Heterogeneous QoS Multicast in DiffServ-like Networks

A. Sai Sudhir and G. Manimaran
Dept. of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011, USA
saisud@iastate.edu, gmani@iastate.edu

Prasant Mohapatra
Dept. of Computer Science
University of California
Davis, CA 95616, USA
prasant@cs.ucdavis.edu

Abstract

Multicasting in DiffServ networks is a challenging problem due to the architectural conflicts between them, namely, stateful vs. stateless core. In this paper, we assume an edge-based multicast (EBM) model wherein the multicast tree is constructed such that the branching occurs only at the edge routers. We propose an algorithm to solve the problem of dynamic member join/leave in heterogeneous QoS multicasting under EBM model. We formally state the problem and propose an algorithm for it, which is optimal when the constraint on the member join/leave requires that there can be no service disruption for on-tree nodes. We then evaluate the performance of our algorithm with respect to a static multicast tree construction heuristic and a source-based shortest path algorithm using "Tree QoS Cost" as a primary metric. Our studies show that the proposed algorithm achieves good performance in terms of Tree QoS Cost, time taken for member join/leave, and number of service disruptions with acceptable storage overhead.

I. Introduction

The Differentiated Services (DiffServ) architecture [1] was proposed to provide scalable QoS on an Internet-wide scale. DiffServ is built upon a simple model of traffic conditioning and policing at the edges of the network in addition to classifying flows into different service classes. The traffic is forwarded using simple differentiated treatments, called per-hop behaviors (PHBs), in the core of the network. These two main components aim to provide QoS guarantees for the service flows. This differential treatment results in differential pricing which is a motivating factor for the adoption of this idea by major network providers and ISPs. Multicasting [2] has been a popular mechanism for supporting group-based applications, such as videoconferencing and content distribution.

Though multicasting and DiffServ are complementary technologies, there are two primary architectural conflicts between them [3]. The first is that multicasting mandates the maintenance of per-group state information at all routers, while DiffServ relies on the statelessness of the core. The second conflict is that multicasting is based on *receiver-driven QoS* whereas DiffServ is based on *sender-driven QoS*. To address these conflicts, an Edge-Based Multicasting (EBM) approach was proposed in [4] and is characterized by a multicast unaware core. Only edge routers participate in multicast and replicate multicast packets. EBM confirms to the DiffServ philosophy of stateless core and is easy to deploy.

Due to the receiver-driven nature of multicasting, different receivers of a multicast group can request varying levels of QoS (PHBs) as supported by the DiffServ framework. The DiffServ framework provides three per hop behaviors (PHBs) or levels of service namely *Expedited Forwarding (EF)* [17], *Assured Forwarding (AF)* [18] and *Best Effort (BE)*. Although it may be acceptable if a receiver requesting a lower level of service receives a higher level of service, a receiver requesting a given level of service must receive it. Due to the heterogeneity in receiver QoS levels, different links in a multicast tree carry different types of traffic such that the QoS requirements of downstream receivers are satisfied. There is a QoS cost associated with the multicast tree and the tree constructed must be QoS cost-efficient.

In this paper, we propose a dynamic member join/leave algorithm that adds/removes members with heterogeneous QoS requirements to/from an existing multicast tree in DiffServ networks. In this context, the problems are informally stated as follows:

- The goal of the member join algorithm is to find an *optimal on-tree attachment point* for the new member such that the increase in *tree QoS cost* (defined later) is minimum.

- The goal of the member leave algorithm is to find an *optimal on-tree pruning point* for the leaving member such that the decrease in tree QoS cost is maximum and such that there is no service disruption for on-tree nodes.

It must be noted that the fact that there can be no service disruptions on the on-tree nodes when a member join/leave operation occurs serves as a constraint on the problem. It is not a feature of the problem or its solution.

The following are the key distinguishing features between the traditional member join/leave problem (e.g., [8], [9], [10], [12], [13], [14]) and the member join/leave problem considered in this paper.

