QoS-Aware Multicasting in DiffServ Domains[*]

Zhi Li and Prasant Mohapatra
Department of Computer Science
2063 Engineering II
University of California at Davis
Davis, 95616, CA
E-mail:{lizhi, prasant}@cs.ucdavis.edu.

ABSTRACT

QoS-aware multicasting is becoming more and more desirable with the expanding usage of group-based applications, especially those involving multimedia objects. Until now, most of the proposed QoS-aware multicasting routing protocols adopt per-flow based resource reservation. Although these schemes can be adopted in integrated services (IntServ) Internet, they are not suitable for more scalable Differentiated Services (DiffServ) Internet. A new QoS-aware multicast routing protocol called QMD is proposed for DiffServ environments in this paper. Based on the design philosophy of DiffServ, the complex multicasting control plane functionalities are removed from the core routers. In addition, for each multicast group, only a limited set of on-tree routers (termed as key nodes) maintain multicast routing states and forward multicast data traffic. The key nodes of a multicast group uniquely identify a QoS-satisfied multicast tree connecting the group members. Although the other on-tree routers between any two key nodes do not maintain any multicast routing states and QoS reservation information, the group members' QoS requirements can still be satisfied. Through simulation experiments based on both random and real intra-domain topologies, we have also demonstrated that QMD can provide higher QoS-satisfaction rate while maintaining the simplicity of core routers.

Keywords

Differentiated Services (DiffServ), Multicasting, QoS-aware Multicast Routing, QMD

1. INTRODUCTION

With the phenomenal popularity of Internet, a broad group of network applications have been deployed over the Internet. More and more of these applications, such as multi-party video conferencing, distance learning, files distribution and caching, involve transmitting information using multi-point connections. The popularity of these applications desire multicast service support from the Internet.

IP multicasting [15] has been proposed to provide efficient one-to-many (many-to-many) data delivery services. Many applications that benefit from multicasting services are also QoS-sensitive. These applications require the underlying network to provide QoS support. In the past years, many efforts have been dedicated to QoS provisioning in the Internet [31]. Among them, the Internet Engineering Task Force (IETF) has proposed two basic QoS provisioning techniques to manage and reserve the QoS resources in the Internet: per-flow-based Integrated Services (IntServ)[10] and aggregation-based Differentiated Services (DiffServ) [8].

In the past years, lots of QoS-based multicast routing protocols[18][17][22][29] based on per-flow reservation have been proposed in the literature. Even though these protocols can perform very well in IntServ environment, they cannot be easily deployed in DiffServ domains. In DiffServ, the packets are marked with different service levels at the edge routers, and then the core routers provide different data forwarding services based on the codepoints carried by the packets. Using this approach, DiffServ releases the core routers from maintaining per-flow QoS reservation states. This idea of scalable QoS-support has attracted many researchers' attention since it was proposed. However, most efforts have been directed to the unicast support while only a few attempts have been reported on supporting multicast operations within DiffServ domains. Several reasons determine that providing QoS-aware multicasting in Diffserv domains is a nontrivial work. First, DiffServ was proposed to achieve scalability by aggregating the traffic to decrease the amount of routing state maintenance in core routers. However, for multicasting protocols, every on-tree router needs to maintain per-group data forwarding state. With the increase in the number of multicast groups, the large amount of state maintenance and related processing work needed could overwhelm the core routers (memory and CPU cycles), which conflicts with the design goals of DiffServ. Second, DiffServ only requires edge routers (or together with bandwidth brokers) to make admission control and maintain resource reservation information. The existing QoS-aware multicast routing protocols do not distinguish the functionalities between edge routers and core routers. Both edge routers and core routers perform the same functions in regular multi-

[*]This research was supported in part by the National Science Foundation through the grant CNS-0240502. A preliminary version of this paper was presented at the Global Internet Symposium, GLOBECOM 2002.

cast routing protocols: processing control messages, making admission control, maintaining multicast routing tables, and forwarding the data traffic. Third, when the regular IP multicasting protocols are deployed within the DiffServ domains, the Neglected Reservation Subtree problem (NRS)[9] also becomes an issue. The NRS problem occurs because of the branching of trees in the core routers within a domain. The amount of outgoing traffic from a domain may exceed the incoming traffic to the domain and consumes additional network resources. This additional traffic cannot receive the desired QoS while adversely affecting other existing traffic flows. Finally, within a multicast group, different group members may have different QoS requirements. How to provide heterogeneous QoS support for multicast group members is an issue that needs further consideration [27].

In this paper, we propose a scalable technique called *QoS-aware Multicasting for DiffServ* (QMD). When a single-source multicast group (formed by CBT[3], SSM[6], or PIM[16], etc.) passes through a DiffServ domain, it usually takes the form of multicasting among edge routers (border routers). Usually, one edge router is the multicast source for other group members within a DiffServ domain. Our goal is to construct a scalable QoS-satisfied multicast tree connecting all of these on-tree edge routers. Similar to DiffServ, in QMD, we separate the control plane functions from data plane functions. The edge routers and the Bandwidth Broker (BB) handle most of the multicast control plane functions: processing join or leave events, searching QoS-satisfied branches, making resource reservation, etc. The core routers only maintain minimal data forwarding states. Using the proposed QMD-DIJKSTRA algorithm, QMD can find a QoS-satisfied branch (identified by a list of *key nodes*, a subnet of on-tree routers) connecting the new member to the multicast routing tree. A set of key nodes can not only identify a multicast routing tree, but also provide predictable QoS services to group members. The multicast traffic can then be recursively transmitted between the key nodes to the receivers. Though other on-tree nodes do not maintain any multicast routing states and the QoS resource reservation information, the multicast traffic can still obtain predictable QoS according to the DiffServ codepoints.

A series of simulations have been carried out to evaluate the performance of QMD. The simulation results have shown that QMD can greatly decrease the core routers' workload in terms of processing multicast control messages and maintaining multicasting routing states. The success ratio, which quantifies the probability of finding the QoS-satisfied paths for new members, is higher in the case of QMD. The basic idea of decoupling the processing of data and control messages in DiffServ domains is shown to have positive impact on the performance of QMD.

The rest of the papers is organized as follows. The related works are discussed in Section 2. In Section 3, we introduce the basic idea of QMD. The key nodes searching algorithm, QMD-DIJKSTRA, is introduced in Section 4. We evaluate the performance of QMD through simulations in Section 5, followed by the concluding remarks in Section 6.

