DIFFSERV OVERLAY MULTICAST FOR VIDEOCONFERENCING

A Thesis
Presented to
The Faculty of the Graduate School

University of Missouri – Columbia

In Partial Fulfillment
of the Requirements for the Degree

Master of Science

---

by
PRASAD J. WAGH

Dr. Michael Jurczyk, Thesis Supervisor

DECEMBER 2005

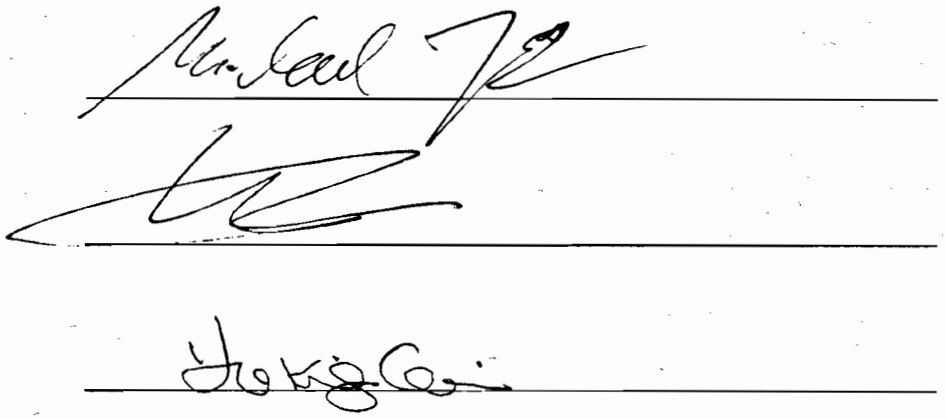
The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation titled:

DiffServ Overlay Multicast for Videoconferencing

presented by

Prasad J. Wagh

a candidate for the degree of **Master of Science** in **Computer Science** and hereby certify that in their opinion is worthy of acceptance.



ACKNOWLEDGEMENT

I would like to express my sincerest gratitude to all who helped me successfully complete my Masters. I am grateful to my parents for their constant support and motivation and for instilling in me an ardent respect for higher education. My sister Pooja provided valuable advice and impetus and helped me make important career decisions. I would also like to take this opportunity to recognize my advisor, Dr. Jurczyk as the most important person in my career as a graduate student. He took valuable time out of his schedule each week to discuss my thesis and provided best possible direction and guidance I could have hoped for. He helped me push myself and an inch at a time towards improvement. Thanks to his relentless efforts I was able to express my thoughts and work as articulately as possible in this document. To Dr. Jurczyk and all others that were concerned, I express my sincerest appreciation and gratitude.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DiffServ Overlay Multicast for Videoconferencing

Prasad J. Wagh

Dr. Michael Jurczyk, Thesis Supervisor

ABSTRACT

This dissertation proposes a multicast overlay framework to support videoconferencing in differentiated services network. Internet started as an experimental network for researchers and primarily was used as a medium for applications like file transfer and email. Over the years there has been tremendous growth in technology and it has given rise to new applications. Some of the modern real-time applications like videoconferencing involve group communication and require certain guarantees in terms of available network bandwidth and end-to-end latency. Two different technologies Differentiated Services (DiffServ) and IP Multicast have been proposed. DiffServ provides service differentiation in the network while IP Multicast preserves network bandwidth for group communications. The integration of the two technologies is however a nontrivial task for the contrasting nature of the two technologies.

In this dissertation a framework for implementing videoconference in a DiffServ network (DiffServ Overlay Multicast) is proposed. DiffServ Overlay Multicast uses an overlay multicast algorithm for calculating the multicast tree for distributing the video in a DiffServ network. By moving tree construction, packet replication and network monitoring to the edge routers, a stateless core is maintained. This dissertation also addresses all the issues concerning the integration of DiffServ and IP Multicast. In addition the issue of traffic engineering and load balancing in the network to improve the quality of videoconference is also addressed.

Chapter 1 - Introduction

1.1 Problem Statement

Internet started as a small experimental network called ARPANET funded by Defense Advanced Research Projects Agency (DARPA). The network was mainly used for research and supported early applications like email and file transfer (ftp) which did not have any bounds on network bandwidth or delay. Over the years Internet has become a strong medium for widespread communication and as advancements are made in technology every year, new applications emerge that have diverse requirements. The current Internet model, which provides best effort service, is inadequate for supporting modern applications. The emerging continuous media Real Time applications like video conferencing demands more stringent requirements than traditional TCP-based applications like the ftp. Three key problems can be identified while implementing a videoconference over the current Internet.

The first problem for implementing a videoconference is that a video stream requires certain minimum amount of network bandwidth and the video packets must reach the destination within a certain amount of time to maintain the video quality. In other words video streams require certain guarantees in terms of bandwidth and latency in order to maintain end-to-end video quality. On the other hand an ftp application can tolerate delay and resending of lost packets without degrading its performance. Hence depending upon the requirements of an application special treatment should be provided. But the current Internet model treats all the packets the same way and hence offers only a single level of service called the best effort service. An alternative solution to this problem is to provide service differentiation in the network and treat packets from

different applications differently depending upon their requirements in other words provide Quality of Service (QoS). The Internet Engineering Task Force (IETF) has proposed Differentiated Services Architecture to provide service differentiation in the network.

The second problem is that in a videoconference there are many senders and receivers involved. It is practically impossible to send individual video streams from a single sender to all the receivers. Hence the obvious solution to this problem is to use IP multicast in which a sender sends out a single video stream, which can be replicated in the network to reach all receivers.

The third problem is that there is no traffic engineering in the current Internet. In normal IP networks, shortest path algorithms are used to forward traffic from source to destination. This may cause congestion on some links while leaving some other links in the network under-utilized. Shortest path between the source and the destination may not always be the best path in terms of available link bandwidth. Shortest paths often might be congested while there might be alternate paths between the source and the destination that are underutilized. The solution to this problem is to implement traffic engineering in the network. The purpose of traffic engineering is to optimize network resources and performance. Traffic engineering is important for providing QoS in the Internet because Diffserv essentially provides differentiated performance degradation for different classes of traffic during network congestion. Hence in case of network congestion the performance of low priority classes is highly degraded. But if alternate routes are provided the traffic can be distributed to reduce performance degradation. Traffic engineering is also needed to prevent concentration of high priority traffic. If there is

concentration of high priority traffic, when there is network congestion, and a new DiffServ connection is requested along the congested path, the connection will be rejected for lack of network resources. Hence to optimize the performance of a DiffServ network traffic engineering must be implemented. One approach to implement traffic engineering would be to develop intelligent routing algorithms that are able to sense network congestion and provide alternate routes for forwarding traffic.

To overcome the problems associated with implementing videoconference in the current Internet, it is necessary to implement IP Multicast in a DiffServ domain. However the integration of the two technologies is a non-trivial task and is associated with different problems, which are discussed, in the next chapter.

1.2 Contribution

This thesis proposes a framework (Diffserv Overlay Multicast) to implement videoconferencing in a DiffServ domain. The idea is to construct an overlay multicast tree for distributing video to the different receivers. Diffserv Overlay Multicast uses an overlay multicast algorithm for calculating the multicast tree for distributing the video in a DiffServ network. The overlay tree consists of IP tunnels connecting different edge routers. Packet replication is allowed only on the edge routers. The core routers do not replicate any packets. By moving tree construction, packet replication and network monitoring to the edge routers, a stateless core is maintained which abides by the rules defined in the DiffServ Architecture. This dissertation addresses the problems associated with implementing videoconference in the current Internet. In addition all the issues concerning the integration of DiffServ and IP Multicast are also addressed.

The work is divided into two areas. First the thesis suggests enhancements for the base overlay algorithm in order to improve the quality of video. The enhancements to the algorithm show a sizable improvement in terms of reducing the total packet loss during the videoconference.

Next the thesis proposes an extension of the enhanced base algorithm to support videoconferencing in a DiffServ domain. The performance of DiffServ Overlay Multicast is studied with extensive simulation results. In addition the thesis also compares the performance of DiffServ Overlay Multicast with existing DiffServ Multicast Algorithms.

Chapter 2 – Related Works

2.1 IP Video Conferencing Technology

A basic IP videoconference setup involves two or more people located in different physical locations trying to converse with each other over the Internet. The idea is to use a video camera for recording the audio and video, connect it to a computer and then transmit the video over the Internet to different people. If the videoconference is used for medical purposes where a team of doctors located at different locations is trying to perform a surgery then the quality of the videoconference must be very high. In a DiffServ network for a videoconference of such importance, video packets can be transmitted using high priority class. If the videoconference is for personal reasons like talking to friends and family, then the video quality can be compromised to some extent.

To implement a good quality videoconferencing system over the Internet, four basic building blocks need to be considered. First, in order to send high quality video to the destination, there must be efficient video compression technique to minimize delay and bandwidth usage. Second, multicast technology must be implemented for the efficient distribution of video to all the recipients. Third, a routing algorithm to create efficient multicast tree to connect all the recipients involved in the videoconference minimizing end-to-end delay and packet loss. And at last, QoS support must be deployed in the Internet. The building blocks are further discussed in detail in the following sections.

2.2 MPEG

MPEG stands for Motion Pictures Expert Group. MPEG is one of the standards for encoding and decoding audio and video for compression and is used in our videoconferencing system. A video stream consists of different frames. Each video frame slightly differs from the next consecutive frame. MPEG technology compresses the video by encoding only the changes between the two consecutive frames.

The MPEG encoding consists of three frames

1. Intra frames – Intra frames of I-frames contain information about the current frame.
2. Predicted frames – Predicted frames or P-frames use information from previous I-frame or P-frames.
3. Bi-directional frames – Bi-directional frames or B-frames are similar to P-frames. The only difference is they use information from previous as well as future frames.

Each frame has a typical duration of 33 ms. In order to reduce the encoding and decoding time, only I frames and P frames are sent when MPEG streams are used in videoconferencing. I_1 P_1 P_2 P_3 I_2.... represents a typical order in which frames are sent during videoconferencing. MPEG typically requires up to 70 milliseconds to encode and decode a single frame. At the same time video packets must be able to reach the destination within 150 milliseconds, which is actually the threshold at which human senses can detect a delay between the signal and the response. Thus a video packet can stay in the network for a maximum of 80 milliseconds. If the video packet does not reach the destination within 80 milliseconds then the packet is dropped.

Though mpeg requires significant time to encode and decode, it has several advantages over other video compression techniques. Mpeg only encodes the differences between consecutive frames and it works well for a videoconference. This is because in a videoconference there is a very high possibility that the background remains same during successive frames. Since not much is changed the video stream will be highly compressed and easier to transmit over the network.

2.3 Multicast

Original idea of Internet was to provide point-to-point data transfer. There would be one source and one destination for an application. Modern applications like videoconferencing require group communication in addition to QoS support. A typical videoconference may have a source sending data to many destinations. IP Multicast is a technology by which we can transfer IP packets from a single source to many destinations. In a multicast session we have group of nodes. Each multicast group has a unique address that specifies a group of hosts, which wish to send and receive messages to and from the group.

2.3.1 Protocols used in IP multicasting

We need two different protocols for effective IP multicasting-

1. First one for group management- Used by hosts to join or leave the multicast group IGMP (Internet Group Management Protocol) [Dee89] is used for this purpose.

2. Second for constructing distribution trees- Used for routing the multicast packets to the correct destinations. There are several existing multicast routing protocols such as DVMRP (Distance Vector Multicast Routing Protocols) [WaP88] and MOSPF (Multicast Open Shortest Path First) [Moy94].

IP multicast allows replication of packets in the core. Hence it is able to create a multicast tree to connect all recipients with the fewest number of hops. But it also comes with potential problems in terms of scalability. Since it allows replication of packets in the core, all the core routers need to store multicast routing information.

2.3.2 Overlay Multicast

Since the deployment of IP multicast hasn't proved to be a scalable solution, an alternative solution, which is termed as overlay multicasting or end system multicasting, has received a lot of attention. The overlay multicasting approach assumes no multicasting support in the network layer, and constructs a multicast delivery tree in the application layer. An overlay network is a virtual network of nodes and logical links that is built on top of an existing network. The basic idea of overlay multicast is to replicate packets only at the edge routers. The overlay multicast tree consists of IP tunnels connecting edge routers in the network. Since edge routers only perform replication, core routers are kept stateless, as they do not have to maintain multicast routing information. To implement videoconference in a DiffServ domain, overlay multicast would be preferred over IP multicast. A comparison of IP multicast, multiple unicast and overlay multicast is illustrated in Figure. 1

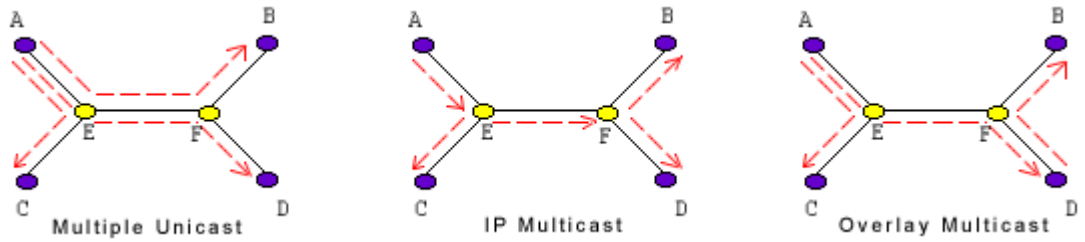


Figure 1: Comparison of different multicast techniques

2.4 Base Algorithm

DiffServ Overlay Multicast uses a modified version of the edge router multicast algorithm [BrJ03] as the tree construction algorithm. The edge router multicast algorithm is further referred as the base algorithm in this thesis. The base algorithm uses the concept of overlay multicast. In the overlay multicast edge routers instead of the end nodes control the algorithm. The edge routers are not only responsible for packet replication, but also for initiating and determining the best overlay tree when one of the end systems on their connected LAN becomes a sender. The edge routers are also responsible for transmitting all of the information that they have to the machine setting up the overlay tree. Since these edge routers are constantly running, they can constantly be monitoring the network, thereby evaluating the latency and bandwidths to other edge routers in the network. This information is used by the edge router to construct efficient multicast tree. Apart from edge routers that monitor the network and perform packet replication, there is a centrally dedicated node known as Videoconference Assistant (VA) that is responsible for setting up the videoconference and a Tree Construction Algorithm that generates the multicast tree.