1) The traditional QoS join/leave algorithms aim to satisfy QoS constraints such as end-to-end delay or loss between source and a receiver, or inter-receiver delay variation, in addition to minimizing cost. Whereas, the algorithms for the given join/leave problem need to satisfy "end-to-end service level constraint", in addition to minimizing QoS cost. The term end-to-end service level constraint refers to finding a source-receiver path whose constituent links must support a service level greater or equal to that of the service level requested by the receiver. For example, for a receiver requesting an AF class of service, the links of the source-receiver path must support AF or EF level of service but not BE (assuming EF is a higher level of service than AF, and BE is a lower level of service than AF). Since the join/leave problem considered in the paper has only one path constraint (QoS cost) and one link constraint (service level), the problem does not belong to NP-complete class and it can be solved by a polynomial time algorithm.
2) The traditional join algorithms aim to minimize (maximize) the tree cost between the attachment (pruning) point and the new member. Whereas, algorithms for the given join (leave) problem need to minimize (maximize) the combined "tree QoS cost", which is the sum of the QoS cost of the attachment path and the QoS cost, if any, due to upgrading (downgrading) of on-tree links to account for higher QoS level of the new (leaving) member.

The rest of the paper is organized as follows. In Section 2, we examine related work. In Section 3, we formulate the problem. In Section 4, we describe the algorithm to solve the problem. In Section 5 we prove that our method is optimal. In section 6 we perform simulation studies and we conclude in Section 7.

II. Related Work

Several multicast tree construction algorithms have been proposed for both static and dynamic multicast groups. We first examine related static tree construction work and then discuss related work in dynamic QoS multicast routing. For the sake of discussion we group static tree algorithms into two categories as follows.

Algorithms for Heterogeneous QoS : The authors in [8] build a multicast tree subject to delay and delay-variation constraints which are QoS parameters. In [9], the author proposes a heuristic to construct a multicast tree that minimizes bandwidth (rate) usage. [10] constructs a multicast routing tree taking into account the heterogeneity of receivers in terms of bandwidth and delay, and at the same time consumes as little resources as possible. [11] presents an approximation algorithm to find a balance between a minimum-cost multicast tree and a minimum-delay multicast tree with a provably good performance when the link delay and link cost are equal.

Algorithms for Heterogeneous QoS in a DiffServ domain : In [4], the authors propose the concept of Edge-Clustered Tree (ECT) in a DiffServ setting which ensures that a receiver with a higher service level cannot sit downstream from a receiver with a lower service level.

QoS-aware dynamic multicast routing : Based on how the new member is connected to the tree, multicast routing protocols can be classfied into two broad categories: single-path routing (SPR) and multiple-path routing (MPR). An SPR provides a single path connecting the new member to the tree, whereas an MPR provides multiple candidate paths. Several SPR protocols such as DCUR and RDM [12] typically use delay and cost tables for making routing decisions during QoS path setup. MPR protocols provide or probe multiple candidate paths in order to increase the chances of finding a feasible path (i.e., a path that satisfies the QoS requirements of the member). Spanning-join, QoSMIC, QMRP, and parallel probing are among the recently proposed MPR protocols [13], [14] (and the references therein). [15] addresses the problem of member join/leave in a DiffServ setting, but it does not assume an EBM model.

It is important to mention that our work can also be applied to the End-system Multicast (ESM) model [16]. EBM and ESM models are somewhat similar in the sense that, in EBM, only edge routers are multicast capable, and in ESM, only the member end hosts are required to be multicast capable. The problem of heterogenous QoS dynamic multicasting, which we pose in this paper, has neither been addressed in the context of EBM nor ESM. Therefore, the solution for member join/leave presented here can be applied to both EBM and ESM models.

III. Problem Formulation

A. Network Model

We represent the DiffServ Network by a weighted undirected graph $G = (V, E)$. V denotes the set of nodes, and E, the set of edges corresponding to the set of communication links connecting the nodes. We define a function $W : E \rightarrow Z^+$ which assigns a nonnegative weight to each link in the network. A *path* from a vertex u to a vertex v is denoted by $(u \rightsquigarrow v)$. Since our EBM model allows multicast tree branching to occur only at the edge routers, we transform the given graph $G = (V, E)$ into another undirected graph $G_R = (V_R, E_R)$, where V_R denotes the set of all edge routers in the DiffServ network $(V_R \subseteq V)$. The edge set E_R is the set of shortest paths in G between vertices $u, v \in V_R$, and the weight of an edge $(u, v) \in E_R$ is the length of the shortest path between u and v in G.

B. Problem Definition

We consider the following multicasting scenario. Packets originating at some source node called the *Ingress node*, $I \in V_R$ have to be sent to a set $M \subseteq V_R - \{I\}$ of destination nodes. Multicast packets are routed from I to the members of M on the links of the multicast tree $T = (V_T, E_T)$ rooted at I. The multicast tree is a subgraph of G_R spanning I and the nodes of M. The following definitions are useful in precisely defining the problem.

