QoS-Aware Multicasting in DiffServ Domains

Zhi Li and Prasant Mohapatra
Department of Computer Science
University of California at Davis
Davis, 95616, CA

Abstract— **Although many QoS-based multicast routing protocols have been proposed in recent years, most of them are based on per-flow resource reservation, which cannot be deployed within differentiated services(DiffServ) domains. In this paper, we propose a new QoS-aware multicast routing protocol called QMD, which is designed for DiffServ environments. QMD can provide scalable QoS-aware multicast services while greatly alleviating the core routers' multicast routing burden. In QMD, we separate the control and data forwarding functions. The edge routers are involved in multicast control plane functions: processing joining and leaving events, searching QoS-satisfied branch, and making resource reservation. Only a set of on-tree routers (key nodes) maintain multicast routing states and forward multicast data traffic. The key nodes of one multicast group are a subset of all the on-tree routers, which uniquely identify a QoS-satisfied multicast tree connecting the group members. Though the other on-tree routers between any two key nodes do not keep any multicast routing states and QoS reservation information, the multicast traffic still can get guaranteed QoS when transmitted from one key node to another. In addition, QMD can provide higher QoS-satisfaction rate while incurring less message overhead compared to other protocols.**

I. Introduction

The main idea of IP multicasting[8] is to construct a tree that connects all the group members and then the data travels along the multicast tree. Packets to a large number of hosts traverse only once through the common parts of the network, effectively saving network resources. Applications that benefit from multicasting include video conferences, large-scale content delivery, distance education, etc. Most of these applications are QoS-sensitive. However, the current Internet infrastructure only supports best-effort service, which is inadequate for these classes of evolving applications. To meet this requirements, many efforts have been dedicated to QoS provisioning in the Internet [16]. Among them, the Internet Engineering Task Forces (IETF) has proposed two different techniques to enhance QoS support in the Internet: Integrated Services (IntServ) and Differentiated Services (DiffServ).

IntServ is based on per-flow resource reservation to achieve QoS-aware communication[4]. To solve the scalability problem of IntServ, DiffServ was proposed, which provides aggregated QoS support in the Internet[2]. The packets are marked with different levels of services at the edge routers, and the core routers provide 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.

Though some QoS-based multicast routing protocols[14][10][11] have been proposed, almost all of these methods are based on per-flow reservation. All the on-tree routers have to maintain the resource reservation states of multicast groups, which make the protocols impractical to be deployed in DiffServ domains. Many other factors also prevent multicast to be deployed within the DiffServ domain. In QoS-aware multicasting, every on-tree router needs to maintain per-group data forwarding state, which conflicts with the idea of DiffServ. QoS-based multicast requires core routers to maintain forwarding states, process multicast control messages and make resource reservation for the groups, while the DiffServ requires only edge routers to make admission control and maintain resource reservation information. In addition, when regular IP multicasting is employed within the DiffServ domains, the Neglected Reservation Subtree problems(NRS)[3] also becomes an issue.

In this paper, we propose a scalable technique called *QoS-aware Multicasting in DiffServ domain* (QMD) to address the problems discussed above. When a multicast group passes through a DiffServ domain, it usually takes the form of multicasting between edge routers. So, our goal is to construct a scalable QoS-satisfied multicast tree within one DiffServ domain that can connect all these edge routers (one sender and multiple receivers). In QMD, we separate the control plane functions from data plane functions: edge routers will process most of the multicast control messages (join or leave events) and maintain the control state. The core routers only keep the minimal data forwarding state. To achieve this goal, the new member's (one edge router) join request is forwarded to an ingress edge router[1].

We assume that edge routers have the knowledge of domain topology and make admission control decision independently based on the current domain's available QoS resource. Using the proposed QMD-DIJKSTRA algorithm, the ingress router can compute a QoS-satisfied branch connecting the new member to the least number of *key nodes*(a small part of on-tree nodes). Then, the ingress edge route sends a construct message to these key nodes to set up the new branch. The multicast traffic can be recursively transmitted between the key nodes. Though the other on-tree

This research was supported in part by the National Science Foundation through the grants CCR-0296070 and ANI-0296034.