2.4.1 Video Conferencing Assistant

In order to organize and manage the videoconference there must exist a central entity in the network. This entity is known as video conferencing assistance (VA). The job of the VA is to start the videoconference and also manage the join and leave of different participants during the videoconference. The participants who wish to join the videoconference must know the address of the VA. The participant who wishes to join the videoconference sends a request to the VA who in turn adds the participant to the list of participants. When the VA starts the videoconference it adds the source first and then adds one destination and the list is passed to the algorithm to construct the multicast tree. The tree construction algorithm is described in detail in section 2.4.2. The VA is also responsible for performing administrative duties. It is responsible for determining the number of participants who can act as senders and can limit the number of senders in the videoconference. The VA is also responsible for setting up security mechanisms for the videoconference. The VA monitors the information sent by the participants and determines the video quality. If the video quality deteriorates beyond a particular threshold, the VA can issue commands to the sender to reduce the bit rate, video size, or frame rate of the sending video in an attempt to increase the overall video quality.

2.4.2 Basic Tree Construction Algorithm

In a videoconference there might be one or more senders and many receivers. When a new receiver edge node (n) wants to join the multicast group, it first contacts the VA to get a list of senders. Then it contacts each source edge node (s) and sends latencies and bandwidths from n to all other existing nodes in the multicast tree. Source (s) then

decides where to place the new node (*n*) in the multicast distribution tree. The new node can be placed at many places in the distribution tree and hence we have many configurations. Figure 2 shows the different configurations that are generated after adding new node *n* to the existing tree. The new node can be added as a direct child of the source (a), grandchild of the source (b) or can replace an existing child (c) as shown in Figure 2.

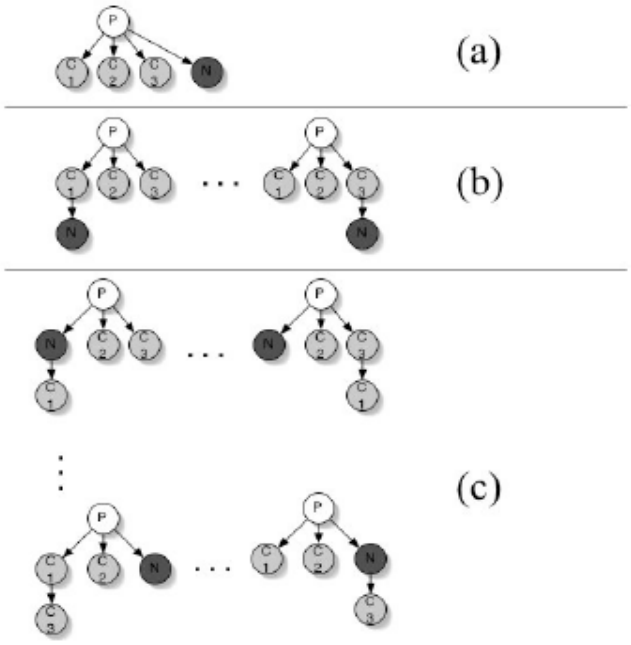


Figure 2: Possible configurations after adding node n to existing tree [Bro03]

Each time a node is added various tree configurations are generated. Each configuration is assigned a score using a score function. The configuration with the highest score is selected as the best configuration. This best configuration is then used as a starting point to add the next node. Thus this process is done recursively till all the participating nodes are covered and a final multicast tree joining all the nodes is created. Figure 3 shows the recursive pseudo code used for creating the tree configurations.

```
FindBestConfiguration(sourceID, joinID){
  /* First generate all the configurations */
  ConfigList =
     GenerateConfigurations(sourceID, joinID);

  /* Assign a score to each configuration */
  ForEach config in ConfigList{
    CalculateScore(config);
  }

  /* Find and return the Configuration
     with the highest score */
  BestConfig = FindConfigWithHighestScore();

}

GenerateConfigurations (parentID, joinID){

  /* Add a configuration where joinID is
     a direct child of parentID */
  joinID.parent=parentID;
  ConfigList=AddCurrentTreeConfiguration();

  If(parentID has no children other than joinID)
    return ConfigList;

  /* All other configurations will be created via the
     following recursive calls */

  ForEach childID of parentID {
    GenerateConfigurations(childID, joinID);
  }

  ForEach childID of parentID {
    Switch(childID, joinID);
    ForEach childID of parentID{
      GenerateConfigurations(childID, joinID);
    }
    Switch(childID, joinID);
  }

  return ConfigList;
}
/* Note that Switch(childID, joinID) makes childID's parent
point to joinID, and the handle used to keep track of the
joining node now keeps track of childID, making childID
play the role of a newly joining node */
```

Figure 3: Recursive pseudo code used for creating the tree configurations [Bro03]

To evaluate the generated tree configurations, each configuration is assigned a score. The score is calculated taking into account available bandwidths and latencies on the path to all the destinations. The configuration with the highest score is selected as the multicast tree. The score is calculated using the score function as shown in Figure 4.

$$Score_j = \alpha * Bw_j + (1-\alpha) * \frac{1}{Lat_j}; 0 \le \alpha \le 1$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} AvgBw_k}{NumConfigs}}; Lat_j = \frac{MaxLat_j}{\frac{\sum_{k=1}^{NumConfigs} MaxLat_k}{NumConfigs}}$$

Figure 4: Score Function for base algorithm [Bro03]

Here *BWj* is the scaled bandwidth number for the configuration *j*. *BWj* is calculated by taking the average available bandwidth from the source to each of the nodes being used by the configuration *j* and dividing it by bandwidth scalar. *AvgBWj* is the average of available bandwidth from source to all destinations in configuration *j*. Similarly, LAT_j is calculated by determining the maximum latency (from the source to all the destinations) of the current configuration *j* dividing by the average maximum latency of all the possible configurations. The configuration with the highest score is selected as the multicast tree.

The base algorithm is further modified and enhanced to support videoconferencing in DiffServ networks. The enhancements are discussed in detail in further chapters.

2.5 QoS

The last piece for implementing good quality video conferencing system is providing QoS support. QoS [FeH98][Wan01] refers to network performance measures such as service availability, delay, delay-variation (jitter), throughput, and packet loss rate as seen by users and applications. While many data oriented applications can live with the current best-effort Internet (with no QoS guarantees other than reliable delivery), audio and video generally require QoS guarantees. There must be upper bounds on the delay to ensure real-time delivery of the packets and packet loss and corruption must be bounded to ensure good quality. Some classes of applications require further assurances from the network, such as bounds on jitter (variability in delay between two back to back packets). QoS can be seen as a means for reserving and allocating existing bandwidth. Thus, to the user, a computer network with QoS functionality is equivalent to the telephone system. When the user wants to make a call he initiates the connection by dialing the number. Then, assuming that the call goes through, the user is assured a clear channel on to communicate. Alternatively, at call setup time, the user may instead get a busy signal and be denied the privilege of connecting at the desired level of QoS. With IP QoS, service providers can achieve greater profitability through more business and more efficient use of bandwidth, enhanced service differentiation and better-than-best-effort service. This service model is different from the "best-effort" service model of the current Internet. In today's Internet, each network element (router) along an IP packet's path makes nothing more than a good effort to forward the packet towards its destination. It works on the principle of First Come First Serve (FCFS). If a queue is overloaded, packets are dropped with little or no distinction between unimportant traffic and urgent

traffic like video. The primary Internet transport protocols (like TCP) have been very carefully designed to support a service model, where connections are essentially never denied and every connection's performance simply degrades as the network load increases. Basically there are two key QoS issues in the Internet: 1) Resource Reservation and 2) Performance Optimization.

2.5.1 Resource Reservation

The current Internet is based on TCP protocol. Applications sense congestion in the network and reduce their transmission rate and thereby reducing packet loss. This works well if all the applications are TCP based and they comply with the rules. But malicious users may try to gain more bandwidth by modifying the TCP stack leading to congestion in the network. Plus not to mention there are UDP based applications that do not back off even in congestion. Thus packets in the network are delayed or dropped whenever there is congestion and thereby reducing the performance of real-time applications. Hence resource allocation (allocating bandwidth and buffer space) is required for important real time applications to be supported in the Internet. The Internet community has done a lot of active research in this field in the past decade. IETF has introduced two main architectures for resource allocation, the Integrated Services (IntServ) [BrC94] that uses Resource Reservation Protocol (RSVP) [BrZ97] and the Differentiated Services (DiffServ) [Ber98] [BlB98]. Between the two main approaches, the DiffServ scheme provides a less complex and scalable solution. While the IntServ maintains per-flow states in each node, the DiffServ only discriminates among packets in different classes and not individual flows.

2.5.2 Performance Optimization

Once the architecture for allocating resources in the network is fixed, the next issue is performance optimization. For allocating resources for a particular application we need to find paths in the network that will satisfy the requirements such as available bandwidth. The current Internet uses routing protocols that select shortest path to the destination. Shortest path might not always be the best path as it might be congested and not always have the resources we need for the application. In such cases if the resources cannot be allocated then the connection will be rejected. On the other hand there might be alternate paths to the destination that might have the resources but are neglected by the shortest path algorithm. Hence we need routing algorithms that calculate the paths taking into account the available link bandwidth and latencies.

2.5.3 Integrated Services with RSVP signaling

The first architecture to provide QoS in the Internet was the Integrated Services (IntServ) architecture [BrC94]. IntServ assumes that resources are reserved for every flow requiring QoS at every router hop in the path between source and destination, using end-to-end signaling and must maintain a per flow soft state at every router in the network. A fundamental component of this architecture is the setup or signaling protocol, which conveys the application's resource requirements into the network. The most important and well known of the IP signaling protocols is the Resource Reservation Protocol (RSVP) [BrZ97]. The objective of the IntServ architecture is to facilitate the transmission of both Best-effort and Real-time traffic over the IP networks.

Drawbacks of the IntServ model

1. Every device along a packet's path, including the end systems like servers and PCs, need to be fully aware of RSVP and capable of signaling the required QoS. Every router must perform the tasks of packet classification, scheduling and admission control. This makes the routers more complex.
2. Reservations in each device along the path are "soft," which means they need to be refreshed periodically, thereby adding to the traffic on the network and increasing the chance that the reservation may time out if refresh packets are lost. Though some mechanisms alleviate this problem, it adds to the complexity of the RSVP solution.
3. Maintaining soft-states in each router, combined with admission control at each hop adds to the complexity of each network node along the path, along with increased memory requirements, to support large number of reservations.
4. Since state information for each reservation needs to be maintained at every router along the path, scalability with hundreds of thousands of flows through a network core becomes an issue.

Since per-flow QoS is difficult to achieve end-to-end in a network without adding significant complexity, cost, and introducing scalability issues, it naturally leads one to think about classifying flows into aggregates (classes), and providing appropriate QoS for the aggregates. For example, all TCP flows could be grouped as a single class, and bandwidth allocated for the class, rather than for the individual flows. In addition to classifying traffic, signaling and state maintenance requirements on core network nodes

should be minimized. This idea led to the existence of Differentiated Services (DiffServ) Architecture.

2.5.4 Differentiated Services

The scalability problems of IntServ and the need to find solutions to IP QoS quicker than a strict end-to-end solution have driven the Internet research community to look for simpler solutions. During 1997, several proposals with similar characteristics emerged as Internet Drafts, and in the December 1998 Differentiated Services (DiffServ) Architecture [Ber98] [BlB98] was proposed.

DiffServ Basics

The Best Effort model and the IntServ represent two extremes in terms of resource allocation. The Best Effort model works on per-packet basis whereas the IntServ deals with the individual flows. The DiffServ approach lies in between these two extremes and offers a simple solution from the viewpoint of implementation. The DiffServ model divides the traffic into a small number of classes called forwarding classes. The type of forwarding class to which a packet belongs to is carried in the IP packet header. For a particular class the forwarding treatment in terms of drop priority and bandwidth allocation is fixed and defined. In DiffServ network, the nodes at the boundaries (Edge Routers) and those inside the network (Core Routers) perform different operations. When the traffic arrives at the boundary, the edge router performs two basic tasks: packet classification and traffic conditioning. They include mapping the packets to different forwarding classes, checking whether the traffic flows meet the service requirements and dropping non-conformant packets. Within the interior of the network,

the packets are forwarded based solely on forwarding classes in the IP packet header by the core routers.

Resource allocation to aggregated traffic flows rather than individual flows: In DiffServ, the resources are allocated to the individual traffic classes that represent the aggregated traffic. DiffServ creates different levels of service and resource assurance but not absolute bandwidth guarantees or delay bounds for the individual flows.

Per-Hop Behaviors (PHBs)

In DiffServ, the Forwarding Treatments at a single node are described by the term PHB. In DiffServ context, Forwarding Treatment refers to the mechanism or the algorithm, which is implemented at a router or a node to forward the packets. Each PHB is represented by a 6-bit value called a Differentiated Services Code Point (DSCP). All the packets with the same DSCP are referred to as Behavior Aggregate (BA) and they receive the same forwarding treatment. The PHBs are used to allocate resources to different behavior aggregates. Different behavior aggregates compete for the resources at the node. A set of PHBs may form a PHB group. A PHB group must be defined meaningfully as a group that shares a common constraint. A PHB group may describe resource allocation in relative terms among themselves in terms of bandwidth allocation and drop priority. For example, a PHB group may be defined as a set of drop priorities for each PHB in the group. PHBs are implemented by the means of buffer management and packet scheduling.

Services: A Service is defined as the overall performance that a customer's traffic receives within a DS domain or end to end. Creating a service requires that many components work together: mapping of traffic to specific PHBs, traffic conditioning at the boundary nodes, network provisioning and PHB based forwarding in the interior of the network. In DiffServ architecture the services are defined in terms of Service Level Agreement (SLA) between the customer and the service provider. A SLA provides the details of the services to be provided to the customer.

