already part of the multicast tree for the group and hence knows the ingress router for the group. In contrast, the join process is somewhat more difficult depending upon the underlying multicast model. If the traditional multicast model is employed, the process can be significantly more difficult as the group address \((\ast, G)\) does not imply the location. As a result, the multicast group may be present anywhere in the global Internet, thus necessitating a search mechanism for the multicast group. Several schemes such as MSDP (Multicast Source Discovery Protocol [25]), MBGP (Multicast Border Gateway Protocol) [26] have been developed to address this. The egress join problem can be summarized thusly:

For a join request to group \(G_x\) from downstream node \(R_y\) that arrives at edge node \((E_{Recv})\), determine the appropriate edge node \((E_{Egress})\) to which the multicast join request should be forwarded to and propagated outwards from the DS domain such that the QoS constraints of the join request are met.

However, if the location of the group is tied to an actual source IP address \((S, G)\), the problem of location vagueness disappears and the problem simply reduces to basic IP routing. For multicast models such as SSM and CBT (Core-Based Trees) where the location is required, the complexity of multicast routing decreases dramatically. In fact, the additional complexities of traditional IP multicast offer little benefit that cannot be accomplished by SSM. A recent study on multicasting noted that very few applications take advantage of the multi-sourced potential of traditional IP multicasting [13]. As a result, the join/leave protocol for DSMCast is optimized for an SSM environment. However, as legacy applications may still require non-specific source support, extensions for traditional IP multicasting to DSMCast are included as well.

A. Egress Join - SSM

When a join request arrives at an edge router, the request will include a source/group identifier \((S, G)\). The edge router must then forward the packet to the appropriate ingress node for multicast source. Due to the fact that IP routing provides only for the next hop to a given destination IP, ascertaining the ingress point for an arbitrary IP may be difficult if not impossible. The solution to this problem is to simply tunnel [10] the join request towards the source of the group. As a result, the egress router can rely on standard IP routing rather than a special routing mechanism. In the tunnel header, a special identifier will be recognized by the ingress router. Upon receiving the packet, the ingress router would intercept and process the packet rather than forwarding the packet.

Similar to how DSMCast provides support for both SSM as well as traditional IP multicasting, DSMCast also provides flexibility in how the egress router joins the multicast group. Rather than forcing a router to explicitly specify a PHB with the initial join, an egress router may first probe for the sender-driven QoS and then select a PHB. Hence, the egress router (i.e. the new router joining the group), may use one of two types of join requests, an absolute join request or a dynamic QoS join request.

1) Absolute Join: The absolute join process operates using a simple request/response message exchange. When an egress router wishes to attach itself to the multicast tree, it tunnels a Join-Request message towards the source of the multicast group. The ingress router intercepts the Join-Request message and process the message. The Join-Request message includes the original inter-domain join request, the IP address of the new egress point, and the requested PHB for traffic to the new egress point.

Upon receipt of a Join-Request message, the ingress router may need to perform admission control by contacting the BB (Bandwidth Broker) of the domain if one is present. Once the new resource allocation has been approved, the ingress router updates the multicast tree by
computing a new TEH and updating its shaping/policing mechanisms if necessary. The ingress router responds with a Join-Ack message to the new egress router. From this point onwards, all subsequent data packets sent out from the ingress router will be distributed to the new egress router.

2) Dynamic QoS Join: An alternative approach is for the new egress router to select a PHB based on network dynamics rather than specifying an absolute PHB. This approach is especially applicable for dynamic scheduling schemes such as relative DiffServ [27] or for adapting to other reservation mechanisms such as RSVP [28]. In the dynamic QoS join case, the egress point first tunnels a Bid-Request message towards the source of the multicast group. Similar to the absolute join, the ingress router intercepts the Bid-Request message and processes the message. In this case, the ingress router responds with a special Bid-Probe message towards the egress router. The packet is routing explicitly using the DSMCast TEH using a multicast control group address.

As the packet crosses the domain, various statistics such as average queue size, average queue delay, and average loss rate are included for a fixed subset of classes. This scheme fits well within the DiffServ notion of a sender-driven QoS giving the egress router an approximate picture of the various performances of each class on the paths the actual data packets would travel on. Once the egress router receives the Bid-Probe message, it selects a PHB to join to the group with and follows the absolute join protocol. Note that this procedure may be invoked periodically to offer adaptive QoS.