[1]The ingress and egress routers are named according to the direction of a multicast group's traffic passing through a domain. The ingress router is multicast source for the other egress routers within the domain.

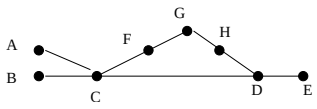


Fig. 1. Example of service degradation.

nodes do not maintain any multicast routing states and the QoS resource reservation status, the multicast traffic can still obtain predictable QoS according to the DiffServ codepoints. This protocol relieves the core routers from processing control messages and QoS reservation messages while maintaining much less routing states.

The rest of the papers is organized as follows. In the Section II, we will introduce the basic idea of QMD. The key algorithm - QMD-DIJKSTRA is introduced in Section III. We will evaluate the performance of QMD in Section IV, followed by the concluding remarks in Section V.

II. QMD Approach

DiffServ concentrates the complex control plane functionalities at the edge routers facilitating scalable QoS provisioning. QMD follows this idea: it concentrates all the control functions at the edge routers while trying to aggregate the multicast forwarding states at the core routers. In QMD, we require all the control messages to be processed at the edge routers. Although branchings could occur at the core routers, the control messages are not processed by them. As the edge routers have the domain topology information, they can find new branches and add the data forwarding states to the corresponding on-tree nodes.

A. Providing Predictable QoS within DiffServ Domain

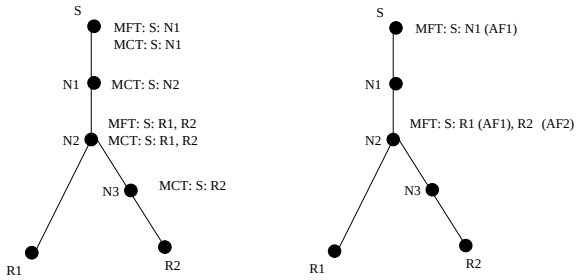
DiffServ provides QoS by conditioning packets at the edge routers and using a differentiated forwarding mechanism based on codepoints associated with the packets. By aggregating the traffic, DiffServ achieves good scalability. However, when the same service class is overloaded at some link, all the flows of that class will suffer serious service degradation.

Consider the topology shown Figure 1. Suppose the links' capacity is 5 Mbps. Consider two flows passing this topology: from A to D, and from B to E. Suppose the traffic from A to D is 3 Mbps and marked with AF1, and the traffic from B to E is 4 Mbps and marked with AF1. Since current Internet uses least-cost based routing protocols, both the flows will pass through link C-D. Since the link C-D's capacity is 5 Mbps, some traffic marked with AF1 will be dropped suffering service degradation. However, both of them will get predictable QoS if one of them takes the path C-F-G-H-D instead of C-D.

Suppose flow A-D knows that the path C-G-D is lightly-loaded and takes this path. The traffic from A to D will get predictable service if we can make sure that they can be first sent to G using the least-cost path, then to C following the least-cost path. We call this kind of nodes (node G) that one flow must pass through to get predictable services as *milestone nodes*.[2]

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

B. Recursive Unicast



(a) REUNITE (b) QMD

Fig. 2. Comparison of REUNITE and QMD.

"Recursive unicast" is proposed in [13]to achieve scalable multicast based on current unicast routing mechanism. Figure 2 (a) shows an enhanced version of recursive unicast [7]. As shown in Figure 2 (a), there is one multicast group in this domain (S, R1 and R2 are group members, S is the sender of this group). For regular multicast protocols, all on-tree nodes' of this multicast group, N1, N2 and N3 should have the multicast routing information for this group. When multiple groups coexist in the domain, the multicast routing table lookup will become a huge burden. In "recursive unicast", the multicast information is separated into two parts: multicast control table (MCT) and multicast forwarding table (MFT). All the on-tree nodes have the MCTs to process the join events while only *branching nodes* (those on-tree nodes have more than one children nodes) have MFTs. Then the multicast traffic is recursively unicasted between the branching nodes according to the MFTs till it arrives at the receivers.