Differentiated Services Code Point

The current IP packet header includes an 8- bit field called IP TOS field as shown in Figure 5. It is composed of a 3-bit precedence bits, 3-bit type of service (TOS), and 2 unused bits that must be zero. The precedence bits represent the priorities for the traffic, whereas the TOS bits indicate the preference for throughput, delay and loss.

Precedence	D	T	R	0	0

Figure 5: IP TOS Field

The DiffServ standard redefines the existing IP TOS field to indicate forwarding behaviors. The replacement field called the DS field supersedes the existing definitions of the TOS octet and also the Ipv6 traffic class octet. The first 6 bits of the DS field are used as a Differentiated Services Code Point (DSCP) to encode the PHB for a packet at each DS node. This is shown in the Figure 6 below. The remaining 2 bits are currently unused (CU).

Figure 6: DS Field

Types of PHBs available to construct a DiffServ-enabled network

The Default PHB (DE):

The default PHB essentially specifies that a packet marked with a DSCP value (recommended) of `000000' gets the traditional best effort service from a DS-compliant node (a network node that complies to all the core DiffServ requirements). Also, if a packet arrives at a DS-compliant node and its DSCP value is not mapped to any of the other PHBs, it will get mapped to the default PHB.

Expedited Forwarding (EF) PHB [JaN99]:

Just as RSVP, via the IntServ model, provides for a guaranteed bandwidth service, the EF PHB is the key ingredient in DiffServ for providing a low-loss, low-latency, low-jitter, and assured bandwidth service. Applications such as voice over IP (VoIP) and video conferencing require such a robust network-treatment. EF can be implemented using priority queuing, along with rate limiting on the class. Although EF PHB when implemented in a DiffServ network provides a premium service, it should be specifically targeted toward the most critical applications, since if congestion exists, it is not possible to treat all or most traffic as high priority. EF PHB is especially suitable for applications (like Video Conferencing) that require very low packet loss, guaranteed bandwidth, low delay, and low jitter. The delay and delay variation occur due to the

congestion or basically traffic queues in the network. This increase in the traffic queue occurs when the departure rate is slower than the arrival rate in the same node. Hence a type of service that ensures no queues for some aggregate should be such that, at every core node, the aggregate's maximum arrival rate is less than it's minimum departure rate. Now EF sets up the nodes in such a way that the aggregate has a minimum departure rate, which is independent of the intensity of the other traffic at the node. Also the traffic conditioners at the network boundary execute the policing and shaping so that the arrival rate at any node is always less than the configured minimum departure rate.

Implementation: EF PHB can be implemented in several ways. A simple priority queue with a token bucket will give the desired result, provided that there is no higher priority queue that would make the packets be preempted for more than a packet time at the configured rate. Token bucket is to limit the amount of EF traffic so that the other traffic will not be starved by bursts of EF traffic. Another way to implement EF is to use Weighted Fair Queuing (WFQ) and configure the weights in such a way that EF traffic has highest priority. An important thing for the EF PHB is the DSCP for this behavior. The DSCP is 101110, which is recommended for EF PHB. If packets are marked for EF PHB then their DSCP can be marked again at a DS domain boundary only if the new DSCP satisfies the EF PHB. Also the packets that are marked with for EF PHBs should not be changed to any other type of PHB by a DS domain.

Assured Forwarding (AF*xy*) PHB [HeB99]:

The rough equivalent of the IntServ Controlled Load Service is the Assured Forwarding PHB. It defines a method by which Behavior Aggregates can be given

different forwarding assurances. For example, traffic can be divided into different AF Classes.

The AF*xy* PHB defines four AF*x* classes; namely, AF_1, AF_2, AF_3 and AF_4. Each class is assigned a certain amount of buffer space and interface bandwidth, dependent on the Service Level Agreement with the Service Provider policy. Within each AF*x* class, it is possible to specify three-drop precedence values. Thus, if there is congestion in a DS-node on a specific link, and packets of a particular AF*x* class (say AF_1) need to be dropped, packets in AF*xy* will be dropped such that the $dp(AF_1) \leq dp(AF_2) \leq dp(AF_3)$, where $dp(AFxy)$ is the probability that packets of the AF*xy* class will be dropped. Thus, the subscript "*y*" in AF*xy* denotes the drop precedence within an AF*x* class. Thus, packets in AF_{13} will get dropped before packets in AF_{12}, before packets in AF_{11}.

Table 1 shows the DSCP values for each class, and drop precedence. And AF*x* class can be denoted by the DSCP where '*xyz*' is 001 / 010 / 011 / 100, and '*ab*' represents the drop precedence bits where '*ab*' is 01/10/11.

	Class 1	**Class 2**	**Class 3**	**Class 4**
Low Drop Precedence	**001010**	**010010**	**011010**	**100010**
Medium Drop Precedence	**001100**	**010100**	**011100**	**100100**
High Drop Precedence	**001110**	**010110**	**011110**	**100110**

Table 1- AF drop precedence

Implementation: An AF implementation must detect and respond to long-term congestion in terms of minimizing it for each class as well as detecting and responding to it also. RED i.e. Random Early Detection [BrC98] is one such method. This method keeps a check on the buffer in which a certain threshold value is given. If the queue size increases beyond the threshold value then the RED algorithm discards the incoming packets with a probability *p*.

DiffServ Architecture

Figure 7 gives a pictorial overview of DiffServ end-to-end architecture as mentioned in [Cis01].

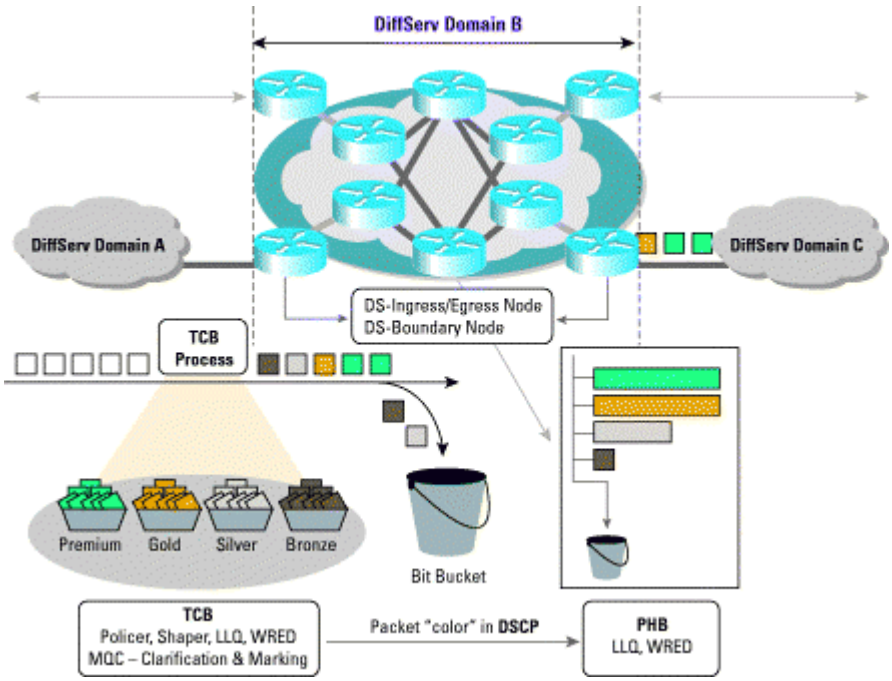


Figure 7: DiffServ End-to-End Architecture [Cis01]

For true QoS, the entire IP path that a packet travels must be DiffServ enabled. An example service policy might be that EF gets 10 percent, Gold 40 percent, Silver 30 percent, Bronze 10 percent, and Best Effort traffic (default class/PHB) the remaining 10 percent of the bandwidth. Gold, Silver, and Bronze could be mapped to AF classes AF_1, AF_2 and AF_3 for example.

DiffServ Domain

A DiffServ-Domain itself is made up of DiffServ Ingress nodes, Core nodes, and Egress nodes as shown in Figure 8. Further, an Ingress or Egress node might be a DiffServ Boundary Node, connecting two DiffServ Domains together.

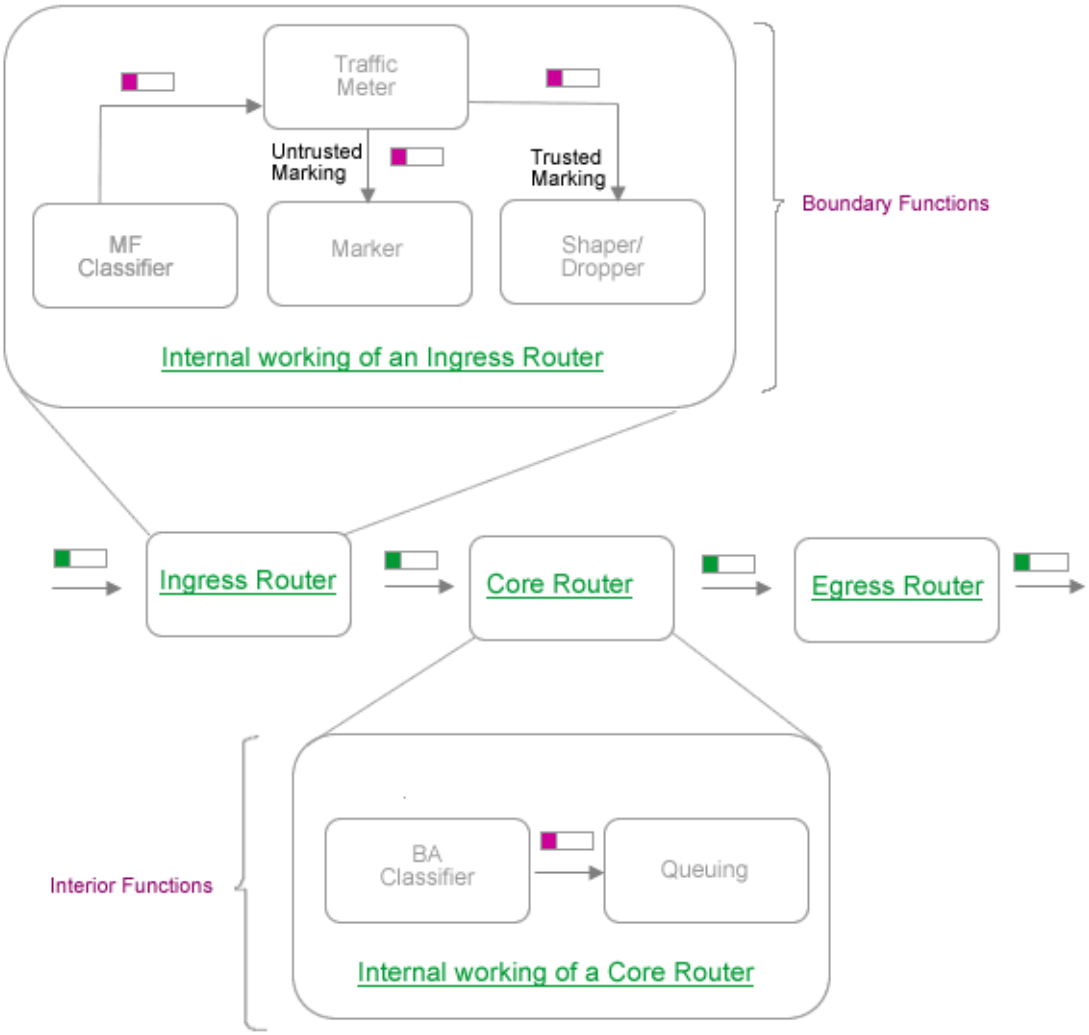


Figure 8: DiffServ Boundary and Interior nodes [Wan01]

Boundary Nodes

In a DiffServ domain, the boundary nodes (Ingress Routers) are responsible for mapping packets to one of the forwarding classes supported in the network and they must ensure that the traffic conforms to the SLA for the specific customer. Once the packets pass the boundary nodes into the interior of the network then the resource allocation is done solely on the basis of forwarding classes. These functions, which the boundary nodes perform, are known as traffic classification and conditioning. The traffic classification module contains the classifier and a marker as shown in Figure 9.

Traffic Classification and conditioning:

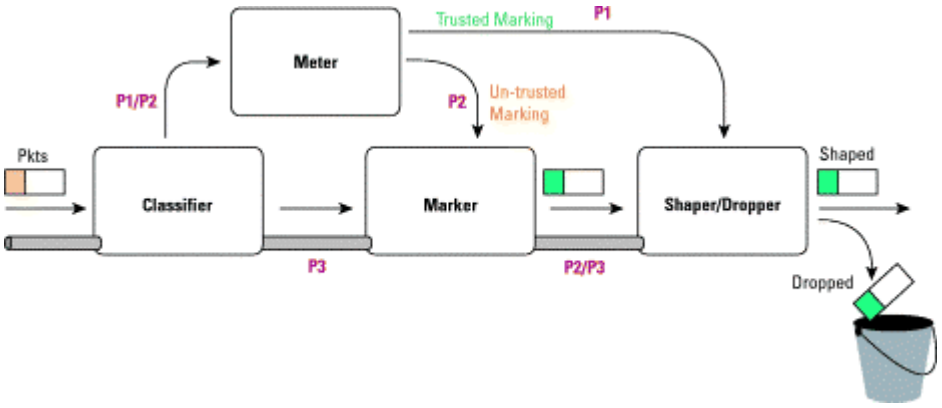


Figure 9: DiffServ Traffic Conditioner Block (TCB) [Cis01]

Classifier: A classifier divides the incoming packet stream into multiple groups based on predefined rules. There are two basic types of classifiers: Behavior Aggregate (BA) or the Multi-field (MF). A BA classifier selects the packets based solely on the DSCP values in the packet header. BA classifiers are used when the DSCP has been set before the packet reaches the classifiers (generally used in Core routers). The MF classifier uses a

combination of one or more fields of the five tuple (source address, destination address, source port, destination port and the protocol ID) in the packet header for the classification. It can be used for more complicated resource allocation policies from the customers (generally used in Boundary routers).

Marker: Markers set the DS field of a packet to a particular DSCP.

