International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:1, 2010

Supporting QoS-aware Multicasting in Differentiated Service Networks

Manas Ranjan Kabat, Rajib Mall, Chita Ranjan Tripathy

***Abstract*—**A scalable QoS aware multicast deployment in DiffServ networks has become an important research dimension in recent years. Although multicasting and differentiated services are two complementary technologies, the integration of the two technologies is a non-trivial task due to architectural conflicts between them. A popular solution proposed is to extend the functionality of the DiffServ components to support multicasting. In this paper, we propose an algorithm to construct an efficient QoS-driven multicast tree, taking into account the available bandwidth per service class. We also present an efficient way to provision the limited available bandwidth for supporting heterogeneous users. The proposed mechanism is evaluated using simulated tests. The simulated result reveals that our algorithm can effectively minimize the bandwidth use and transmission cost

***Keywords*—**Differentiated Services, multicasting, QoS heterogeneity, DSCP

I. Introduction

DURING the last years there has been an increasing deployment of multimedia applications. Many of these applications, like video broadcasting, distance education demand quite stringent quality of service (QoS) to provide smooth play-out at the receiver. Such requirements are not possible to meet with the current best-effort Internet. In order to provide QoS to users across the Internet, the limited bandwidth available to the users should be prioritized among users. A video conferencing application that tolerates minimal delay and minimal packet loss can be assigned the highest priority. Conversely, a web browsing application that tolerates maximum packet loss and has high delay tolerance can be assigned the lowest priority.

Apart from QoS assurances, another important aspect of the Internet usage is bandwidth utilization. Several evolving applications like World Wide Web (WWW), on-demand audio/video services and teleconferencing consume a large amount of bandwidth. Multicasting is a useful operation for supporting such applications. Multicasting enables group communication to simultaneously transmit messages from one source to a group of destinations and at the same time minimizing the total network bandwidth consumed [1]. Before transmitting message, most multicast routing protocols establish the packet delivery path from the source to all the destinations called a multicast tree for efficiently transmitting the message to individual destination nodes. When the same packet is transmitted to n different receivers, one traffic flow is created in the network instead of n different traffic flows. The same flow is replicated to the nodes that guide to different paths in order to reach the receivers. This way, the bandwidth use is minimized.

Although the bandwidth of the Internet is continually increasing, the backbone of the Internet itself is still far from being able to support QoS without appropriate resource allocation mechanisms. In addition, as the available bandwidth to end users increases, new applications are continually developed over the Internet would make it very difficult for service providers. So for the foreseeable future some form of resource provisioning is solely needed for provisioning QoS guarantee to end-users. One of the more promising models for providing QoS across the Internet is the Differentiated Services (DiffServ) model [2, 9]. The DiffServ model offers a number of QoS classes and traffic flows asking for the same QoS are served in an aggregate manner. A set of DiffServ CodePoints (DSCP) has been defined in the 6-bit DiffServ (DS) field of an IP header. Each DSCP is associated with a particular QoS class characterized by a particular packet forwarding treatment termed as Per-Hop Behavior (PHB).

From the perspective of both the end user and the network service provider, multicasting could offer tremendous benefit to both network efficiency and QoS. However, the issue of how to support multicasting in DiffServ has received relatively little research attention. Although the two concepts of bandwidth conservation (multicast) and scalable QoS management (DiffServ) are complementary, the emphasis on scalability by DiffServ creates architectural conflicts with multicasting that make the integration of two technologies a nontrivial task. The objective of our study is to construct a multicast tree so that the total bandwidth consumption in a multicast tree will be minimum. In our paper, we propose a QoS-driven multicast tree generation (QMTG) algorithm for the multicast tree generation and show that our algorithm consumes less bandwidth than Multicast Tree Calculation Algorithm (MTCA) [7].

The rest of the paper is organized as follows. In section II, we discuss the related works on supporting multicast in

Manas Ranjan Kabat is with the Veer Surendra Sai University of Technology, Burla, Sambalpur INDIA. (Corresponding author phone: +91-9861173326; e-mail: manas_kabat@yahoo.com).

Rajib Mall is with the Indian Institute of Technology, Kharagpur, INDIA. (e-mail:rajib@cse.iitkgp.ernet.in)

Chita Ranjan Tripathy is with the Veer Surendra Sai University of Technology, Burla, Sambalpur INDIA (e-mail: cse_uce@yahoo.co.in).

Differentiated Service networks. The proposed DiffServ multicast architecture is presented in section III. Section IV presents the proposed multicast tree generation algorithm including member join/leave and data-forwarding mechanism, we present the performance analysis of our scheme in section V, and finally a summary of conclusions is presented in section VI.

II. PRIOR WORK

Many researchers deal with the problem of integrating multicasting with DiffServ. The current solutions can be classified into four main categories: (a) State-based (b) Edge-based (c) Encapsulation based (d) Aggregate based.

The state based solutions require all the routers (edge and core) to store state information per multicast group. The solutions of this category do not scale well and also violates the basic principles of DiffServ logic. The QMD [3] (QoS aware multicasting in DiffServ domains) multicast routing protocol is state-based solution in which the core keeps the information about network topology and the available network resources for each link. Though this approach moves complexity to the edges, it still demands state information maintenance in some core routers.

Within the edge based solutions, the core routers are not multicasting capable and the multicasting functions are limited to the edge routers. Since no tree is created and no packet replications are performed in the core network, the bandwidth usage is not the most efficient. The most typical algorithm of this category is EBM (Edge-based Multicasting) [4], MMT (MPLS Multicast tree) [11] and ERM (Edge Router Multicast) [12]. The encapsulation-based solutions are based on the extension of the IP header of the multicast packets, in order to include the tree topology. The routers parse the IP header and decide whether to replicate the incoming multicast packets or not, determine the number of copies and select the proper DSCP value of the DS byte for each copy. This approach does not scale well because the IP header, and consequently the total length of the multicast packet, increases proportionally to the number of receivers. The main representative of this category are the DSMcast (DiffServ Multicast) [5, 10] and the XCAST [6] protocols.