In contrast to REUNITE [13], QMD only requires the ingress edge routers (S in Figure 2(b)) to process the control messages and maintain the control states while the core routers only keep the necessary data forwarding states. Because all the control states are kept at the edge routers, our approach releases the core routers from maintaining the necessary control states and processing control messages.

When join requests arrive at an ingress router, the ingress router computes the multicast branches and identifies the *key nodes*. Key nodes of one multicast group comprise of a sub-set of the on-tree nodes, and it includes two types of nodes: branching nodes, which can uniquely identify a multicast tree; milestone nodes, which can guarantee the QoS service on the paths from the ingress router to all the group members. The key nodes can determine a QoS-guaranteed multicast tree within one domain. In QMD, only the key nodes of one multicast group need to maintain its multicast data forwarding state.

C. QMD multicast tree maintenance

This section shows the necessary control messages and data structures that QMD needs to support QoS-based multicast routing.

As discussed in previous section, to support scalable QoS-based multicast, there are two kinds of nodes: *milestone nodes* and *branch nodes*. We call them together as *key nodes*. For one multicast group, only the key nodes of

the multicast routing tree need to keep MFTs. Each of the MFTs is composed of the group id, the children key nodes, and the DiffServ codepoint for the following branches. It has the following fields: <group address>, <<child key node IP1, service level1>, <child key node IP2, service level 2>,...>. Figure 2 (b) depicts the information that the on-tree nodes should maintain when using QMD for the same multicast group shown in Figure 2 (a).

Besides the MFT, the ingress edge router also needs to maintain *Multicast group table*(MGT). An MGT entry records the a multicast group information within this domain. An MGT has the following fields: <group address>, <<Multicast receiver IP1, TTL1>,<multicast receiver IP2, TTL2>, ...>, <<key node IP1,service level1, IP1's MFT entry for this group>, <key node IP2, service level2, IP2's MFT entry for this group >,...>. The second field of an MGT table is the list of the group members and their TTLs (time to live). When the ingress edge router cannot get a receiver's refresh message after TTL, the receiver will be removed from the list of group members. The third field of an MGT table is the list of all the key nodes for this group and their corresponding information: IP address; the service level for the traffic from the source to this key node; and its MFT entry for this group.

If an edge router wants to join a multicast group, it first gets the ingress router's IP address. Then, it sends a JOIN message to the ingress router, which includes its own IP address and service level requirement.

When an ingress routers receives the JOIN request, after admission control, it first uses the QMD-DIJKSTRA algorithm (Section III) to identify the key nodes along the path to transmit QoS-satisfied multicast data to the new receiver. Then, it updates the MGT entry and sends out CONSTRUCT messages. CONSTRUCT messages are used to set up corresponding MFTs at the key nodes. The messages carry the MFT entries for the key nodes along this path. When a node receives a CONSTRUCT message, it looks up the corresponding entry for itself, removes it from the CONSTRUCT message and updates the entry to its MFT table. Then, it finds its children key nodes, duplicates the CONSTRUCT message and sends to them.

The MFT entries at the key nodes are maintained in soft-state. So, the ingress routers need to send CONSTRUCT messages periodically 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 stops sending refresh messages. If the ingress router finds that a group member's TTL becomes 0, it removes the receiver from the group member list and releases the reserved resources. If some key nodes need to be deleted, the corresponding entries are removed from the key nodes list. It then sends a new copy of CONSTRUCT message to refresh the routing states at the key nodes.

D. Data Forwarding

The multicast traffic is carried by DATA type message which includes the group information and real data. When a key node retrieves a DATA type message, it first gets the group information and obtains the corresponding MFT entry. If the entry only has one child key node, it sends the same copy of DATA message to its child key node with the service level codepoint. Otherwise, it duplicates the DATA message and sends to its children key nodes with the corresponding service level codepoints. Thus, key-node-by-key-node, the multicast traffic is forwarded to the receivers. The intermediate nodes between any two key nodes do not maintain any QoS reservation states for the multicast group.