2. RELATED WORK

The multicasting data plane implementation of our work shares the idea of "recursive unicast" with REUNITE[25] and HBH[14]. In regular "recursive unicast", only the routers at branching nodes maintain multicast forwarding states. The multicast traffic is recursively unicasted between the branching nodes without bothering other on-tree nodes. All other on-tree nodes still need to maintain the multicast tree control information and process the join and leave events. However, in QMD, the multicast traffic is unicasted between the key nodes which can guarantee the receiver's QoS requirement at the same time. Other on-tree nodes do not maintain any control states. Besides, QMD releases the core routers from processing large amount of multicast control messages. The control plane operations are handled by border routers and bandwidth broker (BB). The design of QMD makes it DiffServ-friendly and relieves multicast service's security and pricing concerns.

Bless [9] first discussed the issue of providing QoS-aware multicast services within DiffServ domains. However, the goal of his work is to cope with neglected reservation subtree problem (NRS) problem by decreasing the QoS service level of the replicated packets. Yang and Mohapatra also proposed an approach called DAM to solve the NRS problem [32]. It has three novel features: weighted traffic conditioning (WTC), receiver-initiated marking (RIM), and heterogeneous DSCP header encapsulation (HDE). The approach solves the NRS problem by sacrificing efficiency in terms of bandwidth usage.

In parallel to our work [23], several other approaches have been proposed to provide DiffServ multicasting routing services. In Striegel and Manimaran's encapsulation-based approach[28], a DSMCast header including the multicasting tree information is added to each multicasting packet at the ingress router. The core routers can duplicate and forward the data traffic based on the DSMCast header information. This method keeps the core router simple but incurs high bandwidth overhead. Gupta and Ammar [19] used the limited branching techniques to achieve scalable multicasting services in DiffServ domains: M-DS. It has two techniques: edge-router branching technique and limited-core branching technique. This method does not incur any overhead on the data packets and requires minimum changes to the existing routers. However, there will be more duplicate multicast traffic and incur extra overhead. Similarly, EBM [26] also leverages the intelligence of edge routers and the proposed Multicast Broker (MB) to provide scalable DiffServ multicasting service. The multicasting data transmission is based on Edge Cluster Trees (ECTs), which ensures that the multicast branching points are only located at the edge routers. QUASIMODO [7] mainly focuses on the providing desirable QoS for new group members. It is based on two approaches: 1) A probe-based approach called "GRIP" to verify the resource availability of a new path; 2) Using a special DiffServ marking value to maintain the replicated packet QoS.

Several works providing QoS-based multicast routing service have been proposed [11][12][17]. However, all of them require the on-tree nodes to process and control per-flow based QoS resource reservation, which is not scalable in DiffServ domains.

In [5], the authors proposed a method that can reduce the amount of states the core routers need to maintain in IntServ environments. It also differentiates the functions between edge routers and core routers. This approach can be applied to both unicast and multicast traffic. QMD can reduce not only the QoS service states (benefited from DiffServ), but also the multicast routing state maintenance at the core routers. In addition, it can also deal with NRS, security, and pricing concerns of multicasting services within DiffServ domains.

Different traffic engineering techniques and algorithms have been proposed to efficiently utilize network resources, accommodate data traffic and provide service quality, such as multi-path routing, traffic splitting, constraint-based routing[4]. In contrast to the existing QoS-aware multicast routing approaches, QMD combines the "key nodes" identification process with the traditional traffic engineering ideas (multi-path routing) to maximally utilize network resource and satisfy the multicast receivers' QoS requirements. In addition, only the key nodes need to maintain QoS and multicast routing states, which can greatly reduce the core router's state maintenance burden without introducing complexity.

3. QOS-AWARE MULTICASTING IN DIFFSERV (QMD) DOMAINS

3.1 Design Philosophies & Assumptions

When a single-source multicast tree (constructed by CBT[3], SSM[6], PIM[16], etc.) passes a domain (an autonomous system), it usually takes the form of multicasting among a subset of the domain's edge (border) routers, one of which is the multicast source router (ingress border router) while others are the multicast receivers (egress border routers). Based on this abstraction, the goal of QMD is to construct a multicast routing tree connecting all of the on-tree border routers (one source and multiple receivers).

The primary advantage of DiffServ lies in its scalability. It concentrates the complex control plane functionalities at the edge routers facilitating scalable QoS provisioning. Following similar idea, QMD tries to remove the control functions from core routers while reducing the multicast forwarding states at the core routers. The design of QMD is based on the following philosophies:

- Relieving core routers from processing control messages and maintaining control state information.
- Decreasing the number of data forwarding states the core routers need to maintain.
- Enhancing Bandwidth Broker (BB) to support QoS-aware multicasting services in DiffServ domains.
- Providing predictable QoS for multicasting applications in DiffServ domains.

As introduced in [1], one or more Bandwidth Brokers (BB) could exist in the DiffServ domains to facilitate QoS resource management and admission control [1]. It is reasonable to assume that a BB has the information of its domain's topology and available QoS resources at each intra-domain link. QMD is based on the assumption of BBs' existence in DiffServ domains. To facilitate multicasting services within DiffServ domains, we propose to add multicast modules to BBs. A BB's multicast module will handle most of the multicast related control plane functions within its domain: processing join/leave requests, admission control, finding new branches, and informing the selected on-tree nodes to install the data forwarding states. This centralized approach can enhance the multicasting service security and ease multicast pricing compared to traditional IP multicasting.

To decrease the multicast routing state maintenance at core routers, for each multicast routing tree, only a sub-set of on-tree nodes (*key nodes*) maintain the multicast data forwarding states. The set of key nodes uniquely identify a QoS-aware multicast routing tree. The multicast traffic will be recursively unicasted among key nodes to multicast receivers. The *key nodes* set is composed of two parts: *branching nodes* and *milestone nodes*. The *branching nodes* are those nodes that have more than one child node in a multicast routing tree. The branching nodes can make sure that the multicast traffic to multiple receivers can share the network resources when it passes through the common part of the network. As mentioned earlier, DiffServ can only provide aggregate QoS support without guaranteeing per-flow service. The *milestone nodes* (as introduced in the following section) are used to enhance QoS support and provide more desirable QoS service to multicast receivers.