Finally, the aggregate–based category tries to aggregate multicast trees into a limited number of super-multicast groups at edge routers, and then only have to service multicast for the super multicast groups. In this way, scalability issue can be well controlled in core routers without sacrificing the benefit brought by the existent IP multicast framework. The aggregate multicast frame proposed in [13] builds several permanent aggregate trees (AT) in the core (MPLS) network. A new multicast tree should find a super-set AT to join at ingress routers. After joining, packets are encapsulated with a new multicast address and distributed over the entire associated AT. The strategies about how to finding the best AT to join and when AT creation and adjustment should be made subject to QoS requirements. Clearly this scheme may force packets to be sent to some undesired branches, so a packet filtering mechanism should be applied at egress routers to filter out misled packets. Also the aggregation forces QoS control to be applied on a per-AT basis rather than per-tree basis. As a result some QoS over provision may happen to some trees in order to satisfy the QoS requirements of all the trees sharing the same AT. This thus leads to wastage of resources.

Taking into account the benefits and weaknesses of the above categories, a solution is proposed in [7] to extend the functionality of the DiffServ components without violating the basic principles of DiffServ architecture, in order to provide multicasting. The Multicast Tree Calculation Algorithm (MTCA) proposed in [7] finds a QoS-driven multicast tree. The data forwarding mechanism is similar to QMD and follows up the idea to further exploit some detours other than the least cost based routes, which satisfies specific QoS requirements of a multicast tree to improve the admission rate. However, in this model the centralized agent, Bandwidth Broker (BB), calculates the multicast tree and processes member join and leave requests.

III. MULTICASTING IN DS DOMAIN

The DiffServ model consists of two types of routers, edge routers and core routers. In order to support multicasting in DiffServ networks, the functionalities of these routers are extended. The core routers are involved only in high speed multicast routing and scheduling packets as per the DSCP in each packet, so that the simplicity of the core can be maintained. The intelligence of the DiffServ multicast network is migrated to the edge routers. The edge router is the key element for proper functioning of the DiffServ multicast networks. In this section, we present the idea to extend the functionality of the DiffServ network routers without violating the basic principles of the DiffServ architecture. The features added to these routers are as follows.

Core routers: - A multicaster module is install in all the core (and edge) routers of the domain. A multicaster receives unicast packets from its parent (source or multicaster) and forwards them to its children (multicaster(s) or receiver(s)). The multicaster sets appropriate DSCPs ($DSCP_1$, $DSCP_2$…DSCP*x*), where '*x*' be the number of traffic classes for each child multicaster before forwarding. Concerning the selection of DSCP values, the multicaster assigns to each replicated flow (directed to a subset of receivers) the DSCP value that corresponds to the highest service level for the receivers' requirements (e.g replicated packets that travel towards an Expedited Forwarding (EF) and a best effort receiver will be marked with EF DSCP). The required DSCP value for each unicast flow is calculated within the multicast tree calculation by the ingress router. Fig. 1 shows the QoS heterogeneity and DSCP assignment at each multicaster. The replicated packet at node B towards C destined to EF receiver A, EF receiver B and BE receiver C is marked with the EF DSCP. If the link does not have enough bandwidth for the

specified class then the edge router should select a different path, which satisfies bandwidth requirements of the specified service level. More than one multicaster may be running within a router, if the router is an intermediate node (router) for more than one multicast groups.

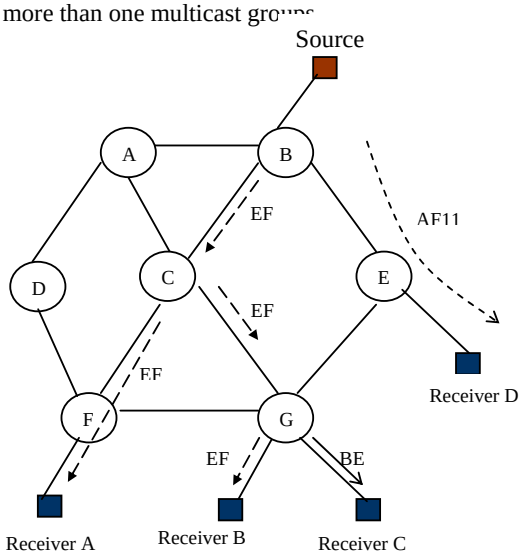


Fig 1. Multicasting in DS domain (Multicaster modules run in the nodes B, C and G)

Edge routers: - The edge router knows the exact network topology and is able to calculate the available bandwidth for each link in the DS domain. In this scheme, the link is organized in a hierarchical manner. Each physical link is statistically divided into multiple provisioned links (PL) and one PL is dedicated to one traffic class. Therefore, the number of PLs on a physical link equals the number of traffic classes that the physical link can support. A source edge router keeps track of the available bandwidth on each PL in a network and performs admission control without hop-by-hop signaling. Each edge router has a virtual IP path (VIP) table that records all virtual IP paths originating from the edge router all other edge routers. A virtual IP path (VIP) is a path from a source edge router to a destination edge router for a specific traffic class. A VIP table at an edge router has an entry for each of the VIPs originating from the edge router. An entry for a VIP path consists of VIP ID, the ID of the destination edge router, traffic class that VIP supports and a list of PLs constituting the path (Table 1).

TABLE I
A VIP TABLE

VIP ID	Destination Edge router	Traffic Class	list of PLs

A provisioned link (PL) table at an edge router records the bandwidth utilization of all PLs in a network. Table 2 shows the format of a PL table. An entry for a provisioned link consists of the PL ID, the ID of the parent node and the ID of the child node, traffic class it supports, available bandwidth and cost.