Figure 5 summarizes the dynamic QoS join routine. In the first step, $E_{RCVR}$ tunnels a Bid-Request message towards $S$. The Bid-Request message is routed via traditional IP routing and is intercepted by $E_3$. $E_3$ processes the message and replies with a Bid-Probe message back to $E_{RCVR}$. As the Bid-Probe message travels back to $E_{RCVR}$, various stats for classes are gathered at each core node along the path. The figure denotes only one of the pieces of core information that could be gathered. Upon receiving the Bid-Probe message, $E_{RCVR}$ would dispatch a Join-Request message to $E_3$ since it now knows the ingress node for the multicast group. $E_3$ would then add $E_{RCVR}$ to the multicast group and respond to $E_{RCVR}$ with a Join-Ack. From that point onwards, new multicast data packets to the group would also flow to $E_{RCVR}$.

Although the dynamic join approach offers users additional flexibility, the flexibility comes at an additional cost. First and foremost, the service requires additional complexity at core routers. Second, the burstiness of traffic may reduce the utility of the results if the results are not weighted properly. Third, the approach increases the join time and may be best applied after an initial join is complete. Finally, such a scheme would need to be rate-limited to prevent unnecessary consumption of router CPU resources.

B. Egress Join - Traditional IP Multicast

As stated earlier, the location vagueness of traditional IP multicast prevents an egress router from knowing the location of a multicast group. Thus, an egress router cannot simply tunnel towards a source since there is not a source correlated to an traditional IP multicast group. The proposed solution for DSMCast is to simply expand the bid/probe routines of DSMCast.

When a new egress router to receive traffic for a multicast group, the egress router asks all of the potential edge routers using a Bid-Request message. Rather than sending out separate tunnels, the egress router multicasts via a special control tree that is a static multicast tree containing all eligible edge routers. Upon receiving a Bid-Request message, each edge router responds with a Bid-Probe message towards the new egress router. Depending upon the number of edge routers responding, the probe (information gathering) function of the Bid-Probe message may be disabled in the case of traditional IP multicast. If necessary, the edge routers may need to invoke inter-domain routing in order to locate the multicast group. The Bid-Probe message is not sent until the edge router has located the multicast group.

Since any edge router could be a potential candidate for the ingress router, the cost of the tree must be included as well. For instance, in a large domain it may be desirable to choose an alternative ingress router.
even though the multicast group is already present in the domain at another router. At the new egress router, the various Bid-Probe messages are collected and processed until either a timer has expired and/or a sufficient threshold of edge routers have responded. Since the traditional IP multicast definition allows for multi-sourced groups (i.e. multiple ingress routers), the egress router will send a Join-Request to all edge routers that responded as being part of the multicast group. The Join-Request message may be conveyed using a subset of the control tree. In addition, the egress router may choose not to request an acknowledgement from each of the on-tree edge routers, thus reducing the control message overhead. To leave the traditional IP multicast group, a Leave-Request message should be multicast to the other on-tree edge routers.

1) Shared Trees: One of the variations of multi-sourced groups in traditional IP multicast is the concept of a shared tree. Rather than each source having its own group address (known as many-to-many multicasting), a single group address is used to offer a consistent QoS or to share bandwidth for intra-group communications. Examples of such applications that might utilize shared trees include audio-conferencing where only one individual is speaking at a time.

From a routing perspective, the goal of the shared tree is to minimize the total consumed bandwidth while still satisfying the QoS of all receivers. In contrast, the shortest path tree prioritizes QoS by using the shortest path at the potential detriment to tree cost. Although shared trees are an interesting theoretical concept, the general trend is away from the complexities of shared trees and traditional IP multicast [13]. Hence, explicit support for working with globally shared trees is not provided. However, the large body of research on shared tree construction and maintenance can still be applied on a domain-wise basis for optimizing the domain distribution tree [29].

V. OTHER ISSUES WITH DSMCAST

With any new architecture for the Internet, the issues for deployment must also be examined as well as any future concerns. Several of the issues that must be considered with DSMCast include IPSec, initial deployment requirements, and IPv6. In addition, this section also considers other related work, enhancements to DSMCast, and implications beyond multicasting.

A. IPSec

The use of IPSec [30] does not present a problem with normal DSMCast operation and DSMCast with tunneling since all information related to the DSMCast header is removed at the edge of the DS domain. However, the use of IPSec and DSMCast can cause problems when DSMCast is used with an adaptive DS field. In its default mode, IPSec does not include the DS field in its cryptographic calculation. Thus, the default mode of IPSec does not conflict with the use of heterogeneous QoS. However, the IPSec tunnel mode does encrypt the IP header, thus representing a problem with heterogeneous QoS. This problem is similar to the dilemma faced by inter-domain edge routers that do packet remarking [2].