3.2 Milestone Nodes

DiffServ provides QoS by conditioning packets at the edge routers and using differentiated forwarding mechanisms based on codepoints associated with the packets. By aggregating traffic, DiffServ achieves good scalability. However, DiffServ can only guarantee QoS for the small amount of traffic marked with Expedited Forwarding [21]. For the large amount of traffic marked with Assured Forwarding [20], it cannot guarantee QoS on a per-flow basis.

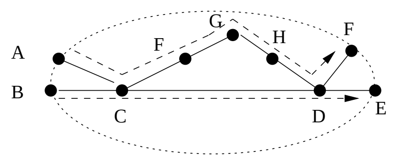


Figure 1: Example of Service Degradation.

Consider the topology shown in Figure 1. Suppose the capacities of the links are 5 Mbps each. Consider two flows passing through this domain: one from A to F, and the other from B to E. Suppose the traffic from A to F is 3 Mbps and marked with AF1, and the traffic from B to E is 4 Mbps and marked with AF1. Based on least-cost based routing, both the flows will pass through link C-D. In a well-engineered DiffServ domain, one of the flows cannot be accommodated. Otherwise, some traffic marked with AF1 will be dropped so that both of the flows will suffer service degradation in DiffServ domains. However, both of them will get predictable QoS if one of them takes the path C-F-G-H-D instead of C-D.

For example, the traffic from A to F can be first sent to G using the least-cost path, then to F following the least-cost path. We call this kind of nodes (node G in this example), where one flow must pass through to get predictable services, *milestone nodes*[1]. Using milestone nodes, the underlying routing protocol is still least-cost based. However, the traffic is recursively unicasted between milestones nodes. Based on this mechanism, more applications can be accommodated and receive predictable QoS service. Furthermore, the network traffic can be distributed in the network avoiding any potential hot spot problem.

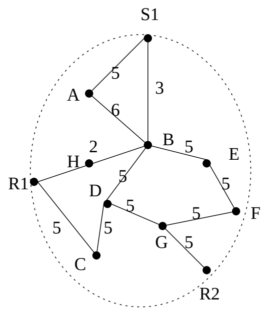
3.3 QMD Multicasting Tree Maintenance & Data Delivery

As mentioned earlier, in QMD, only the key nodes need to maintain the multicast data forwarding states, while all other on-tree routers just process the multicast traffic as unicast traffic. Figure 2 shows an example comparison between QMD and regular IP-based multicasting protocols. Figure 2 (a) shows a sample DiffServ domain topology, in which S1, R1, and R2 are the border routers and the values near the links are link capacities. Suppose S1 and S2 want to join a multicast group with a bandwidth requirement of 4 units, which has the source S1 within this domain. Most current IP-based multicast routing protocols are based on least-cost routing protocols, with the multicast routing tree constructed as shown in Figure 2 (b). The table entries (in the format of *groupid: list of children nodes*) beside the nodes are the corresponding multicast routing tables. Figure 2 (c) shows the corresponding multicast routing tree constructed by QMD, in which only a subset of on-tree routers maintain the multicast routing table and QoS reservation information. The multicast traffic is unicasted between two key nodes until it reaches the two receivers. Though other on-tree routers do not maintain any information, the receivers still can receive multicast traffic with predictable QoS.

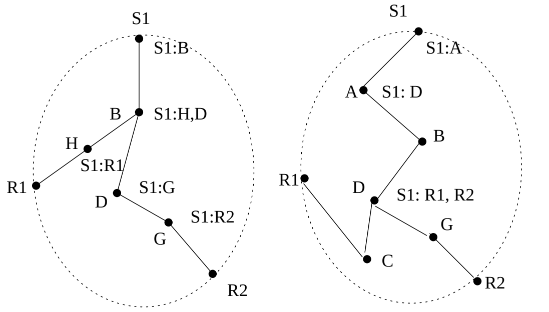
To facilitate QMD data forwarding, each *key node* maintains a *Multicast Forwarding Table* (MFT). Each Entry of MFT is composed of three fields: the group id, its children key nodes, and the DiffServ codepoints for the traffic to its children key nodes. The DiffServ codepoints make it more easier to satisfy multicast receivers' heterogeneous QoS requirement. The MFT routing entries are also softstate-based. If after some time, the MFT entry cannot get refreshed and the value of TTL becomes 0, the corresponding MFT routing entry would be removed.

The multicast traffic is carried by DATA type message which includes the group address and real multicasting data. A DATA message is a unicast packet with the destination address as the next hop key node. The DiffServ codepoint of DATA message is set as the corresponding multicasting traffic codepoint. When a key node receives a DATA type message, it first retrieves the group address and obtains the corresponding MFT entry. If the entry only has one child key node, it sends the DATA message to its child key node with the corresponding service level codepoint. Otherwise, it duplicates the DATA message and sends the message to each of its children key nodes with the corresponding service level codepoints. Thus, key-node-by-key-node, the multicast traffic is forwarded to all the receivers. The intermediate nodes between any two key nodes do not maintain any QoS reservation states or multicast routing states for the multicast group. They just forward the packets based on the destination addresses and DiffServ codepoints as normal unicast packets.

[1]Here, we do not intend to do per-flow management within DiffServ domain. The milestone nodes searching will be combined with searching of new multicast branch as discussed in the following sections.



(a) An Intra−domain Topology



(b) Regular Multicast Routing Tree (c) QMD's Multicast Routing Tree

Figure 2: Comparison of QMD and Regular IP-based Multicasting Protocols.

3.4 QMD Membership Management

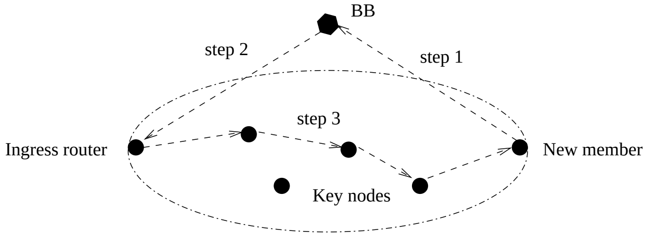


Figure 3: Basic Procedure of New Member Join.

Figure 3 shows the procedure of processing a new member's join request. As mentioned above, QMD's goal is to construct a multicast routing tree connecting a set of border routers. In QMD, the multicast's join request (a JOIN message) is usually sent from one of the border routers (edge routers) to its local domain BB (step 1 in Figure 3). The JOIN message sent by a border router includes its own IP address, service level requirement, and multicast group address. The JOIN messages can be initiated from other domains via inter-domain multicasting join requests. The inter-domain multicasting protocols can be CBT[3], MBGP/MSDP[2], SSM[6]. Each on-tree border router may have more than one downstream group members located in the other domains.