TABLE II
A PROVISIONED LINK TABLE

Provisioned Link ID	Parent node ID	Child node ID	Traffic Class	Available Bandwidth	Cost

TABLE III
GROUP MEMBERS TABLE IN EDGE ROUTERS

Group Address	Members
224.24.57.24	147.102.7.45-EF,147.102.8.89-BE, 147.23.34.23-AF11

A VIP table and a PL table may be pre-conFigured or constructed by routing mechanisms. The source edge router calculates the multicast tree and stores the relevant information. The source edge router uses this information to calculate the bandwidth consumption on the domain's links during multicast sessions. The multicast tree information is used to export information for the exact path that the multicast packets will follow the multicaster nodes. The edge router broadcasts the tree information as a tree object to all other edge routers so that the provisioned link table at each edge router can be updated. All the edge routers are assumed to be connected in a ring topology and a token always moves through the ring. The edge router when receives a multicast transmission request message, keeps the token and releases it after constructing the tree and broadcasting the tree information to all edge routers.

The (logical) topology of a single DiffServ domain comprising of n nodes and L links; each link $1 \le l \le L$ is supported to reserve capacity C_l to EF, AF, and BE class of service. Each node can be classified as either ingress/egress node if users are connected to that particular node, or core routers if no traffic is generated or directed to that node. The edge routers are responsible for the construction of the multicast tree. The problem of multicast tree can be described as: given a source node $s \in V$ and a set of destination nodes $D \subseteq V$ ($s \notin D$) with specific QoS level requirements specify the nodes and links of DS domain that can connect the source with all nodes in D, without violating the available resources, consuming totally the minimum bandwidth. The multicast tree is the sub tree of the graph G(V, E) rooted from s and the destinations are the set of leaf nodes. When multicasting a message to D, node s sends a copy of the message to its children in the tree. These children in turn transmit the message to their children until all nodes in D have received the message.

IV. THE PROPOSED ALGORITHMS

In this section we present a heuristic algorithm QoS-driven multicast tree generation algorithm (QMTG) to construct a multicast tree in which the total bandwidth consumption is less than MTCA [7]. For any multicast tree generation requests, it is assumed that the edge routers has the

information about all VIPs originating from that edge router to all other edge routers and also the bandwidth availability at all provisioned links. Our algorithm constructs the multicast tree by adding provisioned links one by one from the VIPs to an empty tree. This algorithm tries to choose a combination of all best paths from source edge router to destination edge routers so that the total bandwidth consumption will be minimum. We also present data forwarding mechanisms and algorithms to join a new member or free an existing member.

A. *Our QoS-driven Multicast Tree Generation (QMTG) Algorithm*

When the source wishes to transmit to a multicast group, sends a multicast transmission request message to its neighboring edge router. The edge router retrieves the group members from the group-members table shown in table 3. Then, the edge router finds a cost sensitive multicast tree utilizing a heuristic algorithm discussed in the following paragraph.

The multicast tree calculation algorithm aims to produce a cost sensitive multicast tree, which connects the source with the group's members. When a source requests a multicast transmission rate '*R*', the neighbor edge router updates the cost field of the PL table that it assigns a NORMAL_COST (e.g 100) if the required rate is less than or equal to the available PL's bandwidth for the specific service class, otherwise it assigns the INFINITE_COST (e.g 100,000) to it. The edge router classifies the receivers into three service classes: Expedited Forwarding (EF), Assured Forwarding (AF) and Best Effort Forwarding (BE). In our network model, we have L physical links and each physical link statically divided to *x* provisioned links where *x* represents the number of traffic classes that the network supports. The provisioned link belonging the physical link '*l*' and dedicated to service class '*m*' is assigned an *ID (l,m)*. The construction of a multicast tree is obtained as union of pre-computed paths according to any given metric, each one connecting the source node s to a particular destination *d*.

```
Function add_to_tree (T, PLij)
{
   // PLij belongs to the physical link 'i' and dedicated for service class j //
   if ( PLiq ∈ T ){ // provisioned link PLiq is already a member of multicast tree //
        if ( j < q) // If PLij is a higher class provisioned link than PLiq //
          T = {T– PLiq } ∪ PLij  //then remove PLiq and add PLij to tree T //
      //else do nothing //
   }
   else // No provisioned link belonging to physical link i is a member of tree T //
        T = T ∪ PLij // Add PLij to tree T //
}

Function create_tree( )
{
   Ts(0) = ∅,    // Initialize tree in the first iteration to null//
    n=0;
  // construct the multicast tree by combining first VIPs from source s to all destination in D //
   for all (d ∈ D)
      add_to_tree ( Ts(0), PLij ∈ VIP1 s →d)
   opt[d] = 1
   cost(Ts(0)) = Σ PLij∈Ts(0) cost(PLij)
   Ts* =Ts (0)
  For ( n = 1; n ≤ Z; n++)
  {
    for all (d ∈ D) {
       while (the ith VIP from s to d is considered )
       {
            Ts (n) = ∅
             // add the PLs of ith VIP to tree Ts(n)
                for destination d //
           Ts(n) = Ts(n) ∪ { PLmn ∈VIPis→d)
          // add the PLs from the optimal VIPs
            for the remaining destinations //
          For all ( j ∈ D–{d})
              add_to_tree (Ts (n), PLpq ∈VIPopt[j]s→j)
           cost(Ts(n)) = Σ PLmn∈Ts(n) cost(PLmn)
           if (cost(Ts(n) < cost (Ts*) ) {
                 Ts* = Ts(n)
           opt[d] = i
           }
       } // end while //
    }// end for all //
  }// end for //
}// end function //
```

The proposed algorithm is used to compute several trees and select among them the one, which minimizes the total bandwidth consumption. Let { $VIP^{i}_{s\to d}$, i= 1,2…K} be an generic order set of K virtual IP paths pre-computed between s and d using hop count as the metric. So the virtual path number 1 is the shortest path. Let $T_s(n)$ be the multicast tree generated in iteration n. For each destination an optimal path is selected and $T_s(n)$ is then obtained as the union of all PLs in $VIP^{opt}_{s\to d}$. Therefore, different solutions are obtained by selecting different set of paths.