III. QMD-DIJKSTRA Algorithm

When a new JOIN request arrives at the ingress edge router, it needs to find a series of key nodes to form a new QoS satisfied branch connecting the new member to the multicast tree. The number of key nodes indicates how many core routers will maintain the MFT entry for the new branch. In QMD, we use a revised version of DIJKSTRA algorithm (QMD-DIJKSTRA) to find the minimum key nodes that can connect the new member to the available multicast tree.

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 edges. An edge 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 edge router V_{Ei} at this domain and $V_{KNG} \in V$ is its set of key nodes. There is another set KN_G depicts the current service levels for the multicast traffic from the ingress router to these key nodes. Each element of KN_G is a 2-tuple <v_k, SL_k>. By default, $KN_G = \{<V_E i, \infty>\}$.

We assume that the edge router V_{Ei} has the knowledge of the domain's topology G(V, E) and the available QoS resources on all e ∈ E. We also assume that the V_{Ei} can compute the shortest paths between any two nodes in V (for example, using FLOYD algorithm[6]).

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

```
1   W ← V
2   Q ← {<v_j, <>, 0 >}
/* Q is the set of 3-tuples. The first field is the node id. The second
is the list of the key node between the node and v_j. The third is the
number of key nodes.*/
3   while Q ≠ ∅
4     Extract u = <u_j, <u1,u2,...>,n > from Q that has the least
number of key nodes (least value of n) to v_j, remove u_j from W.
5     If u_j ∈ V_KNG and SL_Uj > SL_j
6       return <u1,u2,...>
7     Else
8       For each v ∈ u_j's neighbors and v ∈ W
9         IfQ(v, u_j)> SL_j
10          If u_j is in the least cost path from v to u1
11            Add <v,<u1,u2,...>,n > to Q
12          Else
13            If v ∈ V_E
14              Add <v,<u_j,u1,u2,...>,n> to Q
15            Else
16              Add <v,<u_j,u1,u2,...>, n + 1> to Q
```

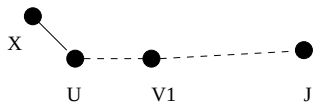


Fig. 3. Explanation of QMD-DIJSKTRA algorithm

When an edge router V_{Ej} wants to join group M with service level requirement SL_i, Algorithm 1 depicts how the ingress router V_{Ei} can find a new branch that meets the new member's QoS requirement with the least number of key nodes. The algorithm can be explained using Figure 3. Suppose we have obtained the path from U to J that has the least number of key nodes, and V1 is the neighbor key node to U along the path. X is U's neighbor node. If U already belongs to the multicast tree, our task is done. Otherwise, if U is on the least-cost path from X to V1, the key nodes from X to J should be the same as from U to J. If U is not on the least-cost path from X to V1, the key nodes from X to J should include U and the key nodes from U to J. In this situation (as shown in line 13 of Algorithm 1), if X is an edge router, the key nodes on the path from X to J are the same as from U to J. The goal of obtaining the least number of key nodes is to minimize the core routers' routing states.

IV. Simulation & Analysis

In this section, we carry a series of simulations to study and compare the performance of QMD with some other multicast routing protocols. Four algorithms are simulated: shorted path tree (SPT), M-QOSPF, QMRP-2[5] and QMD.

In CBT (Core-Based tree)[1] and PIM (Protocol Independent Multicast)[9], 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. M-QOSPF is based on the QoS routing extensions to OSPF proposed by [12]. It assumes that all the routers know the domain's topology as well as the QoS status. When a new router wants to join a multicast group, it computes a QoS-satisfied path toward the source (or core router) and sends join request to the source along the path[14]. QMRP is a distributed QoS-aware multicast routing algorithm proposed in [5]. From the simulation results as shown in [5], QMRP-2 can achieve higher success ratio with lower message overhead.