To support QMD, a BB needs to maintain a *Multicast group table*(MGT). An MGT entry records a multicast group information within this domain. An MGT has the following fields: 1) Group address; 2) A list of group members (border routers) and their TTL (time to live values). When the BB cannot receive a border router's refresh message (a JOIN message sent by a border router) after the TTL, it will be removed from the group member list. 3) A list of key nodes for this group and their corresponding information: IP address, the service level of the path from the multicast source to the nodes, and each key node's multicast data forwarding information.

When a BB receives a JOIN request, after admission control and locating this multicast ingress border router, it first uses the QMD-DIJKSTRA algorithm (as described in Section 4) to identify the key nodes which connects the new member to the existing multicast tree via a QoS-satisfied path. The number of key nodes indicates how many core routers will maintain the *Multicast Forwarding Table (MFT)* entry so as to provide QoS-aware multicast service to this new member. Then, it updates the MGT entry and sends out CONSTRUCT message (step 2 in Figure 3) to the ingress border router (multicast source). The CONSTRUCT message is used to set up the corresponding multicast forwarding entries at the key nodes. When a node receives a CONSTRUCT message, it looks up the corresponding entry for itself, removes it from the CONSTRUCT message and updates its multicast forwarding table (MFT). It then locates its children key nodes' addresses, duplicates the CONSTRUCT message and sends it to them (step 3 in Figure 3).

As mentioned earlier, the MFT entries at the key nodes are maintained in soft-state. The BB needs to periodically send CONSTRUCT messages to the ingress border routers so as to refresh the corresponding MFT entries at the key nodes. The membership of a multicast group is also maintained in soft-state, which means that the edge routers must refresh their membership status periodically (by sending JOIN messages). When a group member wants to leave (it has no downstream receivers), it can either stop sending refresh messages or send a LEAVE request to the BB. The BB can then remove the node's membership and release the reserved resources. If some key nodes' corresponding entries need to be updated, a new CONSTRUCT message will be sent out by the BB.

4. KEY NODES SEARCHING METHOD

In QMD, we use a modified version of DIJKSTRA algorithm (QMD-DIJKSTRA) to find the list of key nodes that can connect a new group member to the group's multicast tree while satisfying the new member's QoS requirement.

A domain can be modeled as a connected directed graph G(V, E), while V is the set of nodes and E is the set of links. A link from v_i to v_j is represented as (v_i, v_j). The available QoS resource between nodes i and j is Q(i, j). If Q(i, j)> SL_k, it means that the available QoS resource from i to j can meet the QoS requirement SL_k. V_E is the set of edge routers while V_C is the set of core routers (V = $V_E \cup V_C$). Suppose group M has ingress source edge router V_{Ei} at this domain and $V_{KNG} \in V$ is the set of key nodes for this group. The set KN_G depicts the DiffServ service levels for the multicast traffic from the ingress router to these key nodes. Each element of KN_G has a 2-tuple notation, $<v_k, SL_k>$.

We assume that the BB has the knowledge of the domain's topology G(V, E) and the available QoS resources on all e $\in$ E. We also assume that the BB can obtain the shortest paths between any two nodes in V (for example, using FLOYD algorithm[13]). Suppose an edge router V_{Ej} wants to join group M (its ingress source edge router within this domain is V_{Ei}) with service level requirement SL_j. Algorithm 1 depicts how the BB can find a new branch (a list of key nodes) that meets the new member's QoS requirement.

Algorithm 1 QMD-DIJKSTRA-2(G, V_{KNG}, KN_G, v_j, SL_j)

```
1  W ← V
2  Q ← {<$u_j$, <>, 0, 0 >}
/* Q is a set of 4-tuples, which is the set of nodes that have been selected. The first field is a node id. The second is the list of the key nodes on the path from $u_j$ to $V_{Ej}$. The third is the number of key nodes. The fourth is the cost of the path*/
3  while Q ≠ ∅
4    Extract $u$ = <$u_j$, <$u_1$,$u_2$,...>, n, c > from Q with the least value of Weight(n,c) to $v_j$, remove $u_j$ from W.
5    If $u_j \in V_{KNG}$ and $SL_{U_j} > SL_j$ /* Satisfying the QoS requirement.*/
6      return <$u_1$,$u_2$,...>
7    Else
8      For each v ∈ $u_j$'s neighbors and v ∈ W
9        If Q(v, $u_j$)> $SL_j$
10         C is the path cost from v to $u_j$
11         If $u_j$ is in the least cost path from v to $u_1$
12           Add <v,<$u_1$,$u_2$,...>, n, c + C > to Q
13         Else
14           Add <v,<$u_j$,$u_1$,$u_2$,...>, n+1, c + C > to Q
```

The algorithm can be explained using Figure 4. Suppose we have obtained the path from u_j to v_j with the list of key nodes, and u_1 is u_j's neighbor key node along the path. v is u_j's neighbor node. If u_j already belongs to the multicast tree, our task is done. Otherwise, if u_j is on the least-cost path from v to u_1, the key nodes from v to v_j should be the same as from u_j to v_j (line 12 in Algorithm 1). If u_j is not on the least-cost path from v to u_1, the key nodes from v to v_j should include u_j and the key nodes from u_j to v_j (line 14 in Algorithm 1).

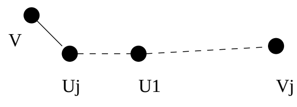


Figure 4: Explanation of QMD-DIJKSTRA Algorithm.

We can see that the Algorithm 1's computing complexity is same as the regular Dijkstra algorithm with $O(|V|^2)$. The definition of *Weight(n,c)* can be varied depending what kind of QMD tree the BB wants to construct. If $Weight(n, c) = n$, the algorithm can find a new branch with the least number of key nodes. This ensures that the new branch adds the least number of data forwarding states (entries) into core routers while meeting the receiver's QoS requirement. If $Weight(n, c) = c$, the new branch will consume the least amount of network resources while meeting the receiver's QoS requirement. If $Weight(n, c) = n * c$, it means that the algorithm considers both of the factors (resource consumption and forwarding states) at the same time.