The algorithm proposed here tries to efficiently explore the state space of a possible set of VIP paths, whose number grows as a combinatorial function of the number of virtual paths. Instead of considering all possible combinations of paths, at each iteration some sensible solutions are tested, each one obtained by simply changing one path at a time i.e. for a given destination d. The trees obtained by considering all virtual paths, $VIP^{i}_{s\to d}$, are tested. This defines the solution that differs from the previous one by a single path. At the end of the iteration, the best solution is selected. A maximum number of iterations Z is defined to limit the complexity of the algorithm.

B. *Our QoS-driven Multicast Tree Installation (QMTI) Algorithm*

When the multicast tree is constructed, the source edge router encapsulates it in a signaling packet and broadcasts it to other edge routers. The edge routers after getting the multicast tree object, updates the available bandwidth field of the PL. Then the source's edge router releases the token so that any other edge router can catch it before constructing the multicast tree. Once the source's edge router constructs the multicast tree, it installs the multicast tree. Specifically, the tree installation refers to the multicatsers' launching and the tree node's routing table update. Concerning the multicasters launching, the source's edge router sends to the key nodes a message to start multicaster. By recording the service class values of the provisioned links in multicast tree, the on tree router knows the parent multicaster (PM) and children multicasters (CM) with the corresponding DSCPs ($DSCP_1$, $DSCP_2$, … DSCPx) for each multicaster. On the other hand, the routing table of all the tree nodes should be confirmed to the tree structure meaning that the multicast packets should follow the paths that are defined by the source's edge router, which are not necessarily the shortest path but definitely include the links with the required resources.

Once the tree installation finalizes, the edge routers to the source the "multicast tree request confirmed" (MTRC) message, in order to start transmission or send a "multicast tree request not confirmed" (MTRN) in case of failure.

C. Data Forwarding Mechanism

If the source receives an affirmative message "MTRC" from the edge router, it starts the multicast transmission. The edge node shapes/polices the incoming packets and forwards them to the local multicaster. The local multicaster sends one unicast flow to each child multicaster with the proper DSCP. We assume that x numbers of DSCP are available for marking packets. Each router maintains a multicast-forwarding table for forwarding packets. Forwarding table has a data structure of {s_add, m_add, PM_id, CM_list} (Fig. 2.), where s_add and m_add indicate source that transmits packet and multicast address, which packets address to respectively. The source address s_add is defined by four octets (32 bits) separated by dots (.) in IPv4 and the multicast group address m_add, distinguished by 24 bits in IPv4. The PM_id indicates the id of the parent multicaster from which the packets were received. The CM_list has a data structure of {CM_id, DSCP_id}, where CM_id and DSCP_id indicate the id of the child multicaster and the corresponding DCSP respectively.

Packet forwarding is taken place as follows. Upon receiving a packet addressing to m_add from parent multicaster 'in', the router seeks for the entry in forwarding table with PM_id = in and m_add = m, then the router obtains the corresponding CM_list and knows the CM_id with the corresponding DSCP.

Considering the example of Fig 3(a), let the multicaster implemented at core node C receives a packet 'P' with the multicast address m_add, the packet is replicated at node C and forwarded to the children multicasters (CM_1, CM_2, … CM_k) marking the packets with the corresponding DSCPs. On the other hand, the multicaster at node C receives packets P and Q with the multicast address m_add and m_add′ respectively (Fig 3(b)), then the packets forwarded to the same child multicaster with same DSCP can be treated aggregately at node C.

s_add	m_add	PM_id	CM_list	
			CM_id	DSCP _id
..*.*	224.*.*.*	in	c1	d1
			c2	d2

Fig. 2. Multicast forwarding table structure

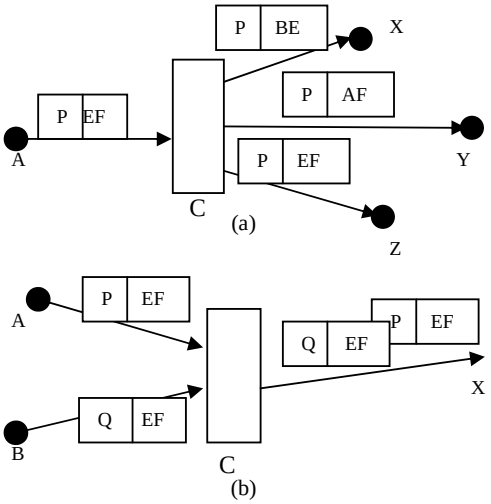


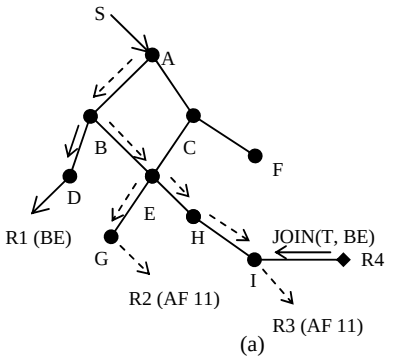
Fig. 3. Packet forwarding behavior at multicasters (a) data replication (b) data aggregation