QoS Level: Let L denote the set containing all possible levels supported by the DiffServ framework. We define a *level assignment function,* $Q : M \rightarrow L$ which assigns a level of service to a multicast receiver $r \in M$. Packets marked with higher levels of service receive better treatment than those with lower levels of service at a router.

Cost of a Level: We also define a *cost assignment function,* $C : L \rightarrow Z^+$ which assigns a positive cost to a particular level. Let l_i and l_j be any two service levels in L such that $l_i > l_j$, i.e. l_i is a higher level service than l_j. Then $C(l_i) > C(l_j)$.

End-to-end QoS: Let $(I \rightsquigarrow r)$ denote the path from *Ingress node* I to multicast receiver $r \in M$ in the tree T. In order to satisfy the "end-to-end QoS requirement" of the receiver r requesting service level l_i, all links $h \in (I \rightsquigarrow r)$ must carry traffic of level l_i or above.

QoS Cost of a link: The QoS cost associated with a link h is $W(h) * C(l_i)$, where function W assigns a non-negative cost to a link.

If h is on the path from I to multiple receivers, then the level of traffic carried by h will be the highest of service levels requested by those receivers. If any receiver has requested a lower level of service it can still receive a better service if it sits upstream from any higher service level receiver. This is called *Good Neighbor Effect* [20].

The QoS cost of a link (more specifically the function C) is a *utility function* dependent upon factors such as traffic models, scheduling disciplines and network topology.

a) The Minimum QoS Cost Multicast Tree (MQMT) problem:: Given a graph $G_R = (V_R, E_R)$, a set of multicast receivers M, a source $I \notin M$, a level-assignment function Q and a cost-assignment function C, a QoS Multicast Tree (QMT) is a tree $T = (V_t, E_t)$ with an assignment of levels $H(h)$ to every edge $h \in E_t$ satisfying the following conditions:

- T is a subgraph of G_R rooted at I.
- For every receiver $m \in M$, the path $I \rightsquigarrow m$ satisfies the end-to-end QoS requirement of m.

The QoS cost of tree T is:

$$\sum_{h \in E_t} W(h) * C(H(h)) \tag{1}$$

The minimum QoS cost multicast tree is the QoS multicast tree T with the smallest possible cost.

In other words, the multicast receivers should receive the QoS they requested and at the same time the *tree QoS cost* of the multicast tree must be minimum. In [21] we proved that Problem MQMT is NP-complete.

Problem QMT-Join-Leave (QMT-JL)

Let us consider a QMT $T(V_t, E_t)$ constructed by a static multicast tree construction algorithm. Let cost of T be $cost(T)$. Let us say a node r requesting service level l_i wants to join T. After joining the tree, lets call the resulting tree $T^{'}(V_{t^{'}}, E_{t^{'}})$. The cost of this tree is $cost(T^{'})$.

In this scenario, A *join* operation is one that minimizes $cost(T^{'})$ - $cost(T)$ and which satisfies the end-to-end QoS requirement of node r. Note that a join operation will not decrease the tree QoS cost because join does not involve tree rearrangement.

A *leave* operation is one that maximizes $cost(T)$ - $cost(T^{'})$ but without disrupting service of any on-tree node.

In this paper we assume that $C(EF) = 3$, $C(AF) = 2$ and $C(BE) = 1$. It must be noted that the prioritization of QoS levels as described above is loosely defined. In reality, they are application specific. For the sake of simplicity and consistency throughout this paper, we treat EF as a better QoS level than AF.

Fig. 1 shows an example of a node join operation. Node c was only receiving BE service before d joined, but since d has requested EF service, the link between c and i is upgraded to EF service so that the end-to-end QoS requirement of d is satisfied.