5. SIMULATION & ANALYSIS

In this section, we conduct a series of simulations to study and compare the performance of QMD with other multicast routing protocols. Four algorithms are simulated: shortest path tree (SPT), QMD1 ($Weight(n, c) = n$), QMD2 ($Weight(n, c) = n * c$), and QMD3($Weight(n, c) = c$). In CBT (Core-Based tree)[3] and PIM (Protocol Independent Multicast)[16], the new member is connected to the multicast tree via unicast least-cost path, and the multicast tree is a shortest path tree (SPT). These routing algorithms can be categorized as "SPT" algorithm.

In reality, different domains can have different intra-domain topologies. Without the loss of generality, we use two random Waxman network topologies[30] and two real intra-domain topologies published by Rocketfuel[24] to simulate the intra-domain topologies. Our goal is to show the generic performance of QMD, not on any specific intra-domain topology. Waxman uses the following approach to generate a DiffServ domain topology: network nodes are randomly chosen in a square ($\alpha*\alpha$) grid. A link exists between the nodes u and v with the probability $P(u, v) = a*e^{-d(u,v)/(b*\alpha^2)}$, where d(u,v) is geometric distance between u and v, a and b are constants that are less than 1. Using this method, we generate two random topologies: one with 300 nodes (150 edge nodes) and 600 undirected links, one with 600 nodes (200 edge nodes) and 1800 undirected links. The two real intra-domain topologies are two typical ISP toplogies: SprintLink (AS 1239) and Level3 (AS 3356). The router-level SpringLink topology published by Rocketfuel has 604 nodes (242 edge nodes) and 2274 undirected links. The Level3 intra-domain topology has 624 nodes (193 edge nodes) and 5300 undirected links.

For each simulation, a multicast source and a set of multicast receivers are randomly selected out of the edge routers. In reality, each of the receivers can have one or more downstream receivers. During the simulation, we assume that each of the members has only one downstream receiver. The receivers join the multicast group in sequence with some random QoS requirement. There are 3 scenario (s1, s2, s3) setups for the physical links' QoS resource. In s1, for each receiver's request, each link has the probability of $\frac{1}{2}$ to meet the receiver's requirement. That means that half of the links can meet the receiver's QoS requirement. In s2, for each receiver's request, each link has the probability of $\frac{3}{4}$ to meet the receiver's requirement. In s3, for each receiver's request, each link has the probability of $\frac{7}{8}$ to meet the receiver's requirement. For each group size and algorithm, we simulated different sizes of multicast groups based on the above scenarios for 1000 times and computed the average results.

Based on above simulation setups, we evaluate the following performance metrics: average routing states, QoS success ratio, average routers involved setting up multicast trees, and average multicast routing tree cost.

$$Avg.\ routing\ states = \frac{\#\ of\ MFT\ Entries}{\#\ of\ multicast\ groups} \tag{1}$$

$$Avg.\ success\ ratio = \frac{\#\ of\ QoS\ satisfied\ requests}{\#\ of\ join\ requests} \tag{2}$$

$$Avg.\ \#\ of\ involved\ cores = \frac{\#\ of\ mcast\ table\ updates}{\#\ of\ groups} \tag{3}$$

$$Avg.\ tree\ cost = \frac{\#\ of\ links\ mcast\ trees\ passed}{\#\ of\ groups} \tag{4}$$

The *Average Routing States* is defined as the average number of multicast routing entries each core router needs to maintain for each multicast group. It is used to evaluate the core routers' burden of specifically routing multicast data traffic. The *Average Success Ratio* is defined as the ratio of finding a QoS-satisfied path connecting the multicast source to the new members (edge routers). To fairly compare other metrics, if a QoS satisfied path cannot be found for a new member, we will connect it to the multicast tree using the least-cost path. That means, the new member can still receive the multicasting traffic without its desired QoS. To compare the core routers' burden of processing the new group members' join requests, we use *Average Number of Core Routers involved* to evaluate how many core routers are involved setting up a new multicast routing tree. It is defined as the average number of multicast routing table changes in the core routers. The *Average Tree Cost* is defined as the average number of physical links a multicast tree passing-by. It can be used to evaluate the different multicast routing protocols' network resource consumption.

Figure 5 and Figure 6 show the average amount of routing states a core router needs to maintain for a multicast group for different group sizes under the different simulation setups. From the figures, we can observe that using QMDs, the core routers only need to maintain about half the amount of routing states compared to the other multicast routing protocols (SPT). Thus, QMD can decrease half the multicast routing burden of forwarding multicast data traffic (which is forwarded as the regular unicast traffic). This is because QMD only requires the key nodes of the multicast groups to maintain multicast routing states. The other on-tree nodes do not maintain any multicast routing states even when there are multicast traffic passing through them. When comparing the performance of QMDs in different scenarios (s1, s2, and s3), we can see that core routers in s1 maintain more routing states than s2, which again requires routers to maintain more states than s3. As we mentioned earlier in the simulation setup, s1 has $\frac{1}{2}$ of total links meet the receivers' QoS requirement while s2 has $\frac{3}{4}$ and s3 has $\frac{7}{8}$. When fewer links meet the receivers' QoS requirements, QMDs require more *milestone nodes* to maintain the routing states.

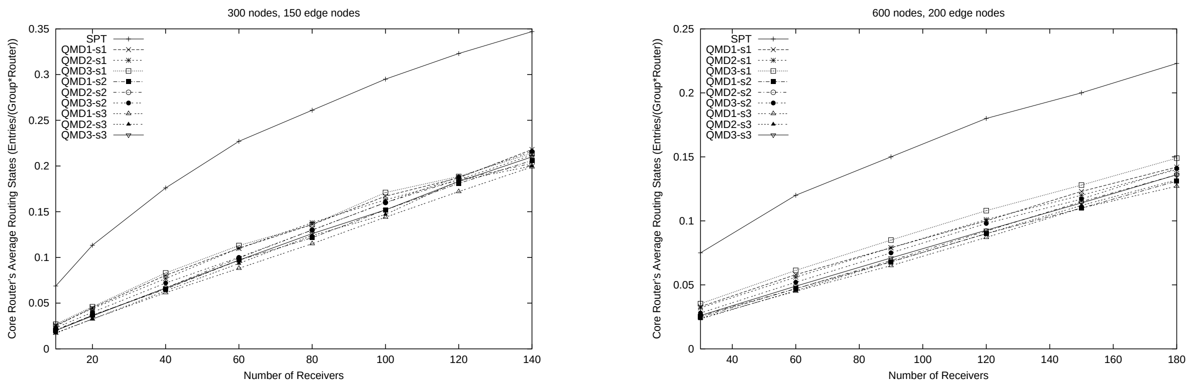


Figure 5: Comparison of Core Routers' Multicast Routing States (Random Topologies).