Traffic Conditioner: Traffic conditioners perform traffic policing functions to enforce the Traffic Conditioning Agreement (TCA) between the customers and the service providers. A traffic conditioner consists of four basic elements.

Meter: For each forwarding class a meter measures the traffic flow from a customer against its own traffic profile. The in-profile packets are allowed to enter the network, while out of profile packets are further conditioned based on the TCAs. Since traffic profiles are typically described in terms of token bucket parameters, most meters are implemented as token buckets. The Meter then passes results to Re-marker and shaper/dropper to trigger action for in/out-of-profile packets.

Re-Marker: When a traffic stream passes a meter and the meter decides that the traffic is non-conformant, these packets may be re-marked with a special DSCP to indicate their nonconformance. These packets must be dropped if the network faces congestion. Thus when a packet stream violates traffic profiles, they may be re-marked to a different DSCP. When a packet remarked with DSCP receives worse forwarding treatment than from the previous DSCP, this is referred to as PHB demotion. If the new DSCP represents a better forwarding treatment than previous one, it is referred as PHB

promotion. The boundary nodes demote out of profile packets to a DSCP with worse forwarding treatment.

Shaper: Shapers delay the non-conformant packets in order to bring them to be compliant with the traffic profile. The shaper prevents the packet from entering the network until the stream conforms to the traffic conditions. Thus shaper is a stronger form of policing than marking.

Dropper: Dropping is another action applied to the out of profile packets. Compared to shaping which needs a buffer to store the packets dropping is much easier to implement. Since the shaper has a finite-size buffer, it also drops the packets when the buffer overflows.

Core nodes: The core Routers provide PHB based on the DSCP bits contained in the DS-field of the incoming packets. These routers examine the code points in the packet header and determine how the packet should be treated. A BA classifier is used to select the packets based solely on the DSCP values in the packet header. These routers typically employ a queue management and scheduling scheme to provide proper forwarding treatment to the packets.

Bandwidth Broker

The bandwidth broker (BB) is a centrally dedicated node in the DiffServ network. It has the knowledge of the entire diffserv topology. The bandwidth broker has knowledge of how much EF traffic is going over each link in the diffserv network. In

order to reserve resources, the bandwidth broker can be contacted to get information about available bandwidth for EF class.

Call Admission Control (CAC)

CAC is implemented at the edge of the DiffServ network. When a request for setting up an EF class connection arrives at the edge of the DiffServ domain, the edge router implements CAC. Routing Algorithm checks to see if it can find a valid path with available bandwidth to support the connection. If a valid path is found the connection is accepted else it is rejected and the source initiating the connection is notified.

DiffServ – A Scalable Solution

DiffServ is preferred over IntServ and is suggested to be a scalable scheme for providing QoS in an IP network. Some of the advantages of DiffServ over IntServ are:

1. The amount of state information that core routers need to maintain is proportional to the number of traffic classes, not the number of application flows. Hence the solution is scalable.
2. DiffServ Edge routers perform packet classification, policing, marking and shaping. Hence all the complexity is pushed to the edge of the network.
3. Core routers only need to classify packets based on their DSCPs (BA classification) and perform queuing. Thus high-speed core routers are kept simple.

At the edge of the network, link speed is usually low. Therefore the edge routers can spend more time on complex classification, policing, marking and shaping. At the core of the network, link speed is high. The core routers must be kept simple in order to be able to forward packets quickly. Due to all these reasons DiffServ is preferred over IntServ as the resource allocation architecture for providing QoS.

DiffServ IP Multicast

We need to implement IP multicast in DiffServ networks for QoS support. The basic concepts of DiffServ can be applied to multicast service where traffic can be forwarded from ingress to many egress routers. However since DiffServ requires core routers to be simple, this makes it difficult to use existing IP multicast directly in a DiffServ domain. Hence certain modifications are required because of the problems of integrating diffserv and IP Multicast technologies [BeW04].

Problems

1) Complexity in the Core: In traditional IP multicast, each router maintains multicast routing tables. When a multicast packet arrives at a router, the router checks entries inside the routing table and replicates the packet onto the links based on the entries. This means each router in the network has to store multicast routing information. But this contradicts the DiffServ principles where DiffServ core routers do not maintain any state and are simple.

2) Resource Reservation Problems: In order to provide the QoS we need to reserve the resources. For IP multicast there are many receivers. Hence the resources have to be reserved between the sender and all the receivers. But providing resources is difficult as the receivers can join and leave the group dynamically leading to dynamic resource consumption. If this fact is not considered then it can lead to Neglected Reservation Sub Tree (NRS) problem as explained in [BeW04]. Resources must be reserved before they can be used. But in IP multicast, group membership is very dynamic and hence we cannot use the sender initiated resource reservation. When a new member is added dynamically, it can affect the other existing traffic if the resources are not reserved for the new member. When a new receiver is added to the multicast group, the corresponding multicast routing protocols construct the multicast tree taking into account the new member. Thus if a DiffServ core node wants to replicate the packets in order to send them to the new receiver then it will simply copy the DSCP value into the replicated packet and send it. The DiffServ core node has no idea if any resources are reserved for the new member. If not reserved then it will affect the QoS levels of the other receivers on that link. This negative effect on the other receivers is known as NRS problem.

NRS problem can occur in two cases:

Case 1:

Here the branching point of the new sub tree and the previous multicast tree is the egress router as shown in Figure 10. The additional multicast flow increases the used resources for the EF group aggregate and will be greater than the reserved amount. The policing component of the egress router discards the excess EF packets till the traffic

conforms to the contract. But the router doesn't have per flow state and hence it starts dropping packets belonging to the EF group at random. This may drop packets of the other flows belonging to the EF group and hence they may not receive guaranteed service.

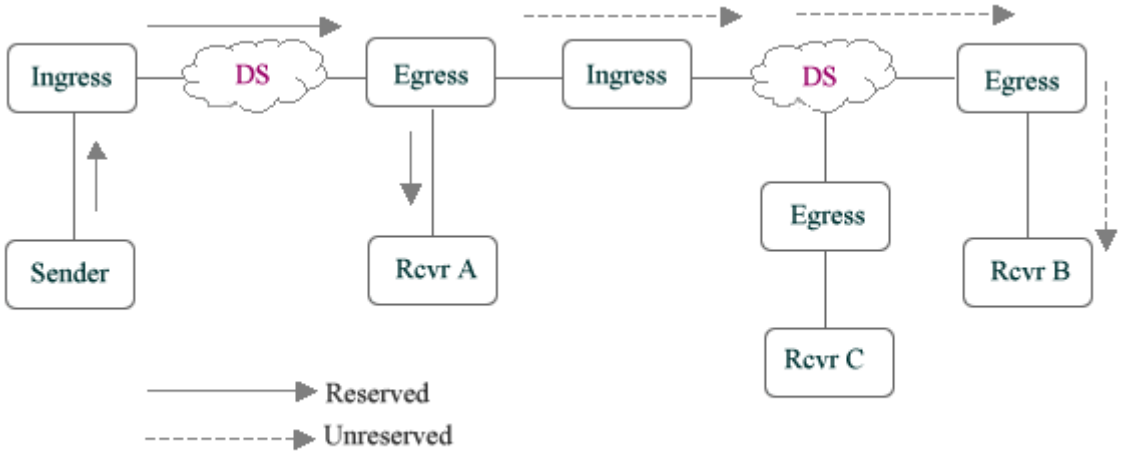


Figure 10: Neglected Reservation Subtree problem case 1 [BeW04]

Assume after addition of the new multicast flow we have 70 % EF on the link. This will be throttled to 40 %. Hence the excess 30 % will be dropped at random. However the AF and the BE flows will not be affected.

Case 2:

In this case the branching point between the previous multicast tree and the new sub tree is a core router as shown in Figure 11. Core routers do not implement metering and policing. They simply check the DSCP bits and provide appropriate forwarding treatment for the packet. Hence the core routers won't recognize excess amount of traffic.

Hence for the new multicast flow, which is EF type, it will simply provide the resources. In doing so it will steal resources from lower priority aggregates like AF and BE.

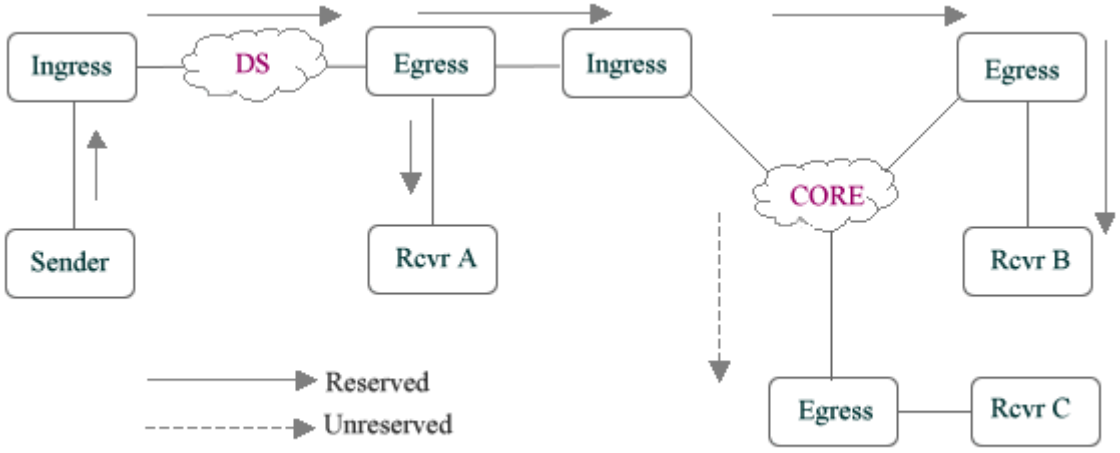


Figure 11: Neglected Reservation Subtree problem case 2 [BeW04]

Thus in this case if we have excess 30% EF, then it will steal 30% from AF, which will steal 20% from BE. Hence there will be no packet loss for EF but there will be 10% packet loss for AF and complete packet loss for BE.

3) Dynamics of arbitrary sender change: DiffServ is unidirectional by definition hence we consider DiffServ multicast as unidirectional point to multipoint communication. But for video conferencing we need multipoint-to-multipoint communication service. Hence every sender must have its own multicast tree and resources have to be reserved along that tree if simultaneous sending is required with better service. If this is not implemented it results in NRS problem.

4) Heterogeneous Multicast Groups: Heterogeneous multicast groups contain one or more receivers requesting different classes of service. A receiver requesting different level of QoS can be grafted at any point of the existing multicast tree. Therefore even interior nodes may have to replicate packets using the different service. Consequently, in order to accomplish this, interior nodes have to change the code point value during packet replication. But this contradicts DiffServ architecture of maintaining simple core routers. An alternate solution would be to use separate multicast trees for receivers requesting different QoS levels. But this solution puts more burden on the network bandwidth.

2.6 DiffServ Multicast Technologies

A relatively small amount of research is going on in the field of DiffServ Multicast. Some of the existing schemes are summarized below. The authors of [BeW04] have outlined the problems of multicasting in DiffServ domain. The existing schemes are the Native Multicast using DVMRP [WaP88], DSMCast [StM01] [Str01], EBM [StB03] and QDM [LiM02].

2.6.1 Native Multicast using DVMRP

Native multicast is used in the current Internet for providing multicast support. Native multicast uses Distance Vector Multicast Routing Protocol (DVMRP) [WaP88] in order to generate the multicast tree. When new receiver wants to join the multicast, the algorithm comes up with the shortest path in terms of hop distance to connect the new receiver to the already existing multicast tree.

Merits:

Since native multicast generates multicast tree using shortest path approach, the Cost of reserving bandwidth in a DiffServ network for the tree would be minimum. Also the end-to-end packet delay from source to destination would be less than that required for an overlay multicast tree to connect the same nodes.

Weakness:

The major weakness of implementing native multicast in a DiffServ network is the scalability. Since native multicast allows packet replication in the core, the core routers need to store multicast routing information. This is against the principles of a DiffServ network, which state that the core routers need to be stateless. Thus native multicast, though it can prove to minimize the cost and end-end delay, is not scalable option for creating multicast trees in a DiffServ network.

2.6.2 DSMCast

Idea: In DSMCast [StM01] each edge router has the complete knowledge of the entire topology of the DiffServ domain. DSMCast header is added at the edge of the DiffServ domain at the ingress router. As the ingress router knows the entire topology it can appropriately construct the multicast tree based on the topology and egress nodes that require data for multicast group. All requests by egress routers to leave/join the group must be made to ingress router, as it is responsible for creating the multicast tree. When the core router receives the multicast packet, it checks the header to determine which interface the packet should be replicated upon. Once replicated the packet will be treated

as an ordinary unicast packet according to its DSCP. Core routers only have the computation of locating branching information in the DSMCast header and replicate the packet on appropriate branches.

In traditional IP multicast the routing table lookup cost increases with additional per group information whereas the lookup cost of DSMCast header depends only on the number of nodes in multicast tree in given domain.

Merits:

1) Stateless Core: Complexity of the core is shifted to the edge of the network. Edge router performs the task of calculating the multicast tree. Multicast routing information is inserted in the packet header. Core routers do not have to maintain multicast routing states. Hence core routers are stateless complying with the DiffServ architecture.

2) Resource management: Edge router (ingress) is responsible for constructing the multicast tree. The edge router is able to easily monitor the resources being consumed by the multicast tree. Hence the edge router can check and reserve resources before joining a new receiver to the existing multicast tree. Hence this solves the neglected reservation subtree (NRS) problem.

3) Heterogeneous QoS support: DSMCast supports heterogeneous QoS extension in the packet header. Thus a single multicast tree can be used rather than requiring separate trees for each level of QoS. The encapsulation can allow the PHB (Per-Hop Behavior) to change as the multicast packet crosses the domain to support heterogeneous QoS in a single multicast tree.

Thus the primary strength of this architecture is that it addresses all the problems of IP Multicasting in DiffServ domain. However, DSMCast does have some key weaknesses.

Weaknesses:

1) Overhead: The main idea of DSMCast is to embed the multicast routing information in the packet header. Hence every multicast packet introduces the extra overhead in terms of multicast routing information. To transmit every multicast packet extra network bandwidth will be consumed. Thus DSMCast achieves stateless core in the network at the price of network overhead.