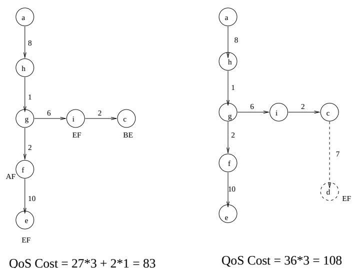

Fig. 1. Example of a Join Operation

Node ID	QoS Cost from Source	EF	AF	BE
a	0	0	0	0
h	24	0	0	0
g	27	0	0	0
f	33	0	0	0
e	63	0	0	0
i	45	0	0	0
c	47	4	2	0

Fig. 2. Information maintained at source

IV. An Algorithm for QMT Join & Leave

Our algorithm relies on maintaining certain information about the type of traffic each link of the tree is carrying and how the cost of that link will vary if the type of traffic through that link changes due to member join/leave. Since the source is informed of every join and leave operation, it knows the entire traffic information pertaining to that tree. So it can maintain the change in the QoS cost of a link if the traffic through that link changes. The information maintained for the tree in Fig. 1 is shown in Fig. 2. Each entry consists of the node ID, the QoS cost from the source to the node and the change in the QoS cost from the source for each type of traffic level.

Member Join

The idea behind our algorithm is that the links on the path from the source to the joining node will have to carry a traffic atleast equal to that of the joining node's service level. Some on-tree links might have to be upgraded for this to happen and the change in the cost due to the upgradation is given by the table. New links will carry the service level requested by the new receiver. The cost of the on-tree links have to be upgraded by a value reflecting the difference in the QoS cost between the traffic through that link before and after the member join, and the links that are not already part of the tree will have a QoS cost corresponding to the traffic level of the new member.

Fig. 3 shows an example of the steps involved in member join. Fig. 3(a) shows the auxiliary graph with the modified edge costs. Fig. 3 shows the shortest path between the source and the new member in the auxiliary graph. Fig. 3(c) shows the new tree with the link costs after node d has joined.

Time Complexity of Member Join

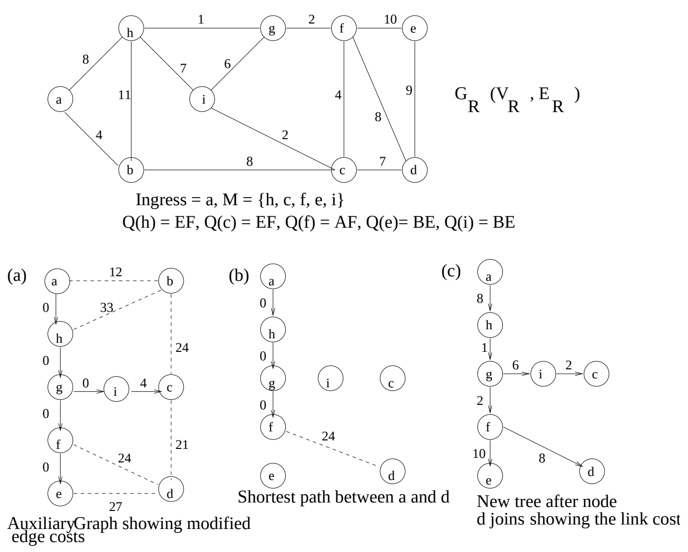
$G_R\ (V_R\ , E_R\ )$
Ingress = a, M = {h, c, f, e, i}
Q(h) = EF, Q(c) = EF, Q(f) = AF, Q(e)= BE, Q(i) = BE
(a)
Auxiliary Graph showing modified edge costs
(b)
Shortest path between a and d
(c)
New tree after node d joins showing the link cost

Fig. 3. Steps in member join

Let n be the number of edge routers and e be the number of edges in the graph. The determination of the attachment point for the new member by the source of the multicast tree requires one shortest path computation and it can be done in O($n.log\ n\ +\ e$) using a *Fibonacci heap* [19] implementation. After the join has taken place, the table maintained at the source has to be altered (if necessary). In the worst case, all links in the multicast tree might have to be updated. Hence, this step takes O(n) time.

Member Leave

We consider two scenarios for members leaving the tree.

- The first case is the leaving member is a leaf node in the tree. In this case we prune the tree until we meet another member or we meet any other node with an out degree of more than one. We then propagate the highest service level of its downstream receivers towards the source.
- The second case is when a non-leaf member wants to leave the group. In this case, we do not physically remove the node from the tree. Instead we simply propagate the highest service level of its downstream receivers towards the source.

Time Complexity of Member Leave

In the worst case all nodes in the existing multicast tree have to be examined while propagating the highest level of any downstream receiver towards the source. This situation can occur when a receiver of the highest level is far away from the source. Hence, the worst case time complexity of member leave is O(n) where n is the number of edge routers. The source table might have to be updated after the member leaves. This step again takes O(n) time.