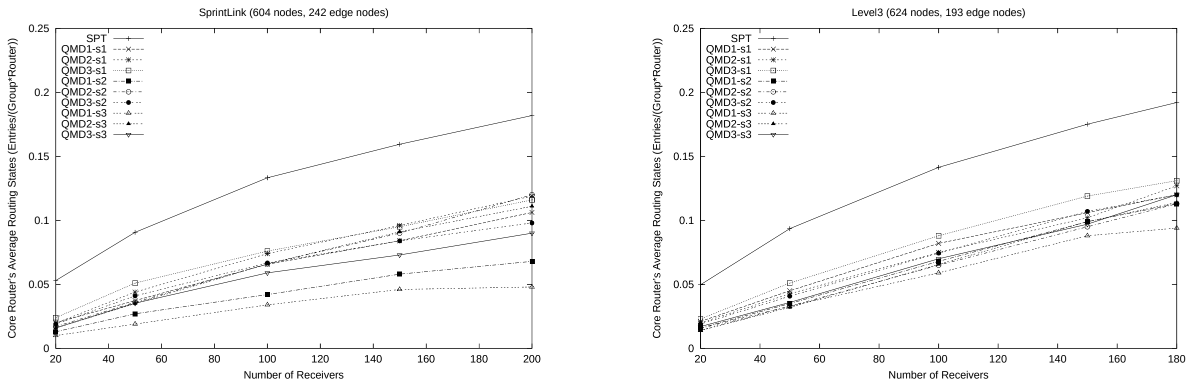


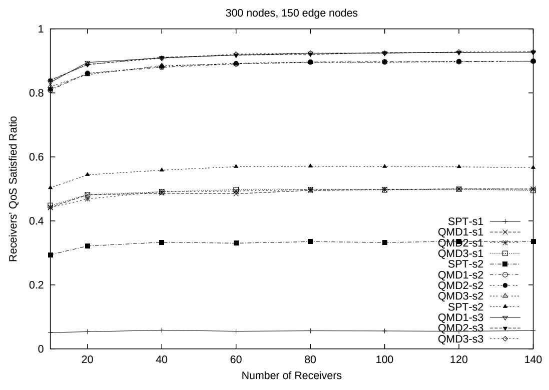
Figure 6: Comparison of Core Routers' Multicast Routing States (Real Intra-domain Topologies).

In addition, QMD3 requires the core routers to maintain a little more routing states than QMD1 and QMD2. This is because the definition of *Weight* in QMD1 only considers the number of key nodes while in QMD2 and QMD3, it also considers the cost of the multicast tree. As a result, QMD1 need a little less number of key nodes to connect a new member to the existing multicast routing tree.

Figure 7 and Figure 8 show the receivers' QoS success ratios using the multicast protocols for the different simulation setups. It can be observed that QMD1, QMD2 and QMD3 can achieve almost the same QoS success ratio for multicast receivers for each scenario. When comparing QMDs with SPT, we can see that QMDs can provide a much higher QoS success ratio. Because SPT's new multicasting branch searching is based on least-cost path search, it cannot provide receivers the desired QoS when some links do not meet the QoS requirement. However, QMDs can effectively avoid these links and provide receivers with better QoS branches.

In Figure 9 and Figure 10, we show the average number of core routers involved in processing multicast control messages for setting up a new multicast routing tree. The results show that QMDs have much fewer core routers participating in setting up a new tree because the join requests are sent directly to the BBs. Only the key nodes of the new branches need to add the new routing entries. The other on-tree nodes are not involved in processing the multicast control messages. Compared to QMD2 and QMD3, QMD1 has lowest number of core routers involved in setting up a multicast routing tree, which benefits from its definition of *Weight*. When comparing the different scenarios (s1,s2 and s3), we observe that s3 requires much fewer routers involved in setting up multicast routing trees. In s3, more links meet the receivers' QoS requirements which means that the multicast trees have fewer *milestone nodes* compared to the other scenarios.

Figure 11 and Figure 12 compare the average multicast tree cost (the number of links the trees pass by) for the different routing protocols under different simulation setups in these intra-domain topologies. From the results, we observe the following trends: the trees constructed by SPT have less cost than the trees constructed by QMDs; the trees in s3 have less cost than the trees in s2 (the same relationship for s2 and s1); the trees constructed by QMD2 have less cost than the trees constructed by QMD1 (same for QMD3 and QMD2). This is because of the following two reasons: 1) The definition of *Weight* of QMD2 and QMD3 contributes more in terms of the tree cost compared to QMD1; 2). If fewer links meet the receiver's QoS requirements, the multicast trees need to deviate more from SPT trees to meet the



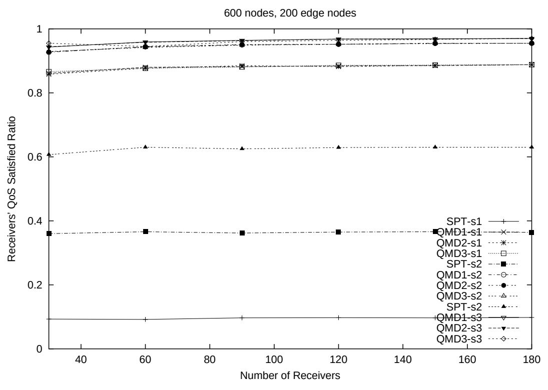
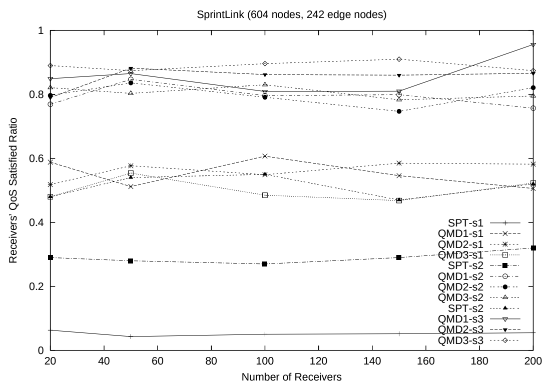


Figure 7: Comparison of Receivers' QoS Satisfaction Ratio (Random Topologies).