2) Hardware Processing: Since the multicast routing information is in the packet header, to decode the information each DiffServ core router will have to be upgraded with extra hardware for processing the header information. Though with the advancement in technology hardware processing can be made fast, a small amount of delay will be introduced at each router from the multicast source to the destination.

3) The author does not explicitly mention the tree construction algorithm used in DSMCast [StM01]. The author mentions that the tree construction algorithm can be optimized for QoS by using shortest path tree as specified in [AdN02]. But in a network shortest path might not always be the best path. Shortest path between a source and a destination may be overloaded. On the other hand an alternate path might take more number of hops to reach the destination but might have available bandwidth and might satisfy end-to-end latency for the application. Hence we need a tree construction

algorithm that takes into account the link latencies and available bandwidths while calculating the multicast tree.

2.6.3 EBM

Idea: EBM [StB03] reduces DiffServ Multicast problem by employing an edge based approach that relies on standard IP routing and tunneling. EBM architecture incorporates the use of Multicast Broker (MB) for group management and Edge Cluster Tree (ECT) Algorithm for tree construction.

Principles of EBM:

1. Stateless Core: Core routers are multicast unaware and do not maintain any multicast routing state information.
2. Tunneling: Packets are tunneled from edge to edge routers.
3. Edge Replication: Packets may be replicated only at edge routers.
4. Multicast Broker: MB manages egress join/leave requests and multicast trees for the given DS domain. MB also manages QoS interactions for multicasting (Resource Reservations).

Edge routers must provide two functions

1. Packet replication/ tunneling
2. Join/leave forwarding

For packet replication/tunneling edge router must be able to recognize the packet and replicate/tunnel the packet into the domain. Edge routers store multicast state information to replicate/tunnel packets. For join/leave forwarding MB plays an important

role. Any join/leave request to an incoming edge router must be tunneled to the MB along with the information about QoS requested (PHB) and the address of the egress router requesting multicast service. The MB then processes the join/leave request and updates appropriate edge routers with the new tree information. Thus the edge router need not be concerned with the specifics of the multicast routing, only appropriate mechanisms for packet tunneling, replication and forwarding to MB.

Edge Cluster Algorithm (ECT):

Idea: To cluster similar QoS class egress points together in an effort to balance cost of the tree versus additional hops required for edge based branching. ECT Algorithm can be summarized in two phases:

Cluster Construction:

Rules:

1. A cluster is constructed centered on each egress point of the multicast group.
2. A cluster consists of all the other egress points within H hops of an edge router (E_X), whose PHBs can be satisfied by the PHB used for packets sent to E_X.

Cluster Linkage:

Rules:

1. Clusters are linked via tunnels to connect the multicast distribution tree.
2. Algorithm proceeds by selecting best cluster according to the metric. The metric $M(E_X)$ for evaluating a cluster centered at node E_X is defined as:

$$M(E_X) = \frac{D(I_G, E_X) + \sum_{i=0}^{|C_X|} D(E_X, C_{X,i})}{|C_X|}$$

where $D(x, y)$ is the number of hops between x and y, I_G is the ingress router for the group (single source), $|C_X|$ is the number of nodes within H hops that are satisfied by E_X's PHB, and $C_{X,i}$ is the i^{th} node in the cluster centered around E_X. A lower value of $M(E_X)$ denotes that the average cost of servicing the nodes in a cluster is lower as well.

3. Once a cluster is selected, all of the nodes inside the cluster are removed from consideration as members of other clusters.
4. Additional constraint may be imposed, cluster depth D, such that a cluster may not be more than D tunnels away from the ingress point.

Merits

1) Stateless Core: EBM uses the edge-based approach in which the replication is carried out at edge routers only. Complexity of the core is thus shifted to the edge of the network. Edge routers maintain multicast routing information and perform replication/tunneling. Core routers do not have to maintain multicast routing states. Hence core routers are stateless complying with the DiffServ architecture.

2) Resource management: EBM uses the concept of multicast broker (MB) for constructing the multicast tree. MB manages egress join/leave requests and multicast trees for the given DS domain. The MB is able to easily monitor the resources being consumed by the multicast tree. Hence the MB can check and reserve resources before

joining a new receiver to the existing multicast tree. Hence this solves the neglected reservation subtree (NRS) problem.

3) Heterogeneous QoS support: EBM uses ECT as the tree construction algorithm. Egress points are clustered together in an effort to balance cost of the tree versus additional hops required for edge based branching. Thus a single multicast tree can be used rather than requiring separate trees for each level of QoS. ECT can allow the PHB (Per-Hop Behavior) to change as the multicast packet crosses the domain to support heterogeneous QoS in a single multicast tree.

Thus the primary strength of this approach is that it addresses all the problems of IP Multicasting in DiffServ domain. However, EBM does have some key weaknesses.

Weaknesses

Two variations of tree construction algorithm are defined in EBM [StB03]. In the first scheme a new receiver is grafted to the nearest edge node in the existing multicast tree. The next scheme uses ECT algorithm to form the multicast tree. But both the algorithms make no attempt to check link latencies and available bandwidths while constructing the multicast trees. Hence the multicast tree formed might not be optimized for available bandwidth or end-to-end latency.

2.6.4 QoS-Aware Multicast in DiffServ (QMD)

Idea: QMD [LiM02] constructs a scalable QoS satisfied multicast tree within one DiffServ domain that can connect all the edge routers (sender and receivers). Incorporates three types of nodes:

1. Edge routers/nodes - Process the control messages (join/leave) and maintain control state information.
2. Core routers – Stateless DiffServ core routers, which perform forwarding based on DSCP.
3. Key routes/nodes – They are upgraded core routers, which have the functionality of storing multicast routing information to replicate and forward packets. They store forwarding state information.

Merits

1) Stateless Core: In QDM the routing and the control messages are migrated from the core to the edge routers. Edge routers process the control messages and perform routing decisions and resource reservations. Hence there is less burden on core routers complying with the DiffServ architecture.

2) Resource management: In QDM, the edge router (ingress) is responsible for constructing the multicast tree. The edge router is able to easily monitor the resources being consumed by the multicast tree. Hence it can check and reserve resources before joining a new receiver to the existing multicast tree. Hence this solves the neglected reservation subtree (NRS) problem.

Weaknesses

1) QDM uses the concept of key nodes. Key nodes are upgraded core routers, which have the functionality of storing multicast routing information to replicate and forward packets. The authors suggest the use of a revised version of DIJKSTRA algorithm (QMD-DIJKSTRA) to minimize the number of key nodes while connecting a new member to the existing multicast tree. Although the number of key nodes can be

minimized using this approach, it does not completely eliminate the multicast state information from the core. Secondly the position of key nodes for one multicast tree might be different for other multicast tree. Hence to implement this scheme all the core routers will need to support the functionality of storing multicast routing information to replicate and forward packets.

2) This work does not address the issue of providing heterogeneous QoS

2.7 Proposed Solution

Although the existing solutions [StM01], [Str01] and [StB03] do provide a framework for solving the problems associated with multicasting in DiffServ domains, none of them address the issue of traffic engineering and load balancing in the network. This thesis proposes, "DiffServ Overlay Multicast", architecture for providing multicasting in a DiffServ domain. The architecture addresses all the problems associated with DiffServ Multicast and also provides a multicast routing algorithm that supports traffic engineering. The architecture uses the base overlay multicast algorithm as described in section 2.4. Before using the base algorithm certain enhancements are done to the algorithm to improve performance of the video during a videoconference. Chapter three provides insight to the enhancements. Chapter four describes DiffServ Overlay Multicast in detail. Chapter 5 and 6 provide details of the simulation setup and the results.

Chapter 3 – Enhancement of the Base Algorithm

The base overlay algorithm [BrJ03] as described in section 2.4 can be applied for video conferencing in a diffserv domain. For the algorithm to be applied in a diffserv domain certain modifications are required to be done. Certain modifications are also done to make the algorithm more enhanced in order to create efficient multicast trees and improve the overall video quality. These modifications can be listed in the following three stages:

1. Modifying the calculation for Bandwidth Scalar.
2. Randomizing selection of configurations having same highest score.
3. Information to calculate available bandwidth in tunnels.

3.1 Modifying the Calculation for Bandwidth Scalar

The score function as described in [BrJ03] is given as follows:

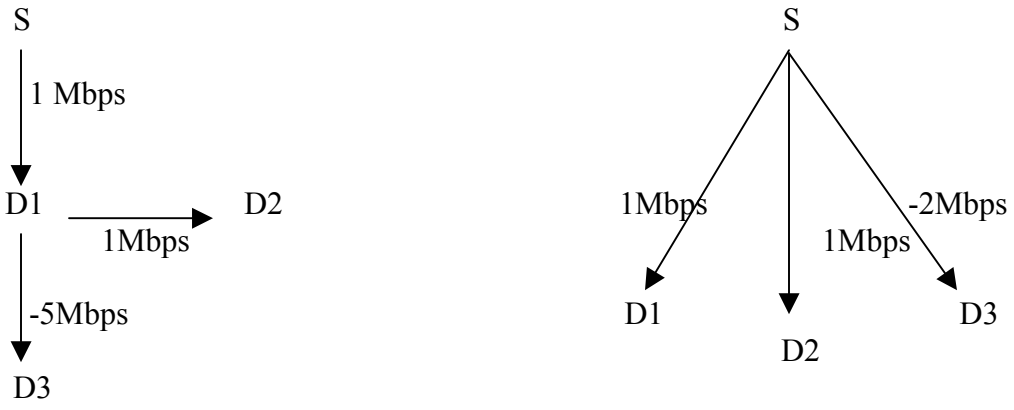
$$Score_j = \alpha * Bw_j + (1-\alpha) * \frac{1}{Lat_j}; 0 \le \alpha \le 1$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} AvgBw_k}{NumConfigs}}; Lat_j = \frac{MaxLat_j}{\frac{\sum_{k=1}^{NumConfigs} MaxLat_k}{NumConfigs}}$$

Here *BWj* is the scaled bandwidth number for the configuration *j*. *BWj* is calculated by taking the average available bandwidth from the source to each of the nodes being used by the configuration *j* and dividing it by bandwidth scalar. *AvgBWj* is the average of available bandwidth from source to all destinations in configuration *j*. The available bandwidth to each destination is the minimum bandwidth available on path from source to each destination. It is calculated as

Available Bandwidth = Original Link Bandwidth – [Bandwidth of Video + Cross Traffic]

The bandwidth scalar is the average of average available bandwidths of all the configurations. This works fine as long as the links are not congested due to added video traffic. If one or more links of a tree configuration are congested due to added video traffic, the average available bandwidth becomes negative for the configuration. If more number of tree configurations have negative average available bandwidths, then there is a high possibility of bandwidth scalar becoming negative. This would mean that the score of a particular configuration would increase as the average available bandwidth lowers. So a tree configuration with a lower available bandwidth would be selected and this would increase the packet loss. An example to explain the above concept is given below. Consider the two tree configurations as show in Figure 12.



Configuration 1 Configuration 2

Figure 12: Multicast Tree Configurations for analyzing the effect of negative Bandwidth Scalar

For Configuration 1:

Average Available bandwidth ($AvgBW_1$) = (1 + 1 – 5) / 3 = -1Mbps

For Configuration 2:

Average Available bandwidth ($AvgBW_2$) = (1 + 1 – 2) / 3 = 0Mbps

Bandwidth Scalar = ($AvgBW_1$ + $AvgBW_2$) /2 = -0.5 Mbps

So if we go with only bandwidth ($\alpha = 1$), then the respective scores will be

Configuration 1: Score = -1 / -0.5 = 2

Configuration 2: Score = 0 / -0.5 = 0

Hence even though the available bandwidth for configuration 1 is less, configuration 1 will be selected over configuration 2 resulting in higher packet loss. Hence the calculation for bandwidth scalar was modified. While calculating the bandwidth scalar, the absolute value of average available bandwidth for each configuration is used. The modified score function is

$$Score_j = \alpha * Bw_j + (1-\alpha) * \frac{1}{Lat_j}; 0 \le \alpha \le 1$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} |AvgBw_k|}{NumConfigs}}; Lat_j = \frac{MaxLat_j}{\frac{\sum_{k=1}^{NumConfigs} MaxLat_k}{NumConfigs}};$$

For the above example the results would be:

For Configuration 1:

Average Available bandwidth ($AvgBW_1$) = (1 + 1 – 5) / 3 = -1Mbps

For Configuration 2:

Average Available bandwidth ($AvgBW_2$) = (1 + 1 – 2) / 3 = 0Mbps

Bandwidth Scalar = ($|AvgBW_1|$ + $|AvgBW_2|$) /2 = 0.5 Mbps

So if we go with only bandwidth ($\alpha = 1$), then the respective scores will be

Configuration 1: Score = -1 / 0.5 = -2

Configuration 2: Score = 0 / 0.5 = 0

Hence configuration 2 will be selected.

3.2 Randomizing Selection of Configuration having same highest score

The base overlay algorithm described in [BrJ03], always chooses the first tree configuration with the highest score. So if there are say four different tree configurations with same highest score then the first will be selected. Tree configurations are generated in a certain order. First tree configurations are generated with the destination being added as a direct child of the source. So the initial trees that are generated are less in height. For such trees with less height there is higher packet loss due to congestion. It has been observed that packet loss due to congestion is the highest for a star topology. If we randomize the tree configurations with same score, there is less probability that trees less in height will be selected.

3.3 Information to calculate available bandwidth in a tunnel

The average bandwidth for this configuration is calculated in the following way:

Average Bandwidth = [*Cost (D1)* + *Cost (D2)*] / 2

Cost (i) = the minimum available bandwidth on any link in the tunnel between *S* and *i.* And this minimum available bandwidth is calculated as Total Link Bandwidth – Bandwidth consumed due to video – Bandwidth consumed due to cross traffic on the link.