Storage Complexity of QMT-JL
The source table contains one entry for every link in the multicast tree. The maximum number of links in the tree can be $n-1$ where n is the number of edge routers. The table also contains one entry for each service level supported by that DiffServ domain. Thus, the overall storage complexity of QMT-JL is O(nl), where l is the number of levels in the DiffServ domain. It must be noted that, this information is maintained on a per-tree basis and not on a per-domian basis. If we assume that we do not maintain the change in the QoS cost for the least level (which anyway would be 0), we have exactly $l-1$ service levels for which the change in QoS cost has to be maintained. For example, if the number of edge routers is 50, the number of distinct QoS levels is 4 and each entry in the table occupies 4 bytes, then in the worst case, the storage needed at the source for one multicast tree is 600 bytes (assuming we do not maintain the lowest QoS level).

V. Optimality Proof

In this section we prove that algorithm QMT-JL provides the minimum increase in cost of the existing multicast tree when a new member requesting any service level joins the multicast group.

Lemma 5.1 *Let T be an existing QMT. Let r be the receiver that wants to join the multicast group. Let the resulting tree be $T^{'}$. Then the join operation minimizes cost($T^{'}$) - cost(T). In other words algorithm QMT-JL finds the optimal attachment point for r upon every join operation without disrupting the service of on-tree multicast receivers.*

Proof There are two cases to consider. The optimal attachment point can have a service level greater than or equal to the member about to join. In this case the links of the existing tree need not be updated as they can satisfy the QoS requirement of the new member. The second case arises when the optimal attachment point has a lower service level than the new member.
Case 1 is optimal- In this case, according to algorithm QMT-JL all on-tree links from the source to the optimal attachment point are assigned a cost of zero because the service level of the new member can be satisfied by these links. Let the optimal increase in QoS cost be x and the optimal attachment point be p. Our algorithm finds the shortest path from the source to the new member r. This shortest path will definitely contain the node p because all the links from the source to p have an increase in QoS cost equal to zero. The shortest path will exit the tree at p and has a cost of x from p to r which in turn is the optimal increase in QoS cost when r wants to join the group. Thus, case 1 is optimal.
Case 2 is optimal- In this case on-tree links from the optimal point of attachment p towards the source have to be upgraded to reflect the change in service level as dictated by the new member. Lets call this change in cost as y and let x be the QoS cost from p to r. Since we are concerned only about the increase in QoS cost when a join occurs, we add y to x and then set y to zero. Thus, the QoS cost from p to r is now $x+y$ and the change in QoS cost from the source to p is zero. Thus, case 2 can now be treated as a special instance of case 1. Thus algorithm QMT-JL will also return p as the optimal attachment point for r. Hence, case 2 is also optimal.

Before, we proceed to simulation studies, we give a description of algorithm *Highest Level In First* (HLIF), a static QoS-aware multicast tree construction algorithm we proposed in [21]. We compare the performance of QMT-JL with HLIF in the next section.
Algorithm HLIF
HLIF builds a multicast tree by adding members in decreasing order of levels as follows. HLIF first considers receivers at the highest QoS level for addition. When all the highest service level nodes have been added, it considers all the nodes in the next lower service level and repeats the addition until all members of the multicast group have been added.

The intuition behind HLIF is that a path from the *Ingress node* to a higher service level node must be made as short as possible. Longer path lengths to higher service level nodes will incur more *QoS cost*. Therefore, we construct a static multicast tree by adding nodes in decreasing order of their service levels. Nodes of lower service levels will be attached to higher service level nodes and hence the *QoS cost* along that path will be low. Thus, we do not over-provision network resources along paths to receivers with lenient QoS requirements.

VI. Simulation Studies

In our simulation studies we compare the following three algorithms.

- QMT-JL : The member join/leave procedure we have proposed in this paper.
- HLIF (Highest Level In First) : The static multicast tree construction algorithm we proposed in [21].

- SSP (Source based Shortest Path) : This method consists of finding the shortest path (in terms of link cost) from the new member to the source and adding the corresponding path to form the new tree and then updating the QoS cost along the newly added path.

The various input parameters for the simulation studies are as follows:

- Random Network topologies were generated using “Average Degree” as the input parameter. This in turn determines the graph density. Higher the average degree, denser the graph.
- The default parameters are (a) Total number of nodes = 60 (b) Average number of group members = 15 (c) Average number of edge routers = 45 (d) Number of QoS levels = 3 (e) Link weight was varied between 25 and 80.
- The per unit QoS Cost was assigned as $C(l_i) = i$.