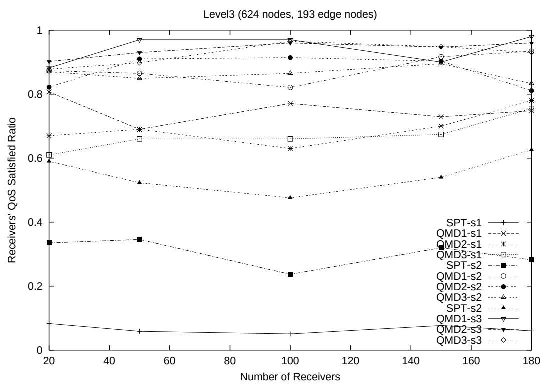


Figure 8: Comparison of Receivers' QoS Satisfaction Ratio (Real Intra-domain Topologies).

receivers' requirement, which results in higher tree cost.

From the above simulation results based on both random and real intra-domain topologies, we can conclude that QMD requires a lower number of core routers to maintain routing states compared to the SPT-based routing protocols. It uses QMD-DIJKSTRA algorithm to locate the key nodes to achieve higher success rate. Besides, it also puts lower burden on the core routers in processing multicast control messages to set up multicast routing trees.

6. CONCLUSIONS

In this paper, we introduce QMD as a scalable QoS-based multicast routing method for DiffServ domains. Based on the basic idea of DiffServ, we decouple the control plane and data plane functions in QMD. The edge routers together with the Bandwidth Broker (BB) process the join and leave events, find the lists of key nodes of QoS-satisfied branches (using the proposed QMD-DIJKSTRA algorithm) and other control functions. The key nodes only need to maintain the necessary data forwarding states. Thus, other nodes in the multicasting tree need not maintain any state information about the multicasting group. The merit of QMD is that it can provide QoS support to multicast group members without requesting the on-tree routers to do resource reservation and maintain QoS routing states. Simulation results have shown that QMD can provide better performance in terms of higher success ratio while incurring less multicast control burden on the core routers. QMD also can easily facilitate group members' heterogeneous QoS provisioning. The low overhead and implementation simplicity makes QMD an attractive candidate for adoption in DiffServ Domains.

7. REFERENCES

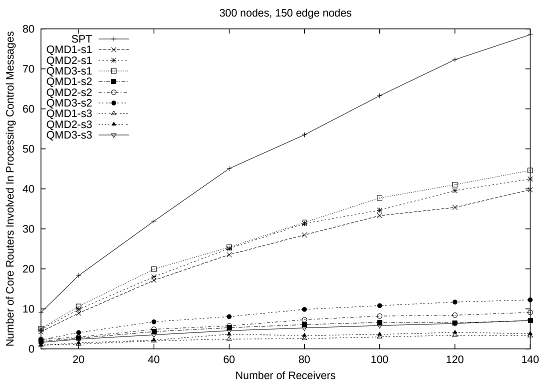
[1] Internet 2 – bandwith broker: requirements for internet2 qbone deployment, http://www.merit.edu/working.groups/i2-qbone-bb/requirements.html.

[2] K. Almeroth. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, pages 10–20, Jan./Feb. 2000.

[3] T. Ballardie. Core Based Tree (CBT) Multicast – Architecture Overview and Specification. *IETF RFC 2201*, 1995.

[4] D. Basak, H. T. Kaur, and S. Kalyanaraman. Traffic engineering techniques and algorithms for the internet. Tech report, Rensselear Polytechnic Inst., 2002.

[5] S. Berson and S. Vincent. Aggregation of internet integrated services state. *Internet-Draft*, Nov. 1997.



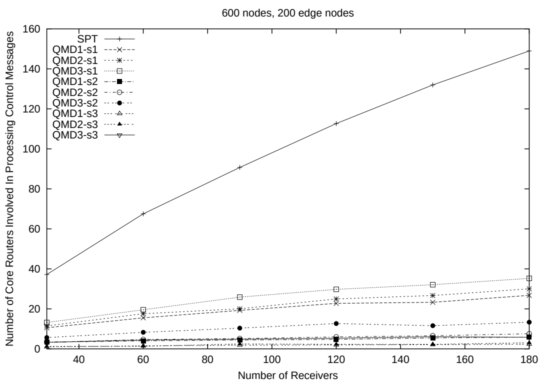
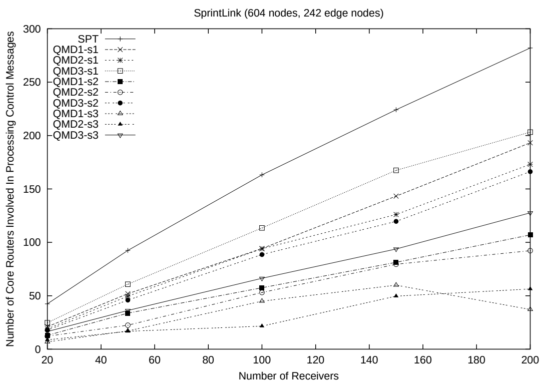


Figure 9: Comparison of Comparison of Core Routers Involved in Setting up Multicast Routing Tree (Random Topologies).



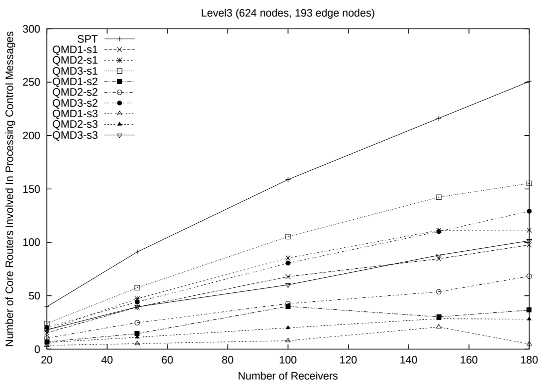


Figure 10: Comparison of Comparison of Core Routers Involved in Setting up Multicast Routing Tree (Real Intra-domain Topologies).

[6] S. Bhattacharyya. An overview of source-specific multicast (ssm). *RFC 3569*.

[7] G. Bianchi, N. Blefari-Melazzi, G. Bonafede, and E. Tintinelli. Quasimodo: Quality of service-aware multicasting over diffserv and overlay networks. *IEEE Network, Special Issue on Multicasting*, 2003.

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

[9] R. Bless and K. Wehrle. Group Communication in Differentiated Serivces Networks. In *1st International Symposium on Cluster Computing and the Grid*, May 2001.

[10] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *RFC 1633*, 1994.