B. Initial Deployment Requirements and IPv6

In order to deploy DSMCast for a given DS domain, all routers in the domain must be DSMCast-enabled. Although it would be possible to introduce a tunnel extension to DSMCast [31], the cost of such support is quite high. An intermediate step for DSMCast is to offer software support until hardware support is available or to employ an edge-only solution.

Since DSMCast is entirely transparent to IP routing (besides the changes to the DS field for variable QoS), DSMCast will not have any significant inter-operability issues with IPv6. The only potential issues lie in the unique ID field of DSMCast and the possibility for variable headers in IPv6. Whereas the last 8 bits of the IPv4 address are a viable approach, such may not be the case for IPv6. Due to the fact that the MAC address may simply become the last 6 bytes of the IPv6 address, it is quite likely that conflicts will emerge. As a result, it may be necessary for the network administrator to use a separate unique ID and to provide a mapping table to all routers in the network. The second potential issue lies with the possibility of multiple headers IPv6 headers. In such a case, a CAM or FPGA approach may be more difficult to use and thus, additional information may need to be included in the DSMCast header to optimize the search time in a software-based network processor.

C. Small Group Multicast

Small Group Multicast [16], a proposal under consideration by the IETF, uses a similar encapsulation-based technique to achieve traditional IP multicast. Under Small Group Multicast (SGM), the multicast tree is embedded inside the packet and sent from the source to a given SGM router. At the SGM router, the encapsulated tree information is appropriately partitioned and the new packet(s) are transmitted onto the downstream links.

DSMCast differs from Small Group Multicast in several key areas. First, DSMCast provides transparent
support for traditional IP multicast across the DiffServ domain. Whereas SGM requires support at the source, DSMCast already works within the existing IP multicast architecture. Second, DSMCast core routers are significantly less complex than SGM-enabled routers. Under DSMCast, no modifications are made to packet length (except for the initial encapsulation) thus requiring core routers to only replicate DSMCast packets and make only minor changes to the contents of the packets. In contrast, the SGM header changes at each router due to partitioning according to the addresses encapsulated within the packet. Although partitioning may reduce the downstream bandwidth, it adds a significant computation cost since SGM routers must partition and modify the size and contents of both the SGM and IP headers of packets.

Third, the DSMCast architecture takes special advantage of the underlying DiffServ architecture in order to provide both group size and group addresses (unique multicast IDs) scalability. Whereas SGM encapsulates the entire end-to-end multicast tree, DSMCast encapsulates only the multicast tree for a given DS domain. Thus, the use of SGM is limited to small groups since the IP addresses of all receivers must be included in the SGM header. In contrast, DSMCast is only restricted by the number of core routers in a multicast tree in a single DS domain, not by the end-to-end multicast tree nor by the number of receivers. Thus, the scalability of DSMCast is dependent upon the size of the domain, not the size of the multicast group which allows DSMCast to serve both large and small groups.

D. Enhancements to DSMCast

Although DSMCast is targeted towards multicasting, the use of DSMCast can have significant impacts in other areas including QoS management, unicast QoS, and fault detection.

1) QoS Management: To start, the Bid-Request/Bid-Probe messages can be adopted for more than simply determining the appropriate PHB for the egress router for multicasting. The Bid-Request and Bid-Probe sequence can be used for adaptively meeting the QoS for unicast aggregations of traffic as well. Bid-Probe messages can be sent periodically to other edge routers, regardless of whether or not the Bid-Probe message was solicited or not. Edge routers could then use the information provided by the Bid-Probe messages to dynamically adapt their shaping/policing or even routing patterns of incoming packets. This approach allows edge routers to be proactive towards QoS rather than waiting for a QoS violation to occur and reacting to the violation.

2) Unicast QoS: The introduction of heterogeneous QoS for multicasting poses an interesting question: would heterogeneous QoS be useful for unicast connections? For instance, would it make sense to offer a near-EF ( Expedited Forwarding ) service instead of absolute AF or absolute EF services?