In the simulation, we evaluate the following performance metrics: average routing states, success ratio, average message overhead, and average routers involved processing multicast control messages.

$$\text{Average routing states} = \frac{total\ number\ of\ routing\ states}{total\ number\ of\ multicast\ groups}$$

$$\text{Average success ratio} = \frac{number\ of\ QoS\ satisfied\ branch}{total\ number\ of\ join\ requests}$$

$$\text{Average message overhead} = \frac{number\ of\ messages\ sent\ out}{total\ number\ of\ join\ requests}$$

$$\text{Average number of core routers involved} = \frac{number\ of\ control\ messages}{total\ number\ of\ join\ requests}$$

For the message overheads, if a message passes through m hops, it is counted as m messages. If we can not find a QoS satisfied branch for a new member, we will connect it to the multicast tree using least-cost path. Because the overhead of processing the multicast control messages and regular unicast messages is different, we use *Average number of core routers involved* to evaluate this measure.

The simulations are based on the Waxman network topology[15]. We use 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. In the simulation, $a = 0.2$,$b = 0.3$, and $\alpha = 100$. Using this parameters, we generate a random topology with 100 nodes. Then, we randomly pick 25 nodes out the 100 as the edge routers. Others are used as core routers. For each simulation, a multicast source and a group of multicast receivers are randomly selected out of the edge routers. The receivers join the multicast group in sequence with some random QoS requirement. Each link of the domain meets the receivers' QoS requirement with the probability of 0.5.

We simulate five different sizes of multicast groups: 2 receivers, 5 receivers, 10 receivers, 15 receivers and 20 receivers. For each group size and algorithm, we run the simulation 1000 times.

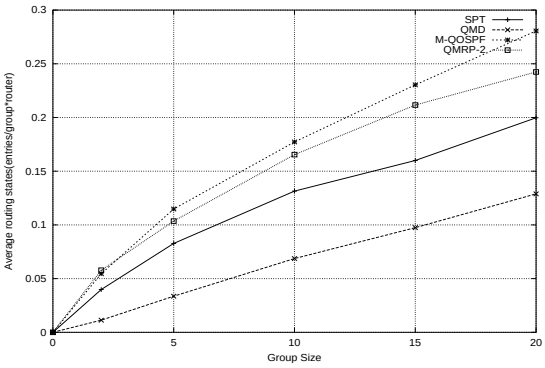


Fig. 4. Comparison of multicast routing states.

Figure 4 shows the average routing states a core router needs to keep for a multicast group under different group sizes situation. From the figure, we can see that when using QMD, the core routers only need to maintain half the amount of routing states compared with other multicast routing protocols. That means that QMD can decrease half the multicast routing burden of forwarding multicast data traffic (which is forwarded as regular unicast traffic). Thus, QMD only requires the key nodes of multicast group to keep multicast routing states. The other on-tree nodes do not maintain routing states even when there are multicast traffic passing through them.

Figure 5 shows the receivers' QoS success ratios using the four multicast protocols. As we have mentioned earlier, each link has 50% chance to meet the receivers' QoS requirement. When using QMD, the average success ratio is around 50%. Because M-QOSPF also uses centralized routing method, it can achieve the same success rate as QMD. However, for SPT multicast, because its branch searching

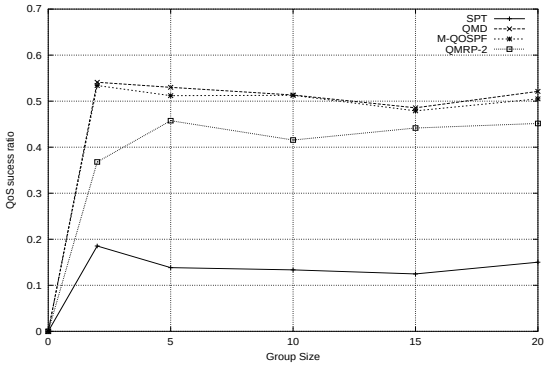


Fig. 5. Comparison of QoS success ratio.

is based on least-cost path search, it cannot achieve high QoS satisfaction ratio as QMD.