D. Member Join

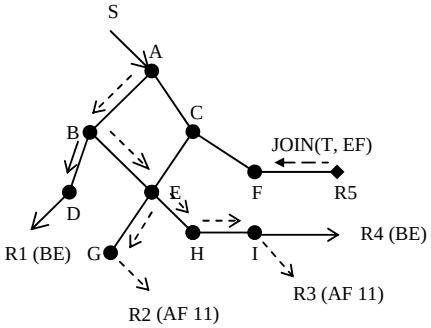
When an end user 'r' wants to join the multicast tree 'T' rooted at source edge router E_s with a desired QoS class 'Q_i', it sends a message JOIN(T,Q_i) to its corresponding edge router 'E_r' (IGMP [8], SSM [16]). If the receiver edge router is already a member of the multicast group and has a QoS state Q_j where $Q_j \geq Q_i$ then E_r sends a join acknowledgement message JOIN-ACK(T,Q_i) to the user. Otherwise, the receiver edge router forwards the join request to the source edge router. The source edge router checks the bandwidth availability for that request. If successful, then E_s sends a message JOIN-ACK to the end-user via E_r. If the source edge router finds that there is not sufficient bandwidth for the desired QoS level invoked by the join request, the user may adaptively choose to select a lower QoS level. On receiving the join request from E_r, E_s waits for the token. Then the edge router tests all feasible virtual paths from E_s to E_r and selects the best path so that the total bandwidth consumption over the multicast tree is minimum. An overview of the algorithm for the join procedure is presented below,

For each VIP from E_s to E_r , the provisioned links that are not with in the multicast tree by calling add_to_tree procedure discussed in section 3. The cost incremented after adding the new VIP is the sum of the costs of newly subscribed provisioned links minus the sum of the costs of provisioned links unsubscribed. The VIP, whose inclusion results the

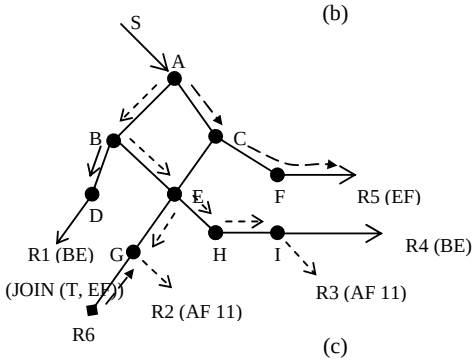
minimum cost incremented will be considered. Then the multicaster at each node in the newly added VIP can be installed as follows. For each new provisioned link, at parent node, the new child multicaster with the corresponding DSCP is added in the fields of CM_id and DSCP_id of the forwarding table respectively. At the child node, if the child has an old parent then the PM_id field is updated with its new parent and the provisioned link from old parent to the current node is unsubscribed. This information is propagated to its old parent at which it will find the highest DSCP at its child multicasters and the provisioned link from its parent multicaster is either unsubscribed or updated. This process is repeated till the source edge node is encountered. The PLs subscribed and unsubscribed are encapsulated in an object and broadcasts to other edge routers. Then it releases the token.



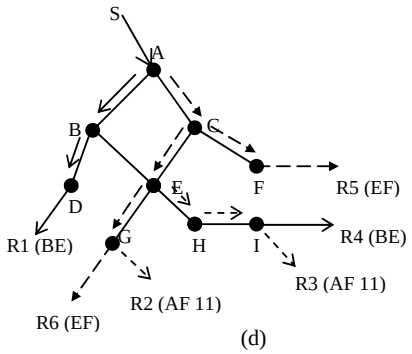
(a)



(b)



(c)



(d)

Fig. 4. Provisioned link subscriptions or unsubscriptions for multicast tree join requests.

In Fig. 4, we assume that initially there already exists a multicast tree connecting the source S to three receivers R1, R2 and R3 with provisioned links for BE and AF11 services. After some time the edge router I receives a join request with BE service from user R4 (Fig 4a). In this case, since I is already a member of the multicast group with QoS state AF11, i.e a state greater than requested service, it sends a JOIN-ACK to the user R4.

In Fig. 4b, we assume that the user R5 sends a join request with EF service to its corresponding edge router F. In this case F is a not a member of the group, so the request is forwarded to the source edge router A. Then router A finds the best virtual IP path for which the bandwidth consumption be minimum is{<A, C> <C, F>}.

In Fig. 4c, we assume that the edge router G receives a join request from the user R with EF service. In this case G is a member of the multicast group with QoS state AF11 i.e less than the requested service class. So the request is forwarded to the source edge router A. Then the router A finds the best VIP path from A to G for which the total bandwidth consumption can be minimum is {<A, C><C, E><E, G>}. The newly added provisioned links are $<C, E>_{EF}$ and $<E, G>_{EF}$. For the provisioned link $<C, E>_{EF}$, the parent node is C and the child node is E. At parent node C, the new child multicaster E is added in the CM_id field with EF DSCP in the DSCP_id field. At the child node E, the PM_id field is updated from B to its new parent multicaster C and the provisioned link $<B, E>_{AF11}$ is unsubscribed. This information is propagated to the old parent of E i.e. B. The highest QoS level at the child multicasters of B is BE. So the provisioned link $<A, B>_{AF11}$ is unsubscribed and $<A, B>_{BE}$ is subscribed. At router B, the child multicaster E is removed from the CM_id field of F.T. This information is propagated to the router A and since A is the root of the multicast tree, the DSCP_id of the child multicaster B is updated from AF11 to BE. Another newly added provisioned link is $<E, G>_{EF}$. At the parent node E, since G is already there in the CM_id field of the forwarding table, only the corresponding DSCP_id field is updated from AF11 to EF. The final multicast tree generated is shown in Fig. 4d.