Suppose that a customer wants better QoS than the AF2x service the company is currently using. Rather than upgrading the traffic to AF1x, selected hops/links could be upgraded to AF1x. An example of such a case would be the traffic receiving AF1x-AF2x-AF2x-AF1x treatment instead of strictly AF1x or AF2x. A service provider would be able to offer a gradient of QoS rather than absolute QoS from edge-to-edge. For underlying schedulers such as relative DiffServ [27], the ability to adaptively manage PHBs over different links in the core offers even greater benefit.

The DSMCast architecture can be adapted for unicast connections by placing only the route to only one egress node in the TEH. Although such a scheme would offer greater flexibility, the cost would not be insignificant. The implications of variable QoS for unicast connections have the potential for dramatic implications in DiffServ and is an open topic for future research.

3) Fault Detection: The final aspect where DSMCast can offer additional benefit is in the area of fault detection. Although the route-pinned nature of DSMCast makes DSMCast slower to react to faults due to delays in network updates, the route-pinned nature of DSMCast allows edge routers to verify the health of link and routers themselves. Rather than relying on CPU intensive link state messages, the edge routers can assist the link state protocol while monitoring the QoS of the network [24].

VI. THEORETICAL AND SIMULATION STUDIES

In this section, the performance of DSMCast is analyzed for both theoretical as well as simulated performance. In the theoretical analysis, DSMCast is examined for its overhead versus the edge-based approach, the state-based approach, and other factors such as network size. The simulation studies examine the performance of DSMCast versus the other two approaches (state-based and edge-based) regarding a variety of parameters including group size, packet size, receiver heterogeneity, and group dynamics.
A. Theoretical Studies

1) Examining the Cost of State in the Core: As mentioned earlier, the primary benefit of DSMCast is that it reduces the cost of multicast state information to make multicasting more scalable. The justification for the additional overhead of DSMCast is that the cost of maintaining state information is greater than the cost of the additional bandwidth imposed by the stateless approach. Formally, this relationship can be summarized as follows:

\[ \text{Cost}_{\text{State}}[T_E + T_C] + \text{Cost}_{\text{BW}}[(T_E - 1 + T_C) \times P_S] > \]
\[ \text{Cost}_{\text{State}}[T_E] + \text{Cost}_{\text{BW}}[(T_E - 1 + T_C)(P_S + OH)] \]

where \( T_E \) is the number of edge nodes in the tree, \( T_C \) is the number of core nodes in the tree, \( P_S \) is the size of packet, \( OH \) is the per-packet overhead introduced by stateless core routing, \( \text{Cost}_{\text{State}}[\cdot] \) is a function denoting the cost of state, and \( \text{Cost}_{\text{BW}}[\cdot] \) is a function denoting the cost of bandwidth. The equation can be reduced as follows:

\[ \text{Cost}_{\text{State}}[T_C] > \text{Cost}_{\text{State}}[T_{\text{Ingress}}] + \]
\[ \text{Cost}_{\text{BW}}[(T_E - 1 + T_C) \times OH] \]

which essentially states that the cost of the state information in the core is greater than the per-packet overhead introduced on the entire multicast tree for the domain. This is a valid assumption for several reasons. First, the cost of bandwidth has dramatically decreased over the last few years and new advances are continually increasing the bandwidth available [32]. Second and most importantly, although multicast may or may not make up a large portion of traffic, the number of unique multicast groups (especially with SSM) will explode dramatically. Although a router may be able to manage the impact of millions of entries in the routing table, the management of the complex per-group state information (timers/etc.) will simply not be scalable. Thus, support for a stateless core is critical to the successful deployment and wide scale acceptance of multicasting beyond MBone [5].

2) Overhead of DSMCast: The overhead of the DSMCast TEH is dependent upon the number of non-leaf routers in the domain that are on the multicast tree. For each router where replication takes place, an entry will be present in the TEH. The cost of the TEH can be summarized below:

\[ \text{Cost} = \text{NumEntries} + \text{Options} + k \times \text{EntrySize} \]

where \( \text{NumEntries} \) is the field giving the number of entries in the TEH (1 byte), \( \text{Options} \) is the options field (1 byte), \( k \) is the number of entries, and \( \text{EntrySize} \) is the size of each entry. The cost of each entry is given below:

\[ \text{EntrySize} = \left\lceil \frac{ID + \text{Rep} + \text{Rep} \times \text{SizeQoS}}{8} \right\rceil \]

where \( ID \) is the size of the unique ID (8 or 16 bits), \( \text{Rep} \) is the maximum number of interfaces on any node in the domain (5 or 6 bits), and \( \text{SizeQoS} \) is the size of the QoS transformation field (2 or 3 bits). In addition, each entry is forced to byte boundaries to ensure ease of hardware support. The 5/2 (\( \text{Rep} = 5, \text{SizeQoS} = 2 \)) and 6/3 settings deliver sizes of 23 bits (rounded up to 3 bytes) and 32 bits (4 bytes) respectively.