The performance metrics used to compare the various algorithms were *Tree QoS cost*, *Number of Service Disruptions*, *Storage Overhead* and *Running time per operation*.

The effects of the Number of QoS levels and the Join/Leave ratio were studied in our simulations.

Effect of Number of QoS Levels

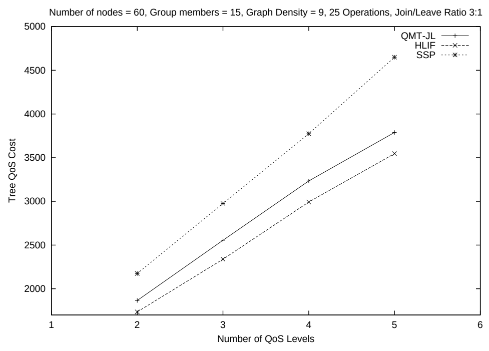

Fig. 4. QoS Cost vs Number of QoS Levels

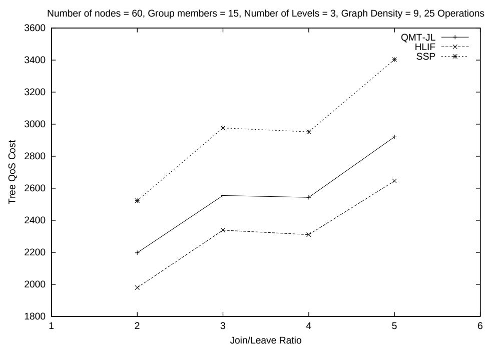

Fig. 5. QoS Cost vs Join/Leave Ratio

Tree QoS Cost-

Algorithm	TQC	SD	Storage	Time
HLIF	Best	Very Poor	No	Poor
QMT-JL	Fair	Best(0)	Yes	Fair
SSP	Poor	Best(0)	No	Best

TABLE I. Comparison Criteria for various Algorithms

Fig. 4 shows the Tree QoS Cost plotted against the number of QoS levels for algorithms HLIF, QMT-JL and SSP. It is clear that as the number of QoS levels increases, the Tree QoS cost increases for all the three algorithms because a higher service level is charged more than a lower service level. HLIF performs better than QMT-JL because it is a static tree construction algorithm and allows for service disruption. Again SSP performs worse than QMT-JL reiterating the fact that a source-based shortest path might always not be a good solution.

Effect of Join/Leave Ratio

Tree QoS Cost-

We plot the Tree QoS cost vs Join/Leave Ratio graph in Fig. 5 for the three algorithms. As the Join/Leave ratio increases, the Tree QoS cost increases due to increased number of joins which contribute to an increase in QoS cost.

Summary of Comparison of QMT-JL, HLIF and SSP

Table 1 compares the three algorithms on the basis of four criteria.

- Tree QoS Cost (TQC)- It is clear that HLIF has the least Tree QoS cost. QMT-JL is better than SSP but worser than HLIF because it does not allow for any service disruptions to occur. It is optimal when there are no service disruptions.
- Number of Service Disruptions (SD)- HLIF performs badly because the tree is reconstructed upon every join/leave operation. Both QMT-JL and SSP do not allow for service disruptions.
- Storage Overhead- QMT-JL is optimal when there are no service disruptions because it maintains some additional tree-related information at the source. SSP and HLIF do not incur any storage overhead for a successful join/leave operation to happen.
- Running time per operation- HLIF has the worst time complexity per operation as the entire tree is reconstructed again. SSP is the best because it just finds the shortest path between the source and the new member. QMT-JL also finds only one path always, but it has the overhead of updating the information it maintains after every operation.

VII. Conclusion

In this paper, we first formulated the member problem of join/leave in heterogeneous QoS multicasting in DiffServ networks and then proposed an optimal algorithm, QMT-JL. The algorithm is based on maintaining the change in QoS cost of an on-tree link when a new member joins. Member leave does not disrupt service of existing on-tree nodes. We also have compared QMT-JL, SSP and the static multicast tree construction algorithm HLIF with Average QoS Cost as performance metric.