To calculate the bandwidth consumed on a link due to video we need the information about the exact number of video packets going over the link or in other words the stress of the link. Hence we need to store the link level information. This would put more burden on the edge routers in a diffserv network. The other way is to calculate the minimum available bandwidth by just neglecting the link level details. This would mean that the algorithm has no idea about how many video packets go over a particular link in a tunnel. It would just take into account that the stress on all the links in a tunnel would be 1. But this would mean that the algorithm might not be able to differentiate between trees on the basis of available bandwidth to the link level and so might end up selecting trees with less available bandwidth that would then result in a slightly higher packet loss. But neglecting link level information while calculating available bandwidth puts less burden on edge routers and hence reduces the complexity and time for calculating the best multicast tree. So it's a trade off between improving the quality of the video versus reducing the time and complexity of the algorithm.

For example, consider a tree configuration with source *S* sending packets to *D1* and *D2* and *N* is just a core node as shown in the Figure 13. For simplicity assume that there is no cross traffic. For the tree configuration shown, we assume bandwidth on all the links is 10Mbps. *S* sends a video of 1Mbps to *D1* and *D2*. If we use the link level information, then stress on link *S* to *N* will be 2Mbps as 2 video packets are being sent. So the average available bandwidth for the tree configuration will be 8 Mbps. But if we neglect the link level information, then the stress on the link from *S* to *N* will be 1Mbps. Hence the average available bandwidth for the tree configuration will be 9 Mbps that is not accurate.

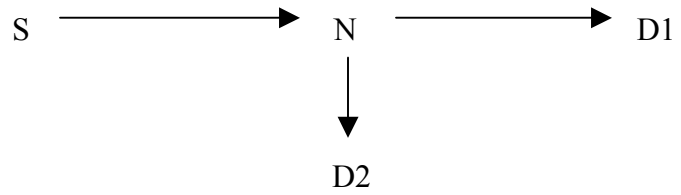


Figure 13: Multicast Tree Configuration for analyzing the effect of information used to calculate available bandwidth in a tunnel.

3.4 Conclusion

After all the enhancements were made to the base overlay algorithm, simulations were run and the results were compared to analyze the behavior of the algorithm, both before and after all the modifications. The results are documented in Chapter 6. The results clearly show an improvement over the base overlay algorithm in terms of reduction of total packet loss.

This refined base algorithm is further used for building DiffServ Overlay Multicast. The next chapter describes in detail the different building blocks of the DiffServ Overlay Multicast.

Chapter 4 – DiffServ Extension of the Base Algorithm

DiffServ Overlay Multicast proposes a solution that relies on only edge routers for replication of packets in the network. DiffServ Overlay Multicast uses the concept of overlay tree where packets are tunneled between edge nodes. Packet replication takes place only at the edge nodes thus eliminating the state information from the core of the network. The distribution tree is constructed using an enhanced version of the base overlay multicast algorithm as described in Chapter 3, which addresses the issue of traffic engineering. DiffServ Overlay Multicast also addresses all three of the conflicts between DiffServ and multicasting. The DiffServ Overlay Multicast architecture trades the penalty of additional bandwidth consumption in order to remove the hardware requirement of DSMCast.

4.1 Supporting different QoS classes for receivers

The enhanced base overlay multicast algorithm can be applied in a DiffServ domain to construct multicast trees where all the receivers demand Best Effort Service. It has to be modified further to support different receivers demanding different QoS classes for the same multicast group. There are two classes of QoS traffic, Expedited Forwarding (EF) and Assured Forwarding (AF), apart from the Best Effort traffic (BE). Although EF class when implemented in a DiffServ network provides a premium service, it should be specifically targeted toward the most critical applications, since if congestion exists, it is not possible to treat all or most traffic as high priority. Since EF is the high priority traffic the EF traffic should be restricted to less than 100% of the link bandwidth so that the lower priority traffic isn't suppressed completely. For EF class it is necessary to have

absolute guarantees in terms of bandwidth and latency. Hence it is necessary to reserve resources in the network. In order to support different QoS classes certain modifications to the enhanced base overlay algorithm are required. First the score function is modified to support QoS. Second the tree construction is modified to take into account the different QoS classes. The remaining part of this chapter discusses these modifications in detail.

4.2 Modifying the Score Function

The score function for single traffic class (Best Effort) is given as follows:

$$Score_j = \alpha * Bw_j + (1-\alpha) * \frac{1}{Lat_j}; 0 \le \alpha \le 1$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} | AvgBw_k |}{NumConfigs}}; Lat_j = \frac{MaxLat_j}{\frac{\sum_{k=1}^{NumConfigs} MaxLat_k}{NumConfigs}}; \quad .....(a)$$

The score function is modified for AF and EF traffic classes. The modifications are shown below in four stages.

4.2.1 End-To-End Latency

End to end latency is the amount of time required for a video packet to travel from source to each destination. This includes the sum of the link propagation delays and the queuing delays on all the links from source to destination. The original score function as stated above uses maximum end to end latency in the score calculation. Recall that MPEG typically requires up to 70 milliseconds to encode and decode a single frame. Also the threshold at which human senses can detect a delay between the signal and the

response is 150 milliseconds. Thus a video packet can stay in the network for a maximum of 80 milliseconds to ensure video quality. If the video packet does not reach the destination within 80 milliseconds then the packet is dropped. Since Call Admission control is implemented for EF traffic, there will be no queuing delay for EF traffic. Thus the end-to-end latency is just the total propagation delay on the links. The maximum end-to-end propagation delay even for a nationwide network is around 40 milliseconds. And as there is no queuing delay for EF traffic the end-to-end latency is also around 40 milliseconds. As long as end-to-end latency is less than 80 milliseconds it does not affect the performance of the video. Hence the latency part of the score function can be eliminated for EF traffic.

4.2.2 Stress

Diffserv overlay multicast distributes the video by setting up overlay tunnels between the edge nodes. Now for the EF and AF class, bandwidth has to be reserved for each overlay tunnel in the multicast tree. For each overlay tunnel the bandwidth has to be reserved between each hop from source to destination of the overlay tunnel. Thus the bandwidth required to be reserved per tunnel in the multicast tree is equal to the bandwidth consumed by the video and can be assumed to be the stress on the tunnel. The stress for i^{th} tunnel can be calculated as

$$Stress_i = Number\ of\ hops\ in\ a\ tunnel_i * Bandwidth\ of\ Video$$

The total bandwidth required to be reserved for the whole multicast tree is equal to the sum of stress on all the tunnels in the multicast tree and can be calculated as

$$Total\ required\ Bandwidth = \sum_{i=1}^{n} Stress_i$$

where n = number of tunnels in multicast tree.

Reserving bandwidth on each tunnel is accompanied with a cost. The cost of reserving bandwidth for a multicast tree is directly proportional to

1. The bandwidth of video.
2. The number of hops in the each tunnel.

$$Cost \ \alpha \ \sum_{i=1}^{n} Stress_i$$

Recall that Diffserv overlay multicast distributes the video by setting up overlay tunnels between the edge nodes. Setting up a tunnel between two edge nodes is also accompanied with an overhead. Each packet has to be included with a tunneling header *T*. Hence along with the bandwidth of the video, extra bandwidth has to be reserved to account for the tunneling header. Assuming that the video is divided into packets of size P_s, each packet is accompanied with a tunneling header of *T* bytes. Thus the overhead can be calculated as $(P_s + T) / P_s$. Hence the overall cost for reserving the bandwidth for the overlay multicast tree can be calculated as

$$Cost = \sum_{i=1}^{n} Stress_i * (P_s + T) / P_s$$

where T = Tunneling header, P_s = Packet Size, n = Number of tunnels and

$$Stress_i = Number\ of\ hops\ in\ a\ tunnel_i * Bandwidth\ of\ Video$$

In order to reduce the cost, the total number of hops in the multicast tree must be reduced. In other words the overall stress of the multicast tree must be reduced. Therefore in order to reduce the cost of the videoconference, stress must be included as a parameter while calculating the score in the score function. After implementing stress parameter in score function and eliminating latency the modified score function is -

$$Score_j = \alpha * Bw_j + (1-\alpha) * \frac{1}{Stress_j}; 0 \le \alpha \le 1$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} | AvgBw_k |}{NumConfigs}}; Stress_j = \frac{TotalStress_j}{\frac{\sum_{k=1}^{NumConfigs} TotalStress_j}{NumConfigs}} .....(b)$$

$Stress_j$ is calculated by determining the sum of stress on all the tunnels in the multicast tree of the current configuration *j* dividing by the average total stress of all the possible configurations. Again, $Stress_j$ indicates how much bandwidth will be consumed by the multicast tree which will influence the cost required for reserving the bandwidth. However, due to the nature of stress, the smaller the value of $Stress_j$, the lesser the amount of bandwidth required to be reserved and hence lower the cost.

4.2.3 Call Admission Control

Since EF class requires certain guarantees in terms of bandwidth it is necessary to generate a multicast tree that satisfies each EF destination in terms of available EF class bandwidth. Hence when the algorithm generates different tree configurations, Call Admission Control is implemented to filter out the trees that do not qualify the available bandwidth requirements. The Call admission control is implemented in the score function as follows.

1. First the minimum available EF bandwidth to all the destinations in the multicast tree is calculated. This information is obtained from bandwidth broker.

2. If the minimum available bandwidth is greater than the bandwidth of the video the tree satisfies bandwidth required for the video and the score is then calculated using the score function. If the minimum available bandwidth is less than the bandwidth of the video then the tree is assigned a score of zero. This is done in order to filter out the tree, as it does not have enough resources to support the video stream.

After implementing CAC in the score function -

$$Score_j = \beta * \left[\alpha * Bw_j + (1-\alpha) * \frac{1}{Stress_j} \right]; 0 \le \alpha \le 1$$

$$MinBw_j = \min\left[MinBw_{S \to D} \right] over\ all\ the\ destinations \quad .....(c)$$

$$\beta = 1, if\ MinBw_j \ge Bandwidthofvideo$$

$$\beta = 0, if\ MinBw_j < Bandwidthofvideo$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} | AvgBw_k |}{NumConfigs}}; Stress_j = \frac{TotalStress_j}{\frac{\sum_{k=1}^{NumConfigs} TotalStress_j}{NumConfigs}}$$

4.2.4 Eliminating α

Simulation results as depicted in section 6.4.1 show that after implementing CAC in the score function, the results are not highly influenced by changing α. In order to reduce the complexity and running time of the algorithm, the parameter α can be eliminated from the score function. After eliminating α from the score function -

$$Score_j = \beta * \left[Bw_j + \frac{1}{Stress_j} \right]; 0 \le \alpha \le 1$$

$$MinBw_j = \min \left[MinBw_{S \to D} \right] \; over \; all \; the \; destinations \quad .....(d)$$

$$\beta = 1, \; if \; MinBw_j \ge Bandwidthofvideo$$

$$\beta = 0, \; if \; MinBw_j < Bandwidthofvideo$$

$$Bw_j = \frac{AvgBw_j}{\frac{\sum_{k=1}^{NumConfigs} | AvgBw_k |}{NumConfigs}}; Stress_j = \frac{TotalStress_j}{\frac{\sum_{k=1}^{NumConfigs} TotalStress_j}{NumConfigs}}$$

4.2.5 Eliminating Bandwidth Parameter (Bw_j)

The value of Bw_j depends on $AvgBw_j$. Here Bw_j is the scaled bandwidth number for the configuration *j* and is calculated by taking the average available bandwidth of the links being used by the configuration *j* and dividing it by bandwidth scalar. $AvgBw_j$ is the average of minimum available bandwidth from source to all destinations in configuration *j*. Also $MinBw_j$ is calculated by taking the minimum of minimum available bandwidth from source to all the destinations in the configuration *j*. From this information it can be concluded that

1. If $MinBw_j > Bandwidthofvideo$, then $AvgBw_j > Bandwidthofvideo$
2. If $MinBw_j < Bandwidthofvideo$, then $AvgBw_j$ is not required as the score would be 0 ($\beta = 0$).

In the first case, for tree configurations that have $MinBw_j > Bandwidthofvideo$, $AvgBw_j$ is not required in order to calculate the score. This is mainly because for these tree configurations available bandwidth to accommodate the video is already present.

Therefore the score should be calculated only to minimize the cost of reserving bandwidth for the tree. Hence Bw_j can be eliminated.

In the second case, since tree configurations have $MinBw_j < Bandwidthofvideo$, Bw_j is again not required in order to calculate the score. This is because for these tree configurations available bandwidth to accommodate the video is not present. Hence the tree configurations are assigned a score of zero to filter them out. Hence Bw_j can be eliminated. Finally after eliminating Bw_j from the score function -

$$Score_j = \beta + \frac{1}{Stress_j}; \begin{bmatrix} \beta = 1, \ if \ MinBw_j \geq Bandwidthofvideo \\ \beta = 0, \ if \ MinBw_j < Bandwidthofvideo \end{bmatrix} .....(e)$$

$$Where \ Stress_j = \frac{TotalStress_j}{\frac{\sum_{k=1}^{NumConfigs} TotalStress_j}{NumConfigs}}$$

The above score function is used as the score function for finding the best multicast tree for DiffServ Overlay Multicast.

4.3 Modifying the Tree Construction

The next thing is to construct multicast trees to connect different receivers demanding different QoS classes. In order to construct a multicast tree in a DiffServ domain for different QoS classes the following set of rules are followed [StB03]:

1. Packets should not be forwarded to a destination (edge router) demanding higher priority PHB from a destination demanding lower priority PHB. For example packets cannot be tunneled to an edge router demanding AF PHB and then tunneled further to an edge router demanding EF PHB.

2. A destination (edge router) should not receive a packet with higher PHB than requested. For example, a packet should not be tunneled to an edge router demanding AF PHB with the EF PHB and then tunneled to an edge router demanding EF PHB.