3) Tradeoff versus the edge-based approach: In order for DSMCast to offer a significant benefit, it must offer a better performance than the simplest edge-based approach, ingress-branching. The ingress-branching solution allows only the ingress node to replicate packets and then such packets are tunneled to the appropriate egress routers. Although less efficient than DSMCast, this scheme satisfies the three conflicts between DiffServ and multicasting. In fact, the scheme on a global scale can still offer performance benefits as tree-wise aggregation still occurs at the edge of the domain. Thus, DSMCast must justify its additional complexity by offering a significant savings over this simple scheme.

In short, DSMCast offers performance a performance improvement when shared links occur in the multicast tree. A shared link occurs when more than one egress
point shares a branch of the multicast tree (see Figure 6). The number of shared links at a given node \(x\) in the multicast tree can be defined as:

\[
SL_x = \sum_{i=0}^{C} SL_i
\]

where \(C\) is the total number of children at the node and \(SL_i\) is the number of shared links at each node in the multicast tree. The total number of shared links at a given node is the aggregation of all of the shared links of its children. With each shared link, DSMCast saves the cost of an entire packet transmission versus the separate tunnels of the ingress-branching approach. However, whereas the ingress-branching approach incurs only a relatively small penalty for the tunneling overhead, the DSMCast overhead is significantly greater. The cost of the ingress-only approach can be summarized below:

\[
Cost_{\text{Ingress}} = \sum_{x=0}^{R} D(E_{\text{Ingress}}, E_x) \times (P_S + T)
\]

where \(R\) is the number of egress points (receivers), \(D(E_{\text{Ingress}}, E_x)\) is the distance (in hops) between \(E_{\text{Ingress}}\) and \(E_x\), \(P_S\) is the size of the packet, and \(T\) is the cost of tunneling. For the same tree, the cost of DSMCast is as follows:

\[
Cost_{\text{DSM Cast}} = (N - 1) \times (P_S + TEH)
\]

where \(N\) is the total number of nodes in the tree, \(P_S\) is the size of the packet, and \(TEH\) is the size of the DSMCast Tree Encapsulation Header. The tradeoff between the two approaches can be captured as the following:

\[
\sum_{i=0}^{N} SL_i \times (P_S + T) \geq (N - 1)(TEH - T)
\]

where \(TEH\) is the size of the of DSMCast header (dependent upon the size of the tree), and \(T\) is the cost of tunneling (Minimal Encapsulation - 12 bytes [10]). The equation informally states that when the number of shared links is sufficient, the packet will be replicated enough additional times in the ingress-branching case to offset the cost of the Tree Encapsulation Header on the entirety of the multicast tree.

Figures 7 plots the tradeoff at which DSMCast offers a benefit versus ingress-branching. In the graph, it is assumed that 50% of the edge nodes are part of the tree and that 50% of the core nodes are part of the tree as well. Although the tradeoff of 450 bytes (total packet size) may seem high for the network of 48 nodes (32 core + 16 edge nodes) and 3 shared links, consider the relatively low numbers of shared links that were plotted. For even the small example listed in Figure 6, there were already 5 shared links present for only 3 receivers. In fact, the simulation studies in the next section frequently found shared links equal or greater than those plotted here.

4) Tradeoff versus state-based: Figures 8 shows the overhead of DSMCast normalized to the performance of traditional IP multicast (state-based approach) on two different network sizes. Once DSMCast passes a sufficient packet size, the costs of DSMCast become
negligible. In fact, without variable QoS enabled, the costs of DSMCast is reduced even further to the point where the overhead introduced by DSMCast would be less than the performance penalty of enabling state-based IP multicast on many multicast routers.