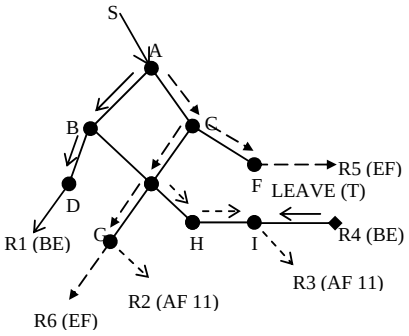
E. Member Leave

When an end-user subscribed with QoS level Q_i wants to leave the multicast group, it sends a request LEAVE(T, Q_i) to its corresponding edge router. If the edge router is at a QoS state Q_j such that $Q_j > Q_i$ then the leave procedure terminates. Otherwise, the edge router waits for the token and processes the leave request as follows.

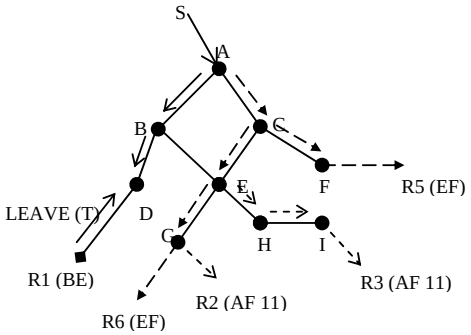
The egress router checks its current QoS state after the unsubscription of QoS channel from edge router to receiver. Then the router unsubscribes the existing provisioned link from its parent multicaster and subscribes a new provisioned link for the desired QoS level. The PL subscription and unsubscription is done at all the upstream routers by checking the available QoS state at that node.

We follow the example in Fig. 5 to illustrate the PL subscription and unsubscription procedure. In Fig. 5a we assume that the receiver R4 sends the message LEAVE(T, BE) to egress router I. Since the edge router I is in a QoS state of AF11 which is greater than requested QoS level, the leave procedure terminates. .

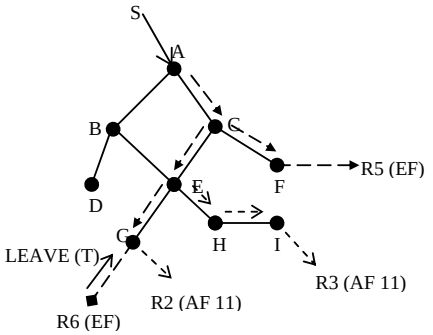
Let us assume that the router D receives a leave request LEAVE(T, BE) from receiver R1. In this case it is noticed that there is no subscriber attached to router D after releasing the user R1. So the provisioned link from B, the parent multicaster of D, to D $<B,D>_{BE}$ is unsubscribed. Similarly, it is seen that there are no more child multicasters connected to router B; so the PL from A to B $<A,B>_{BE}$ is unsubscribed and the tree is shown in Fig 5c. In Fig.5c, we assume that receiver R6 sends a message LEAVE(T, EF) to edge router G. Since the router G is in a QoS state EF that is equal to the QoS level of the unsubscribed user, it downgrades it QoS state from EF to AF11. Meanwhile, the provisioned link from E to G $<E,G>_{EF}$ is unsubscribed and the new PL $<E, G>_{AF11}$ is subscribed. At router E it is noticed that the highest QoS level at its children multicasters is downgraded to AF11, the PL $<C, E>_{AF11}$ is subscribed after the unsubscription $<C, E>_{EF}$. When it is noticed that at router C, the QoS requirement is not changing the leave procedure terminates. The final multicast tree is shown in Fig. 5d.



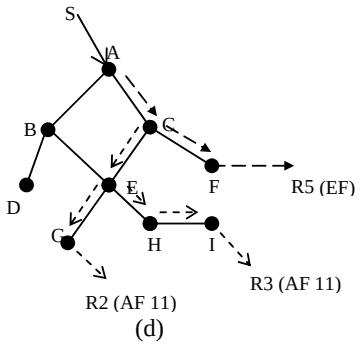
(a)



(b)



(c)



(d)

Fig. 5. Provisioned link subscription or unsubscriptions for multicast tree leave requests.

V. SIMULATION RESULTS

In this section, we examine the efficacy of our algorithm through simulations. We simulated the proposed architecture using java (JDK1.5.0). A Microsoft Windows (Windows XP) PC with CPU speed 2.4 GHz (Pentium 4) and 1024 MB RAM memory was used for all the performance tests. In this model n nodes are randomly scattered on a rectangular applet window of size 800X600 pixels, where each node is located on an integer coordinate. We set the boundary of 100 pixels from each side of the applet window. The nodes that are located in the boundary are designated as edge routers and the remaining nodes are noted as core routers. In our experiment, we found approximately 40% of the total number of nodes are

the edge routers. The source and destinations are selected randomly from the edge routers. The Euclidean metric is then used to determine the distance between each pair of nodes. The network model G(V,E) employed by our computer simulations comes from the one proposed in [15]. As far as any two nodes *v* and *w* on the network are concerned, the probability of forming a link between them is $(a_1 \exp(-dist(v,w)/La_2)$, L represents the maximum distance between any two nodes , i.e. $L=max_{v,w \in V} \{dist(v,w)\}$. The values of a_2 and a_1 fall between (0, 1]. If a_1 becomes larger, there will be more links on the network. If a_2 becomes larger, there will be longer links on the network; that is there will be fewer shorter links.

In order to control the quality of the network, we introduce an additional control parameter P to be used with the probability of forming a link. If the probability of forming a link is greater than or equal to P, then there will be a link between the two nodes. In our simulation, the parameters a_1, a_2 and P are set 0.6, 0.8 and 0.4 respectively. The bandwidths of the links are set randomly from 1 to 10. Each link is statically provisioned 30% of the available bandwidth for EF class, 40% for AF class and the remaining 30% for BE class. The destination nodes in the multicast group will occupy 10, 20, 30 and 40% of the overall nodes on the network, respectively. The destinations and their corresponding classes are selected randomly. The packet transmission cost ratio for EF: AF11: BE PHBs are 10: 6: 2 i.e the cost of carrying each bit of data using EF PHB is 5 times that of BE PHB. Simulations using the java implementation described above have been carried out with same traffic conditions for both MTCA [7] and our proposed heuristic algorithm QMTG. The flow rate of the traffic is randomly selected within the range [0,3]. The simulation results for both the algorithms are presented in the following format. First, the total bandwidth consumption and second the total transmission cost over the multicast tree.