There are some important issues meriting further research:

- Tree rearrangement- Since members join and leave the group based on locally optimal paths (e.g., in terms of cost), the global optimality of the tree will degenerate as more and more join/leave operations happen. This calls for tree rearrangement whose goal is to keep the cost of the tree near optimal with minimal service disruptions. Addressing this issue in the context of heterogeneous DiffServ QoS is an interesting future work.
- In the current form, our algorithm does not take into account minimizing the number of good neighbor instances. As a future work, we could develop a cost model and an algorithm that accounts not only for the Tree QoS cost but also takes into account the good neighbor cost.

References

[1] K. Nichols, S. Blake, F. Baker, and D. Black, ”Definition of the Differentiated Services field (DS filed) in the IPv4 and IPv6 headers,” *IETF RFC 2474*, Dec. 1998.

[2] S. Deering, ”Multicast Routing in Internetworks and Extended LANs”, ACM SIGCOMM Computer Communication Review, 1995.

[3] A. Striegel, and G. Manimaran, ”A scalable approach to DiffServ Multicasting,” in *Proc. of ICC'2001*, Helsinki, Finland, June 2001.

[4] A. Striegel, A. Bouabdallah, H. Bettahar, and G. Manimaran, ”EBM: A New Approach for Scalable DiffServ Multicasting”, in *Proc. of Network Group Communications (NGC)*, Munich, Germany, Sept. 2003.

[5] H. Takahashi and A. Matsuyama ”An Approximate solution for the Steiner problem in graphs,” *Math. Japonica*, vol. 24 (1980), pp. 573-577.

[6] L. Kou, G. Markowsky and L. Berman, ”A fast algorithm for Steiner trees,” *Acta Informatica* 15 (1981) 141-145.

[7] S. Ramanathan, ”Multicast tree generation in networks with asymmetric links,” IEEE/ACM Transcations on Networking, 4(4):558-568, November 1996.

[8] G.N. Rouskas and I. Baldine, ”Multicast Routing with End-to-End Delay and delay variations constraints,” IEEE INFOCOM'96, 1996, pp. 353-360.

[9] N.F. Maxemchuk, ”Video Distribution on Multicast Networks,” IEEE JSAC, April 1997, vol.15, no.3, pp. 357-372.

[10] B. Wang and Jennifer C. Hou, ”QoS-Based Multicast Routing for Distributing Layered Video to Heterogeneous Receivers in Rate-based Networks,” IEEE INFOCOM'00, 2000, pp. 480-489.

[11] G.L. Xue, ”Minimum-cost QoS multicast and unicast routing in Communication Networks,” *IEEE Transactions on Communications* 51(5): 817-824 May 2003.

[12] R. Sriram, G. Manimaran, and C. Siva Ram Murthy, ”Preferred link based delay-constrained least cost routing in wide area networks,” Computer Communications, vol.21, no.18, pp.1655-1669, Nov. 1998.

[13] S. Chen, K. Nahrstedt, and Y. Shavitt, ”A QoS-aware multicast routing protocol,” IEEE INFOCOM, pp.1594-1603, 2000.

[14] G. Manimaran, H. Shankar Rahul, and C. Siva Ram Murthy, ”A new distributed route selection approach for channel establishment in real-time networks,” IEEE/ACM Trans. Networking, vol.7, no.5, pp.698-709, Oct. 1999.

[15] A. Striegel and G. Manimaran, ”A scalable protocol for member join/leave in DiffServ multicasting”, in Proc. IEEE Local Computer Networks (LCN), Florida, USA, Nov. 2001.

[16] Yang-hua Chu, Sanjay G. Rao and Hui Zhang, ”A Case For End System Multicast”, Proceedings of ACM SIGMETRICS, Santa Clara,CA, June 2000, pp 1-12.

[17] V. Jacobson, K. Nichols, and K. Poduri, ”An Expedited Forwarding PHB,” Internet Draft, IETF, Nov. 1998.

[18] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, ”Assured Forwarding PHB Group,” Internet Draft, IETF, Nov. 1998.

[19] T.H. Cormen, C.E. Leiserson and R.L. Rivest, ”Introduction to Algorithms,” The MIT Press, Cambridge, Mass., 1990.

[20] A. Striegel, and G. Manimaran, ”Dynamic DSCPs for Heterogeneous QoS in DiffServ Multicasting,” in Proc. of IEEE GLOBECOM, Nov. 2002.

[21] A. Sai Sudhir, G. Manimaran and S. Tirthapura, ”Heterogeneous QoS in DiffServ Networks - Static Multicast,” Technical Report, Iowa State University, 2004.