VII. SIMULATION STUDIES

The performance of DSMCast versus other solutions was analyzed through extensive simulation studies. The simulation studies were conducted using the ns-2 simulator [33] and the GenMCast extension for multicasting [34]. The performance of DSMCast was compared to the performance of the ingress-branching solution and a generic version of PIM-SM [35]. The parameters for the simulations were as follows:

- The network topology consisted of 32 core nodes connected with an average node degree of 3. 16 edge routers connected by at least one core router in between two edge routers. Although not shown, NSFNet was also used as a realistic topology for comparison in [18].
- Each edge router was the source for two SSM multicast groups (32 total groups).
- Each multicast group produced CBR traffic with an exponentially distributed packet size and packet inter-arrival time. Unless otherwise noted, the mean for the values were 400 bytes and 100 ms.
- The starting number of egress points for each simulation was 160 egress points (average group density of $\frac{160}{32} = 5$ egress points) with the egress points randomly distributed between the various groups.
- The events that constitute group dynamics (join/leave) were uniformly distributed with a mean inter-event time ranging from 150 ms to 250 ms. The probability for a join or leave was 0.5 for each.
- The network was assumed to have sufficient buffer space and bandwidth such that packet dropping does not occur. This is done to ensure an adequate comparison between the various solutions without bias by packet drops.
- Additional parameters are summarized in Table IV. The simulations were conducted for a single domain scenario of varying random network topologies and varying QoS distributions (uniform and non-uniform). The models were evaluated on basis of the normalized cost versus the state-based approach (PIM). The normalized overhead gives an indication of the relative bandwidth cost of the model in a loss-free network. For example, an overhead of 1.2 denotes that an algorithm requires 20% extra bandwidth. For each of the figures, the following values are graphed:

- **PIM-SM**: A generic version of PIM-SM is used as a benchmark for the simulations. The results of the simulations are normalized to the cost of PIM-SM.
- **Ingress-branching**: The ingress-branching approach only replicates packets at the ingress of the domain. Packets are then tunneled across the domain using Minimal Encapsulation [10].
- **DSMCast-All**: The DSMCast-only version relies exclusively on DSMCast regardless of packet size. Hence, if the packet size is too low, the performance may be worse than the ingress-branching solution.
- **DSMCast-Adaptive**: The DSMCast-adaptive version alternates between either DSMCast or ingress-branching depending upon the optimal solution. At the worst case, the performance of this solution should be the same as the ingress-branching solution.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Bandwidth/Delay</td>
<td>1.0 Gb/s, 5 ms</td>
</tr>
<tr>
<td>PHBs</td>
<td></td>
</tr>
<tr>
<td>Uniform</td>
<td>AFxy (x=1...3,y=1.4)</td>
</tr>
<tr>
<td></td>
<td>x(25%,25%,25%,25%)</td>
</tr>
<tr>
<td></td>
<td>y(33%,33%,33%)</td>
</tr>
<tr>
<td>Non-Uniform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x(10%,10%,40%,40%)</td>
</tr>
<tr>
<td></td>
<td>y(10%,30%,60%)</td>
</tr>
</tbody>
</table>

TABLE IV

DSMCast NS PARAMETER LIST
A. Effect of Packet Size

Figure 9 shows the performance of the three schemes as the mean packet size is varied from 100 to 850 bytes. As the average packet size increases, the overhead of DSMCast decreases due to the fact that the overhead of DSMCast becomes a smaller percentage of the overall packet size. This assertion is supported by the results shown in Figure 10. Figure 10 shows the selection choices of the adaptive DSMCast model. As the packet size increases, the tradeoff point of DSMCast falls, thus causing the adaptive algorithm to choose DSMCast more than separately unicasting the packets (ingress tunneling). In fact, even at a relatively low average packet size (100 bytes), the tradeoff for DSMCast is sufficiently small enough to justify almost 80% of the packets being sent by DSMCast. Furthermore, we believe the multimedia nature of most group-based QoS applications will on average have packet sizes well beyond those simulated, thus reducing the overhead to well below 15% more than the state-based approach.

B. Effect of Group Size

Figure 11 examines the performance of DSMCast further as the number of egress points is varied. An increase in the number of egress points implies that additional downstream receivers wish to subscribe to the multicast group. The number of egress points are kept fairly sparse, ranging from less than 25% of the total egress points (average 3 egress points per group) to 50% of the total egress points (average 8 egress points per group). Whereas the overhead of DSMCast does rise, it can be directly attributed to the additional links that are present in the tree.

Figure 12 breaks down the cost of the per-packet overhead for both the ingress-branching (fixed tunnel cost [10]) and DSMCast. The percentage represents the overhead as a percentage of the data packet size used by PIM. Due to the nature of the random network, each additional egress point may potentially increase the number of core nodes in the tree, hence increasing the size of the DSMCast TEH.