[11] S. Chen and K. Nahrstedt. An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions. *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, Nov./Dec. 1998.

[12] S. Chen, K. Nahrstedt, and Y. Shavitt. A QoS-Aware Multicast Routing Protocol. In *IEEE INFOCOM*, May 2000.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. *Mc Graw Hill Higher Education*, 2001.

[14] L. H. M. K. Costa, S. Fdida, and O. C. M.B. Duarte. Hop-by-hop multicast routing protocol. August 2001.

[15] S. Deering. Multicast Routing in Internetworks and Extended LANs. Aug. 1988.

[16] S. Deering, D. L. Estrin, D. Farinacci, C. Liu V. Jacobson, and L. Wei. The PIM Architecture for Wide-area Multicast Routing. *IEEE/ACM Transcations on Networking*, 4(2), 1996.

[17] M. Faloutsos, A. Banerrjea, and R.Pankaj. QoSMIC: Quality of Service Sensitive Multicast Internet Protocol. *IEEE/ACM Transactions on Networking*, Vol.10, No.1, Feb. 2002.

[18] A. Fei and M. Gerla. Receiver-initiated multicasting with multiple qos constraints. In *IEEE INFOCOM*, May 2000.

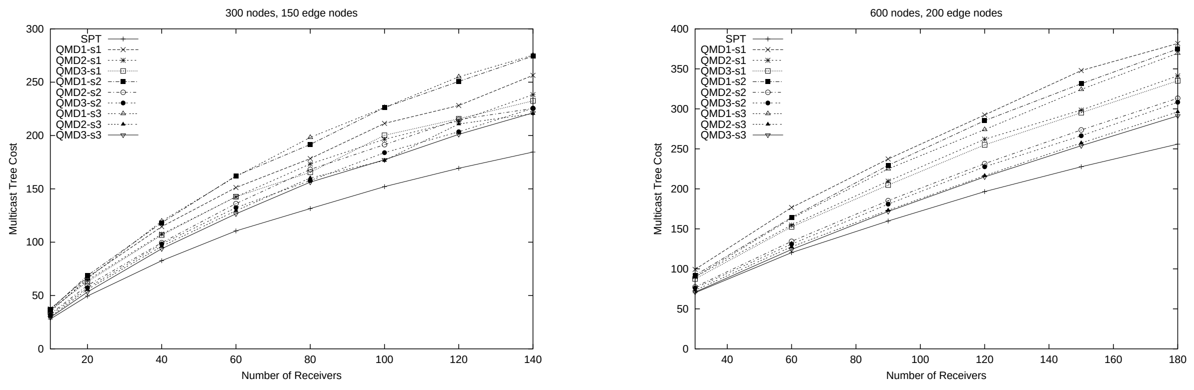


Figure 11: Comparison of Multicast Routing Tree Costs (Random Topologies).

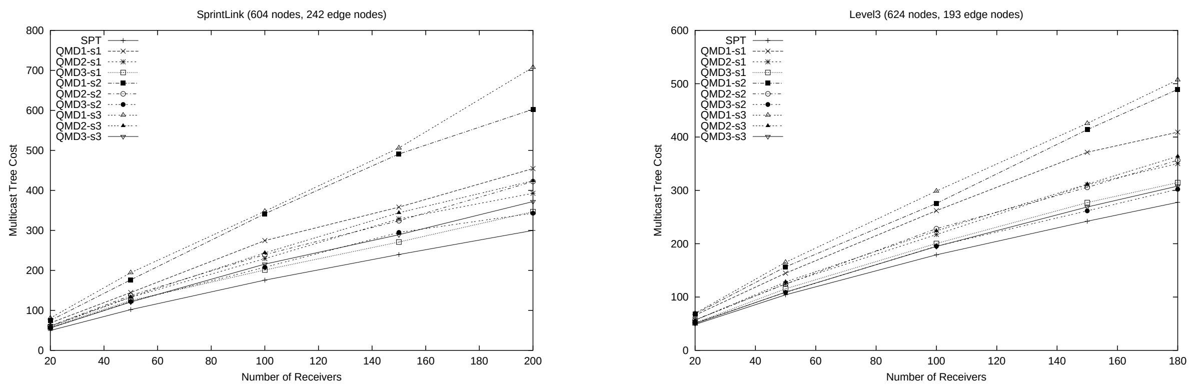


Figure 12: Comparison of Multicast Routing Tree Costs (Real Intra-domain Topologies).

[19] M. Gupta and M. Ammar. Providing multicast communication in a differentiated services network using limited branching techniques. In *IC 2002*.

[20] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding phb group. *IETF RFC 2597*, June 1999.

[21] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding phb. *IETF RFC 2589, Internet Engineering Task Force, June 1999.*, June 1999.

[22] Z. Li and P. Mohapatra. QoS-aware Multicast Protocol Using Bounded Flooding (QMBF) Technique. In *IEEE ICC*, May 2002.

[23] Z. Li and P. Mohapatra. Qos-aware multicasting in diffserv domains. In *IEEE Global Internet Symposium, Globecom*, Nov. 2002.

[24] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *Proceedings of ACM/SIGCOMM '02*, August 2002.

[25] I. Stoica, T.S. Ng, and H. Zhang. REUNITE: A Recursive Unicast Approach for Multicast. In *IEEE INFOCOM*, 1999.

[26] A. Striegel, A. Bouabdallah, H. Bettahar, and G. Manimaran. Ebm: Edge-based multicast in a diffserv network. In *Workshop on Network Group Communications (NGC)*, Sept. 2003.

[27] A. Striegel and G. Manimaran. Dynamic dscps for heterogeneous qos in diffserv multicasting. In *IEEE GLOBECOM*, 2002.

[28] A. Striegel and G. Manimaran. Dsmcast: A scalable approach for diffserv multicast. *Computer Networks*, vol. 44, no.6, April 2004.

[29] B. Wang and C. Hou. A Survey on Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols. *IEEE Network, Vol.14, No.1*, Jan./Feb. 2000.

[30] B. M. Waxman. Routing of Multipoint Connections. *IEEE Jornal on Selected Areas in Communications*, Vol.6, No.9, Dec. 1988.

[31] X. Xiao and L. M. Ni. Internet QoS: the Big Picture. *IEEE Network*, Mar. 1999.

[32] B. Yang and P. Mohapatra. Multicasting in differentiated service domains. In *IEEE Globecomm'02*, Nov. 2002.