A. Performance Analysis

First of all, we investigate bandwidth conservation performance, and comparisons are made between MTCA and QMTG algorithm. We calculate the bandwidth consumption as the sum of the bandwidths consumed in all the provisioned links over the multicast tree. Similarly, the total cost is calculated as the sum of the transmission costs over all the provisioned links in the multicast tree.

In order to evaluate the network utilization, we define the percentage bandwidth gain (BG) for the multicast tree T as follows.

$$BG = (1 - \frac{U^T_{QMTG}}{U^T_{MTCA}}) \times 100$$

Where U^T_{QMTG} is bandwidth consumption of tree T generated by QMTG algorithm and U^T_{MTCA} is that by using MTCA. Similarly, to evaluate the cost savings, we define the percentage gain in transmission cost (CG) for the multicast tree as follows.

$$CG = (1 - \frac{C^T_{QMTG}}{C^T_{MTCA}}) \times 100$$

Where C^T_{QMTG} is the total transmission cost over multicast tree T generated by QMTG algorithm and C^T_{MTCA} is that by using MTCA.

TABLE IV.
COMPARISON OF BANDWIDTH CONSUMPTION AND TRANSMISSION COST FOR QMTG AND MTCA. NETWORK SIZE IS 50 NODES.

No. Of Des	Flow rate	Bandwidth (BW) Consumption		% Gain in BW	Cost		% Gain in cost
		MTCA	QMTG		MTCA	QMTG	
5	0.71	07.819	05.686	27.28	62	40	35.48
10	2.70	59.435	54.030	09.09	120	104	13.33
15	2.09	60.630	43.900	27.59	178	126	29.21
20	1.42	58.386	44.145	24.39	270	206	23.70
25	0.92	32.440	26.879	17.14	228	188	17.54

TABLE V.
COMPARISON OF BANDWIDTH CONSUMPTION AND TRANSMISSION COST FOR QMTG AND MTCA. NETWORK SIZE IS 100 NODES.

No. Of Des	Flow rate	Bandwidth (BW) Consumption		% Gain in BW	Cost		% Gain in cost
		MTCA	QMTG		MTCA	QMTG	
10	0.81	16.269	13.829	14.99	108	88	18.51
20	0.90	28.828	24.324	15.62	158	124	21.51
30	1.90	97.135	64.109	34.00	228	192	15.79
40	1.61	116.205	88.767	23.61	484	378	21.90
50	1.06	81.698	68.967	15.58	412	336	18.44

TABLE VI.
COMPARISON OF BANDWIDTH CONSUMPTION AND TRANSMISSION COST FOR QMTG AND MTCA. NETWORK SIZE IS 150 NODES.

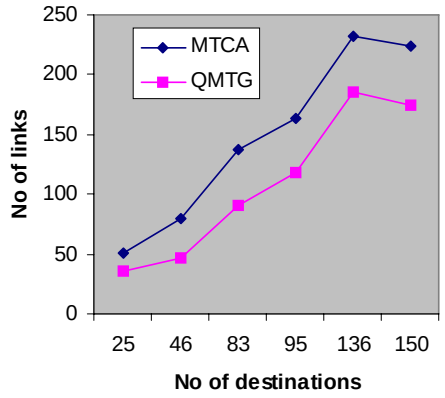
No. Of Des	Flow rate	Bandwidth (BW) Consumption		% Gain in BW	Cost		% Gain in cost
		MTCA	QMTG		MTCA	QMTG	
15	2.48	69.636	59.688	14.28	136	108	20.59
30	1.75	87.660	71.881	18.00	302	246	18.54
45	1.78	156.710	117.53	25.00	476	340	28.57
60	1.22	133.269	109.70	17.68	694	540	22.19

TABLE VII.
COMPARISON OF BANDWIDTH CONSUMPTION AND TRANSMISSION COST FOR QMTG AND MTCA. NETWORK SIZE IS 200 NODES.

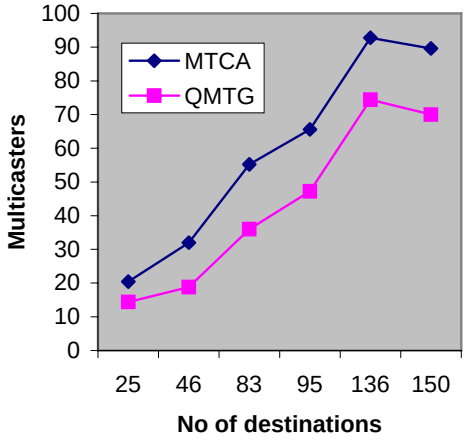
No. Of Des	Flow rate	Bandwidth (BW) Consumption		% Gain in BW	Cost		% Gain in cost
		MTCA	QMTG		MTCA	QMTG	
20	2.12	53.118	40.369	24.00	154	114	25.97
40	1.94	118.18	83.306	29.50	338	228	32.54
60	2.02	222.07	179.67	19.09	686	586	14.57
80	1.22	156.99	130.22	17.05	800	664	17.00

Table IV illustrates the bandwidth consumption and total transmission cost for multicast trees generated by QMTG algorithm and MTCA. The percentage gain in bandwidth and cost is also shown in table 4 for a network of 50 nodes. Tables

V, VI and VII illustrate the same performance metrics for networks of 100, 150 and 200 nodes respectively. The experiments with random Waxman grids confirmed that our algorithm provides better performance than MTCA [7] to the tree optimization problems.