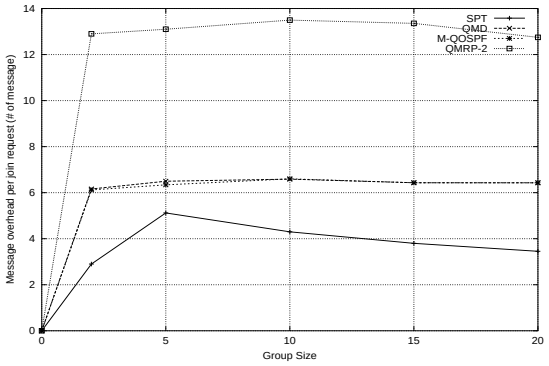


Fig. 6. Comparison of message overhead of searching new branch.

Figure 6 compares the average message overhead (number of messages per join request) for the four different routing protocols. QMRP-2 has the highest message overhead because it uses distributed feasible branch search methods. For each new branch, it will try many paths. QMD and M-QOSPF use centralized methods to find feasible branches and thus incur low message overhead. SPT has the lowest overhead because it always takes the least-cost path.

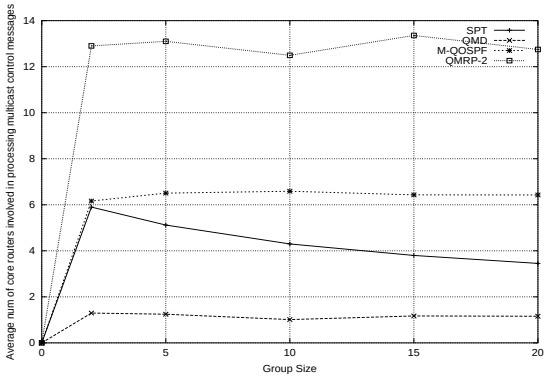


Fig. 7. Comparison of core routers involved new branch setup.

In Figure 7, we shows the average number of core routers involved in processing multicast control messages for one join request. QMD has the lowest impact because the join requests are sent directly to the ingress routers (multicast source within the domain). Only the key nodes of the new branch will process CONSTRUCT messages. The other on-tree nodes are not involved in processing the multicast control messages.

From Figures 4-7 we observe that, QMD requires less core routers to maintain routing states compared to other multicast 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 by processing multicast control messages and data messages separately.

V. 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 separate control plane and data plane functions in QMD. The edge routers process joining and leaving events, find QoS-satisfied path with least number of key nodes (using QMD-DIJSKTRA algorithm we proposed) and other control functions. The key nodes only need to keep the necessary data forwarding states. The merit of QMD is that it can provide QoS guarantee to multicast group members without requesting the on-tree routers to do resource reservation and maintain QoS-routing states. Simulation results show that QMD can achieve higher success ratio with less multicast control burden on the core routers.

References

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

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

[3] R. Bless and K. Wehrle. Group Communication in Differentiated Serivces Networks. *IETF draft*.

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

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

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

[7] L.H.M.K. Costa, S. Fdida, and O. C. M.B. Duarte. Hop-by-hop multicast routing protocol. *ACM SIGCOMM' 2001*, pages 249–259, August 2001.

[8] S. Deering. Multicast Routing in Internetworks and Extended LANs. *SIGCOMM'88*, Aug. 1988.

[9] 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.

[10] M. Faloutsos, A. Banerrjea, and R.Pankaj. QoSMIC: Quality of Service Sensitive Multicast Internet Protocol. In *ACM SIGCOMM*, Sep. 1998.

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

[12] R.Guerin, A.Orda, and D. Williams. Qos Routing Mechanisms and OSPF Extensions. *IETF Draft*, Nov. 1996.

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

[14] 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.

[15] B. M. Waxman. Routing of Multipoint Connections. *IEEE Jornal on Selected Areas in Communications*, Dec. 1988.

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