The tree construction as modified from [BrJ03] based upon the above rules is as follows: When a new node (*n*) wants to join the multicast group it contacts the source (*s*) and sends total stress and bandwidths from *n* to all the other nodes in the multicast group. Apart from this it also sends the information about the desired level of QoS. Source node then determines where to place the new node taking into account the above-mentioned rules. To generate a list of possible configurations, the new node *n* is added at different places in the existing tree. The new node is first added as a direct child of the source (Figure A). Next set of configurations were the ones where the new node is added as a grand child of the source but only if the PHB of the new node is same or lower than that of the child to which it is being added (Figure B). And final set of configurations is where the new node replaces an existing child of the source. The displaced child node however can only be attached to a node demanding same or higher PHB (Figure C). After the configurations are generated, each configuration is assigned a score according to the modified score function described in 4.2.5 (e). The tree with the highest score is selected as the multicast tree.

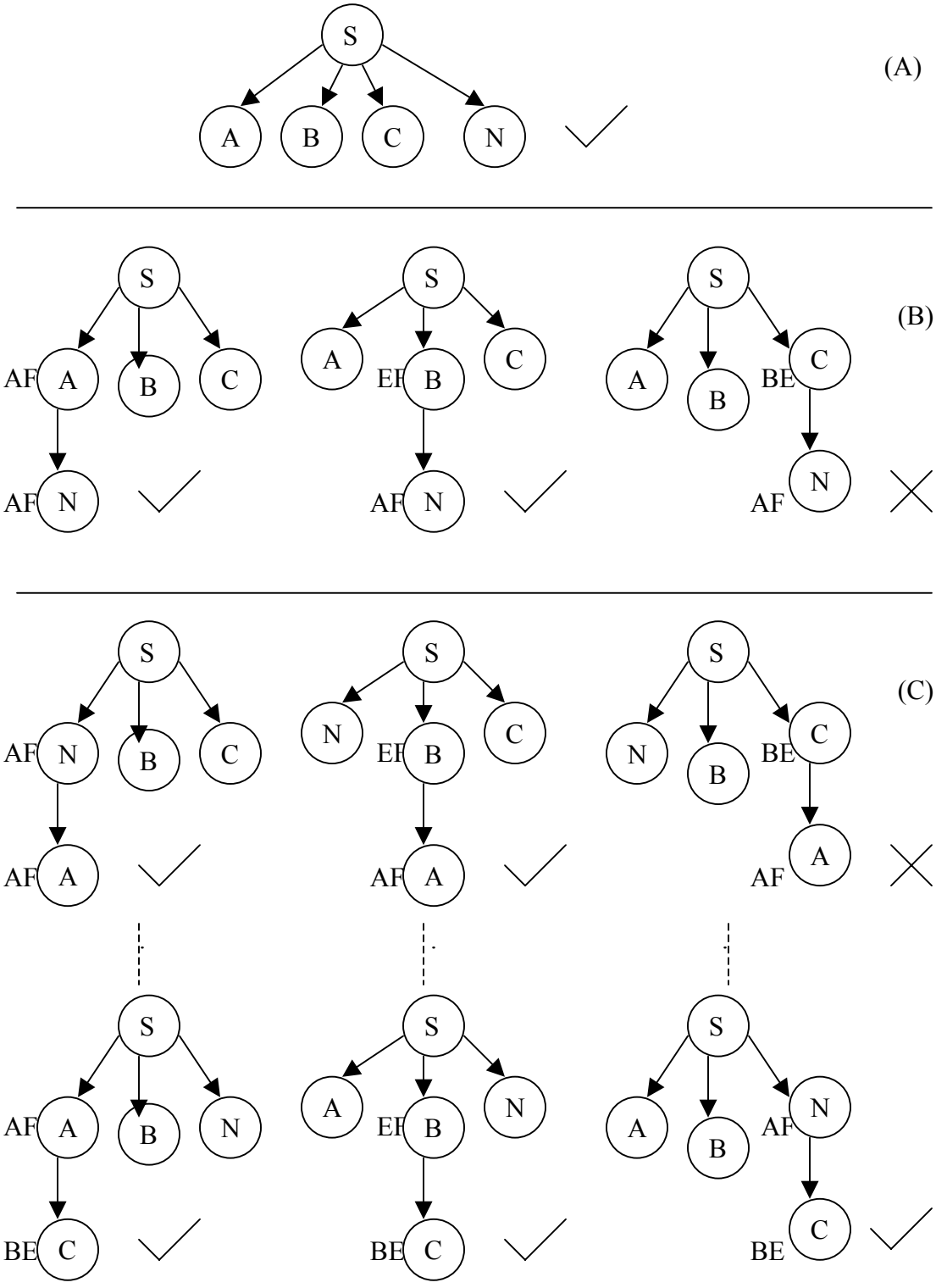


Figure 14: The possible options for adding new node N to an existing tree with different QoS levels.

4.4 Addressing the DiffServ Multicast Issues

1) Stateless Core: In DiffServ Overlay Multicast only edge routers are responsible for packet replication. The multicast overlay tress is constructed using a set of tunnels between the edge routers. Hence only edge routers have to maintain multicast routing information. The core routers only have to do packet forwarding. Hence DiffServ Overlay Multicast maintains a stateless core.

2) Resource management: DiffServ Overlay Multicast uses the concept of a centralized entity for constructing the multicast tree. This entity is known as video conferencing assistance (VA). The VA is responsible for managing join/leave requests and constructing the overlay tree for a particular multicast group. The VA with the help of bandwidth broker is able to easily monitor the resources being consumed by the multicast tree. Hence the VA can check and reserve resources before joining a new receiver to the existing multicast tree. Hence this solves the neglected reservation sub tree (NRS) problem.

3) Dynamics of arbitrary sender change: In DiffServ Overlay Multicast each sender (edge router) runs the modified edge router algorithm to come up with the multicast tree. Hence each sender can reserve resources along its tree. Since resources can be reserved along each tree, simultaneous transmission of data by multiple senders would not result in NRS problem.

4) Heterogeneous QoS support: The modified Edge Router Multicast Algorithm as described in sections 4.1 to 4.3 supports different receivers demanding different level of QoS for the same multicast group. Hence DiffServ Overlay Multicast provides heterogeneous QoS support.

5) Traffic Engineering: All the edge routers constantly monitor network resources for evaluating the bandwidth and total stress to other edge routers in the domain. Edge routers also transmit this information to source router help it determine the best overlay tree for a particular multicast group. After all valid configurations are generated a score is assigned to each configuration according to the score function described in 4.7. The score is based taking into account the total stress and average available bandwidth for each configuration. Hence the DiffServ Overlay Multicast generates trees taking into account the available link bandwidth and stress on the links. Hence it finds alternate paths when there is congestion thus increasing the call admission success ratio and minimizing the average total packet loss for AF class. In addition the stress parameter in the score function also minimizes the cost for reserving EF and AF bandwidth in the diffserv network. Simulations are run to compare DiffServ Overlay Multicast with other DiffServ Multicast algorithms and the details are mentioned in the next couple of chapters. The next chapter presents the details of the simulation setup. Chapter 6 gives in depth analysis of the simulation results.

Chapter 5 – Simulation Setup

5.1 Topology Generation

A typical DiffServ domain consists of a set of nodes randomly connected to each other. A flat topology is generated which consists of nodes connected randomly with a defined probability. There are two sets of nodes in a DiffServ domain, the intelligent DiffServ edge routers and the DiffServ Core routers. The Topology was generated using GT-ITM topology generator from Georgia tech [ZeC96].

The random flat topology was generated using the Waxman topology model proposed by Waxman [WaP88]. According to this model, links were generated between the nodes with a probability given by the following equation:

$P(u,v) = \alpha e^{-d/(\beta L)}$

Where $0 < \alpha, \beta < 1$ are the parameters of the model, d is the Euclidean distance between nodes u and v and L is the maximum distance between any two nodes. Selecting the parameters α=0.2 and β=0.3, a set of 40 different random flat topologies were generated. Each flat topology generated consists of a total of 100 nodes and an average of around 150-170 links. Each link is a full duplex link, thus having a link to node ratio of around 3. Each link has a scaled down bandwidth of 10 Mb/s. The topology generator depending upon the physical length of the links randomly generates the Link latencies.

Depending upon the average link latencies for a topology we have three different DiffServ networks: the Campus/Citywide network, the statewide network and the nationwide network as show in Table 2.

Type of network	Link Length		
	Minimum	Average	Maximum
Campus/Citywide	0.05 miles	1 mile	2.5 miles
Statewide	0.5 miles	10 miles	25 miles
Nationwide	2.5 miles	50 miles	125 miles

Table 2 – Types of DiffServ Networks

5.2 Description of simulation scenarios

Simulations are run for two different scenarios. In the first scenario the simulations are run with single source involved in a videoconference. There are eleven participants with one sender and ten receivers. In the second scenario the simulations are run with multiple sources involved in a videoconference. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers. The results can be grouped in two stages. In the first stage the simulations are run to analyze the effects of enhancing the base algorithm. The score function described in 3.1 is used for calculating the score. The simulations are run over three different topologies depicting campuswide, statewide and nationwide networks. The simulation results are identical for the campuswide and statewide network. Hence only the simulation results for statewide network and nationwide network are documented. For the second stage the simulations are run to analyze the effects of the DiffServ extension of the base algorithm. The score function described in 4.2.5 is used for calculating the score. There is no latency parameter influencing the score. Hence in the absence of latency parameter in the score function, the

simulations results for different topologies viz Campuswide, statewide and nationwide are identical. Hence simulations results are only documented for nationwide topology in stage two. The results are grouped as follows:

Stage 1 – Enhancing the Base Algorithm

In this stage simulations are run to judge the performance of the base overlay multicast algorithm after enhancements made to create more efficient multicast trees. The modifications done in this stage are listed below:

1. Making Bandwidth Scalar Absolute.
2. Randomizing configurations having same score.
3. Information to calculate available bandwidth in tunnels.

First simulations are run to compare the base overlay algorithm after the above modifications with the original algorithm. After the algorithm is modified to create efficient multicast trees, simulations are run to determine the best alpha for statewide and nationwide diffserv domains.

Stage 2 – DiffServ Extension of the Base Algorithm

Finally simulations are run to compare the performance of diffserv overlay multicast after modifying the algorithm further to support receivers demanding different QoS classes. In this stage simulations are run for three different scenarios.

1. All the destinations demanding EF class of service
2. All the destinations demanding AF class of service.
3. Destinations demanding a mix of EF and AF class of service.

First simulations are run to find the best alpha for all the scenarios. Once the best alpha is figured simulations are run to compare diffserv overlay multicast with different existing diffserv multicasting technologies when receivers demand different levels of QoS.

5.3 Cross Traffic Generation

In order to study and compare the performance of different algorithms simulated for multicasting in diffserv domain a certain amount of cross traffic was generated. In a diffserv environment, only the edge nodes would be eligible to generate the cross traffic. The core diffserv nodes would only be able to forward the cross traffic to the respective destination. Hence only the edge nodes are assigned to be sources and destinations of the cross traffic. The cross traffic is a mixture of TCP and UDP traces. The volume of cross traffic is set randomly depending upon the type of diffserv network and the type of algorithm we are simulating. Tables 3 and 4 provide the details for the generation of cross traffic for the different scenarios for the two stages.

Stage 1 - For Simulating the Enhanced Base Algorithm

The sending rate is 5Mbps for all traces.

Type of network	Single Source	Multiple Sources
Campus/Citywide	40 TCP Traces 10 UDP Traces	20 TCP Traces 5 UDP Traces
Statewide	40 TCP Traces 10 UDP Traces	20 TCP Traces 5 UDP Traces
Nationwide	20 TCP Traces 10 UDP Traces	10 TCP Traces 5 UDP Traces

Table 3 – Number of traces for generation of cross traffic for different scenarios for simulation of enhanced base algorithm.

Stage 2 - For Simulating the DiffServ extension of the Base Algorithm

Type of network	Single Source	Multiple Sources
Campus/Citywide/Statewide/Nationwide	20 UDP Traces EF Class Traffic Sending Rate: 3Mbps	20 UDP Traces EF Class Traffic Sending Rate: 2Mbps

Table 4 – Number of traces for generation of cross traffic for different scenarios for simulation of DiffServ extension of the base algorithm.

5.4 Performance Metrics

Stage 1 – Enhancing the Base Algorithm

In this stage two performance metrics, Latency and Packet Loss, are used to evaluate the video quality.

1. Latency:

Latency is used as a parameter to determine the end-to-end delay for the video packet to reach the destinations. The maximum delay a packet can tolerate is 80 milliseconds. If the packet is delayed more than 80 milliseconds, then the packet is dropped this affects the video quality. Hence latency is evaluated to keep the packet loss due to delay minimum.

2. Packet Loss:

Packet loss is used as a parameter to determine the quality of video during the videoconference. There are two types of packet loss in the simulations. There is packet loss due to congestion in the network. This is mainly due to the fact that, if a packet arrives at the router and the router queues are full, then the packet is dropped. The second type of packet loss comes from excessive delay. If the packet traverses through a path that has many routers on its way, the packet might spend a longer period of time and might violate the delay bound of 80 milliseconds. Thus the packet might be dropped due to excessive delay.

Simulations are run and both these performance metrics are evaluated to find the best balance to minimize the total packet loss which is a combination of packet loss due to congestion and packet loss due to delay.

Stage 2 – DiffServ Extension of the Base Algorithm

In this stage four performance metrics are evaluated in order to determine quality of the video during videoconference.

1. Call Admission Control acceptance percentage:

Call admission control (CAC) acceptance percentage is an important performance metric for receivers demanding EF class traffic. When a receiver demanding EF class wants to join the videoconference, the CAC module determines if there is available EF bandwidth to accommodate the new receiver in the multicast tree. If required bandwidth is available then CAC accepts the connection or else rejects it. In order to compare the performance of the DiffServ Overlay Multicast with other existing DiffServ multicast technologies this metric is very important. Higher CAC acceptance percentage results in better connectivity and minimum wait for the participant to join the conference.

2. Cost for reserving EF bandwidth:

Cost for reserving EF bandwidth in the network depends on the total number of stress on all the EF tunnels in the multicast tree and also on the extra header information sent along with the packet. The header information varies with different diffserv multicast technologies. The cost is calculated using the following –

$$Cost = \sum_{i=1}^{n} Stress_i * (P_s + T) / P_s$$

where T = Tunneling header, P_s = Packet Size, n = Number of tunnels and

$$Stress_i = Number\ of\ hops\ in\ a\ tunnel_i * Bandwidth\ of\ Video$$

While calculating the cost the following parameters are used.