In contrast, the introduction of additional egress points does offer several benefits to DSMCast versus ingress-branching. As the number of egress points increases, the chance for additional shared links increases as well, thus driving down the tradeoff for when DSMCast becomes practical. Figure 13 plots the increase in shared links
as the number of egress points increases. The number of shared links increases faster than the number of egress points as a single egress point will typically utilize several shared links. In fact, as the group density increases, shared links will increase at a faster rate due to the presence of other egress points already being on the multicast tree. The increase in shared links produces a noticeable effect on the adaptive DSMCast model as the additional shared links reduce the tradeoff and hence, cause the adaptive model to select DSMCast more frequently (see Figure 14).

A second way to analyze the tradeoff between ingress-branching and DSMCast is to examine the number of packets transmitted onto the network. Whereas both DSMCast and PIM minimize the number of packets due to the nature of the multicast tree, the ingress-branching method introduces one packet per egress point. As a result, the number of packets dramatically increases versus DSMCast and PIM. Figure 15 plots the average number of multicast packets per second transmitted onto each link in the entire network. The number of packets employed by PIM is the same as the number of packets employed by the non-adaptive version of DSMCast (DSMCast-All). The increase in packets can be directly tied to the increase in shared links as show earlier in Figure 13.

C. Overhead of the State-based Approach

While the earlier figures showed the performance of DSMCast relative to PIM and traditional IP multicasting, Figure 16 and 17 examine the state cost associated with the previous simulations. In the simulations, the amount
of state information in the network represents over 50% of the total state information supported by the network. In fact, unless a sizeable amount of egress points are achieved in the group, the core state information is likely to dominate the memory/storage/maintenance requirements for multicast routers in the domain.

D. Effect of Group Size - Non-Uniform PHBs

Figure 18 plots the performance of the models as non-uniform traffic is applied. The performance versus the original uniform traffic model does not change as the size of DSMCast is dependent only upon the number of nodes in the tree, not the QoS requested by the egress points. Although cases may arise where the QoS transformation code may be insufficient to meet the needs of the requested QoS, the impact on the overhead of DSMCast would be negligible.

E. Effect of Network Size

Finally, Figures 19 and 20 show the effect on overhead as the size of the domain is varied. Figure 19 shows the impact of additional edge nodes (ranging from 32 to 288) with a base network of 32 core nodes. Beyond an increase in group density (average 10 egress points per group), all other settings are identical to the earlier simulations. Most notably, an increase in the number of edges has little or no impact on the performance of DSMCast. The only noticeable impact point occurs when DSMCast is forced to employ extended addressing (16 bit address) once the number of total nodes in the domain
exceeds 255 nodes. This performance trend shows that DSMCast is clearly scalable in terms of the number of edge nodes in the domain.

Figure 20 further examines the performance of DSMCast with even larger network sizes. In Figure 20, the number of edge nodes is fixed at three times the number of core nodes (domain sizes of 128 to 932 total nodes). The number of egress points is kept at an average of 10 egress points per group to better illustrate the impact of the domain size. Unlike the edge ratio, the actual domain size can have a profound impact on DSMCast. With a larger domain, the tendency will be towards larger trees (in terms of hops, not necessarily egress points), thus increasing the cost of DSMCast. The relatively small group size places an ever decreasing cap on the benefits of multicasting due to the fact that a larger domain will make the tree itself more sparse and hence less beneficial. However, the adaptive version of DSMCast still offers a relatively solid gain despite the sparsity of the multicast tree with a relatively stable overhead. Thus, we believe adaptive DSMCast can offer the benefit of a stateless core while still being scalable to large networks.

VIII. SUMMARY & CONCLUSIONS

In this paper, the DSMCast architecture was proposed and analyzed through both simulation and theoretical studies. Despite the fact that DSMCast incurs per-packet overhead due to the encapsulation header, it is able to operate reasonably within the cost of the state-based approach. In addition, DSMCast can offer the performance while dramatically reducing the state requirements of the network. Although DSMCast does need hardware acceleration to avoid impacting the router CPU, the performance benefits of multicasting and the reduction of state information more than offset the cost of the hardware. Unlike simply extending traditional IP multicasting, DSMCast satisfies the three conflicts between DiffServ multicasting while incurring a negligible overhead. Thus, DSMCast can offer a viable approach for DiffServ multicasting.

REFERENCES