(a)



(b)

Fig. 6: (a) the number of links of the produced multicast trees for MTCA and QMTG algorithm. (b) the number of running multicasters of the produced multicast trees for MTCA nad QMTG algorithm.

Then, to find the number of links and number of multicasters produced by different multicast tree generation algorithms, we again consider graphs with 50, 100, … 300 nodes. The boundary nodes are selected from java applet boundary as discussed in section V. One node is randomly selected from the boundary nodes as source and the remaining nodes are the destinations. We run different multicast tree generation algorithms to evaluate the performance in terms of number of links and number of running multicasters. The number of links with respect to the number of destinations for various heuristic algorithms is shown in Fig 6(a) and the number of running multicasters versus the number of destinations is plotted in Fig 6 (b).

VI. Conclusion

We have presented an architecture for supporting QoS-aware multicasting in Differentiated Service networks. In our approach, the edge routers perform most of the processing and multicast operations leaving the core routers stateless. Our approach supports heterogeneous QoS inside the DiffServ domain by statically multiplexing the bandwidth of the physical link among all the classes that the network supports and implementing the multicasters.

Moreover, we introduced a heuristic algorithm for the multicast tree generation. QMTG aims to solve the Steiner tree problem with the links available bandwidth constraints. The multicast tree generated by QMTG algorithm consumes less bandwidth than that of MTCA. Furthermore, we analyze the transmission cost over the multicast trees generated by both MTCA and QMTG algorithm and realize that the transmission cost over the multicast tree generated by QMTG algorithm is less than that of MTCA. We have also proposed an approach for providing scalable join/leave for DiffServ multicast networks. Furthermore, we evaluate the number of links and number of multicasters for a varying number of receivers.

References

[1] K.C.Almeroth, “The evolution of multicast”, IEEE Networks 14(1) 2000. pp.10-21.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated services,” IETF RFC 2475, December 1998.
[3] Z Li, P. Mahapatra, “ QoS-aware multicasting in DiffServ domains” Proc. Of Global Internet Symposium 2002.
[4] A. Striegal, A. Bouabdallah, H.Bettahar, G. Manimaran, “EBM: Edge based multicasting in DiffServ networks”, in Proc. Of Network Group Communications (NGC), Munich, Germany, Sep’ 2003.
[5] A. Striegal, G. Mannimaran, “DSMCast: A Scalable approach for DiffServ multicasting”, Computer Networks 44(6) (2004) 713-735.
[6] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, Explicit multicast (Xcast) basic specification, IETF Internet Draft <draft-ooms-xcast-basic-spec-05.txt> Aug’ 2003.
[7] S. Vrontis, E. Sykas, Extending differentiated services architecture for multicast provisioning, Computer Networks 48(2005) pp 567-584.
[8] W. Fenner, IGMP Internet Group management protocol, version 2, RFC 2236, Nov 1997.
[9] K. Nichols, S. Blake, F. Baker and D.L. Black, “ Definition of the Differentiated Services Field (DS field) in the Ipv4 and Ipv6 Headers, Dec 1998. IETF RFC 2474.
[10] A. Striegal, G Manimaran, Dynamic DSCPs for heterogeneous QoS in DiffServ multicasting in: IEEE GLOBECOM, 2002.
[11] B. Yang, P. Mahapatra, Multicasting in MPLS domains, Computer Communications 27(2)2004 pp162-170.
[12] A. Boudani, B. Cousin, J. Bonnin, An effective solution for multicast scalability: The MPLS multicast tree (MMT), IETF Internet Draft draft-boudani-mpls-multicast-tree-09.txt, March 2004.
[13] A. Fei, J.Cui, M. Gerla, M. Foloustos, Aggregate multicast: An approach to reduce multicast state, Proceedings of Sixth Global Internet Symposium (GI 2001), 2001
[14] J.H. Cui, J. Kim, A. Fei, M. Faloustsos, M. Gerla, Scalable QoS: Multicast provisioning in DiffServ supported MPLS networks in Proc of IEEE GLOBECOM 2002, Taiwan, Nov’ 2002.
[15] S. Floyd, V. Jacobson, The synchronization of periodic routing messages, IEEE/ACM Transaction on networking 2 (1994) 122.
[16] H.Holbrook, B. Cain, “ Source Specific Multicast for IP”,IETF Internet Draft. Draft-ietf-holbrook-ssm-arch-00.txt.

Manas Ranjan Kabat has completed B.E in Electrical Engg and M.E in Computer Engg from Utkal University, India and Bengal Engineering College, India in 1998 and 2000 respectively. He has submitted his Ph.D

thesis to Sambalpur University, India in 2009. He is now working as a Senior Lecturer in Comp. Sc. & Engineering in Veer Surendra Sai University of Technology, Burla, India. His current research interests include QoS routing, Reliable multicasting and TCP performances. He has published about six research papers in various referred international journals and conferences.

Rajib Mall has completed his M.Tech and Ph.D from Indian Institute of Science, Bangalore. He is currently working as a professor in the Department of Computer Science and Engineering in Indian Institute of Technology, Kharagpur. He has published more than 100 research papers in referred journals and conferences. His research interests include QoS support in Wired and Wireless networks, Real-Time systems and Software Engineering. He has also authored two books named as Real-Time Systems and Software Engineering published by Pearson Education and Prentice Hall, India respectively.

Chita Ranjan Tripathy has completed his M.Tech and Ph.D from Indian Institute of Technology, Kharagpur. He is currently working as a professor in the Department of Computer Science and Engineering in Veer Surendra Sai University of Technology. He has published more than 40 research papers in referred journals and conferences. His research interests include reliability, fault tolerance and computer networks. He is a fellow of Institute of Engineers, India and life member of I.S.T.E, I.S.I and O.I.T.S.