1. Bandwidth of video = 1Mbps
2. T = Tunneling header = 12 bytes
3. P_s = Packet Size = 1500 bytes

In order to calculate the cost in terms of dollars to reserve EF bandwidth, the cost can be calculated using the general formula and then later the cost can be multiplied with a scaling factor to reserve 1 Mbps. For e.g. the cost in dollars for reserving 1 Mbps of EF traffic is X dollars. The cost for reserving EF bandwidth for the conference is 20 Mbps. Then the total cost in terms of dollars would be 20 X dollars.

3. AF Packet Loss:

The third performance metric is the AF packet loss. Since AF is not the highest priority traffic, in case of congestion the AF packets can be dropped in comparison to the EF packets. This will not affect the receivers demanding EF class, but those demanding AF class in the videoconference. This metric is useful in order to compare the performance of the DiffServ Overlay Multicast with other existing DiffServ multicast technologies. Lower the AF packet loss better will be the video quality for AF recipients. To calculate AF Packet loss for a multicast tree configuration, minimum available bandwidth is calculated for each AF tunnel in the multicast tree. A parameter '*p*' is

assigned to each tunnel for the purpose of AF Packet loss calculation. The value assigned to p is based on the following rules

1. If $MinBw_j \geq Bandwidthofvideo$, then $p = 1$
2. If $0 < MinBw_j < Bandwidthofvideo$, then $p = MinBw_j / Bandwidthofvideo$
3. If $MinBw_j \leq 0$, then $p = 0$

After assigning appropriate values for the parameter p for each AF class tunnel in the multicast tree the average AF packet loss is calculated as follows

$$Average\ AF\ Packet\ Loss\,(\%) = \left[1 - \frac{\sum_{i=1}^{n} P_i}{n} \right] * 100$$

4. Cost for reserving AF bandwidth:

Cost for reserving AF bandwidth in the network depends on the total number of stress on all the AF tunnels in the multicast tree and also on the extra header information sent along with the packet. The header information varies with different diffserv multicast technologies. The cost is calculated using the following –

$$Cost = \sum_{i=1}^{n} Stress_i * (P_s + T) / P_s$$

where T = Tunneling header, P_s = Packet Size, n = Number of tunnels and

$$Stress_i = Number\ of\ hops\ in\ a\ tunnel_i * Bandwidth\ of\ Video$$

While calculating the cost the following parameters are used.

1. Bandwidth of video = 1Mbps
2. T = Tunneling header = 12 bytes
3. P_s = Packet Size = 1500 bytes

Cost of reserving AF bandwidth in terms of dollars is calculated in the same way as that for EF class. The only difference would be that the cost to reserve 1 Mbps of AF traffic would be less compared to EF traffic. For e.g. if the cost in dollars for reserving 1 Mbps of EF traffic is X dollars, then the cost in dollars for reserving 1 Mbps of AF traffic is 0.5X dollars. The ISP providing the service can decide this cost.

5.5 Description of Graphs

Stage 1 – Enhancing the Base Algorithm

Each simulation results in a set of three graphs. The first graph determines the packet loss due to congestion in the network. Second graph tells us average latencies required to send packets from source to respective destinations. Finally the third graph tells us the total packet loss for each destination node. Total packet loss is the sum of packet loss due to congestion and packet loss due to delay. The goal is to minimize the total packet loss to improve the quality of the video. Lowering the total packet loss will better the video quality.

Stage 2 – DiffServ Extension of the Base Algorithm

There are four different graphs for the simulation results in this stage. The first graph determines the acceptance percentage for Call Admission Control. Second graph tells us the cost of reserving EF bandwidth per EF destination in the multicast tree. The third graph tells us the cost of reserving the AF bandwidth per AF destination in the multicast tree. The final graph describes the total packet loss for AF class per AF destination. The first two graphs describe the performance for EF class while the third

and the fourth describe the performance of the video for AF class. The goal is to maximize the call admission control acceptance percentage and minimize the total packet loss to improve the quality of the video. At the same time the cost for reserving resources (bandwidth) must also be minimized.

Chapter 6 – Simulation Results

6.1 Enhancement of the Base Algorithm

The base overlay algorithm is modified to improve the quality of video during videoconference. The modifications done to the algorithm are listed as follows –

1. Modifying the calculation for Bandwidth Scalar.
2. Randomizing selection of configurations having same highest score.
3. Information to calculate available bandwidth in tunnels.

After modifying the algorithm as mentioned above, simulations are run over statewide and nationwide networks for single and multiple source videoconferences. These results are used as baseline results and are compared to the simulation results after individually switching off each enhancement just to study how each modification affects the video quality during a videoconference.

6.1.1 Modifying the calculation for Bandwidth Scalar

In order to study the effect of modifying bandwidth scalar, the modified algorithm (modified score function) with all three enhancements is compared to the algorithm in which bandwidth scalar modification is switched off (original score function).

Single Source Statewide Network

The simulations are run with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. The simulations are run over a single source statewide network. Figure 15 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network for original score function. Figure 16 displays the simulation results

for single video source sending an mpeg stream across a diffserv statewide network for modified score function.

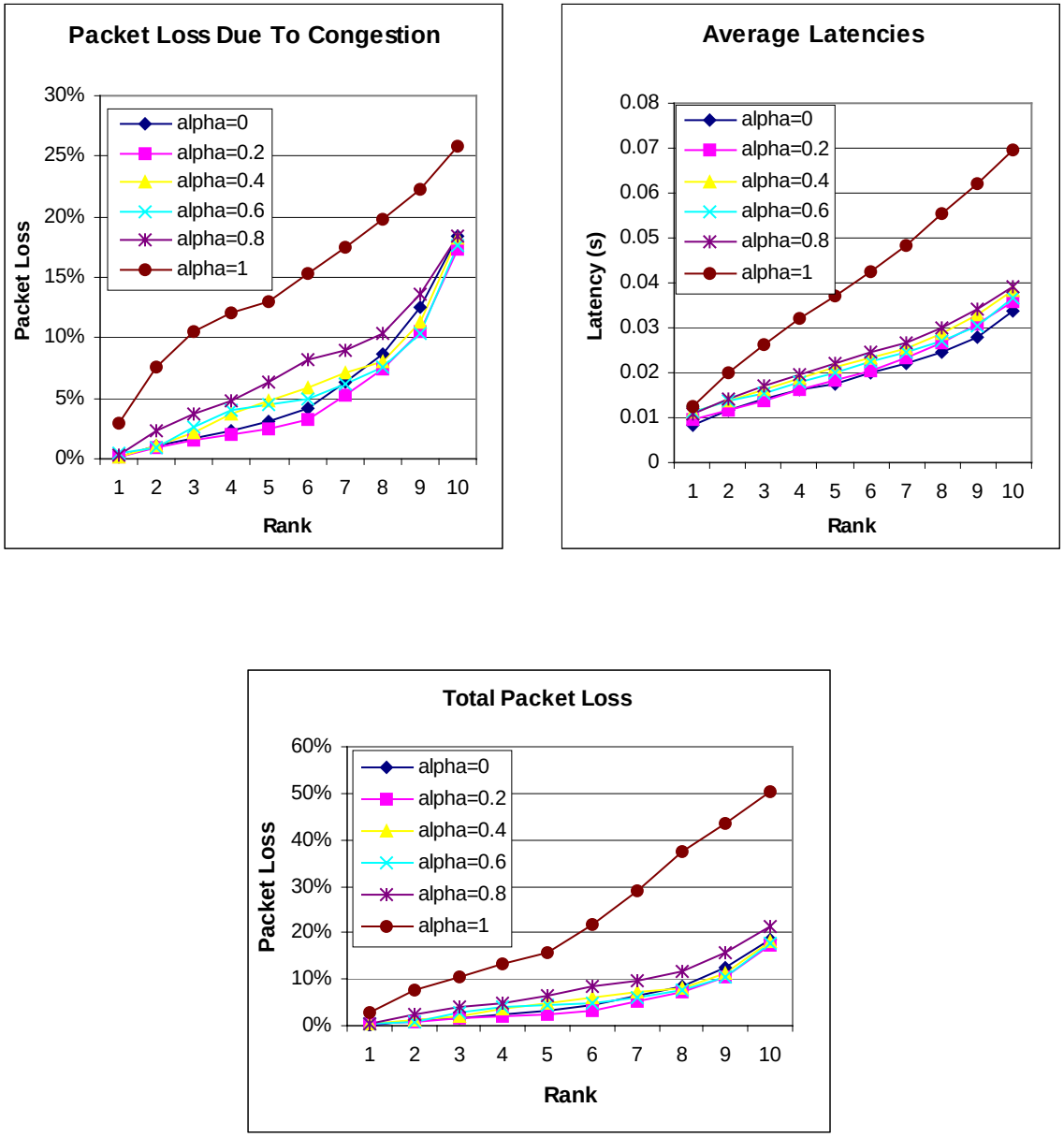
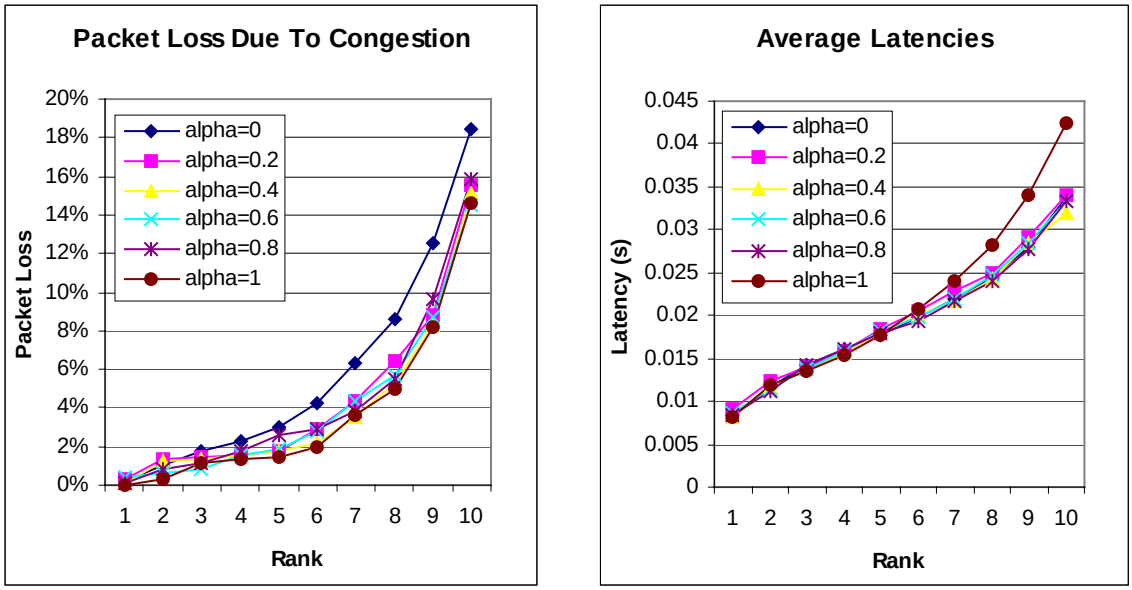


Figure 15: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted for original score function.



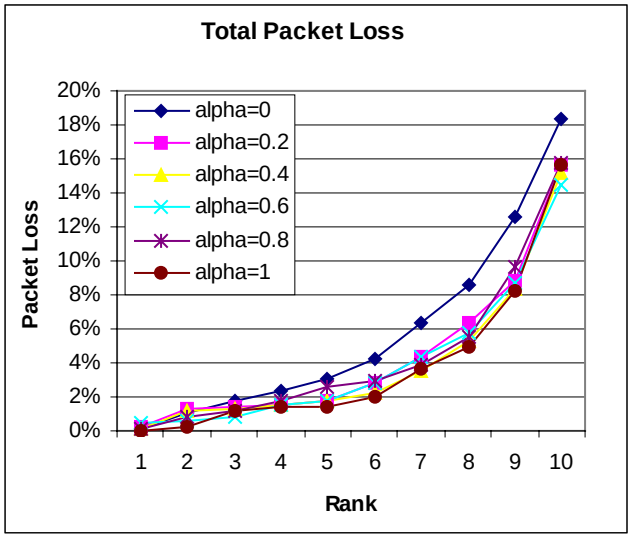


Figure 16: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted for modified score function.

Figure 17 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss for algorithm with all three enhancements (modified score function) vs algorithm

with bandwidth scalar modification switched off (original score function). Recall that for the original score function scenario, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.1. The total packet loss is lowest for $\alpha = 0.2$ and is nearly 17%. For the modified score function, $\alpha = 0.6$ has the lowest packet loss of 14.5%. Thus the packet loss is decreased by 3 % for the worst rank by modifying the score function.

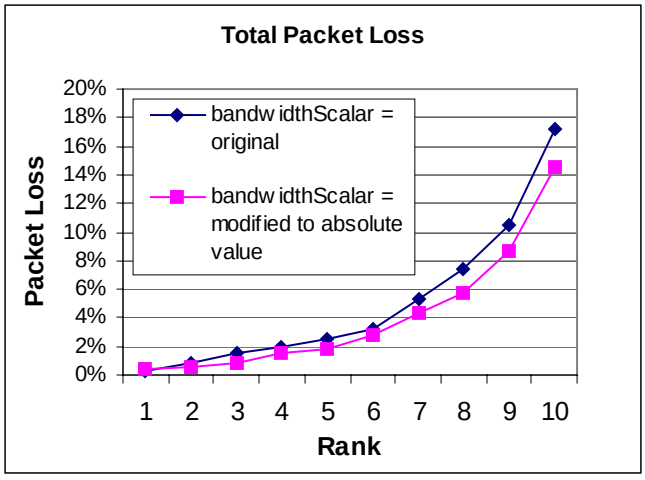


Figure 17: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss for original vs modified score function.

Multiple Sources Statewide Network

Next the simulations are run with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 18 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network for original score function. Figure 19 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network for modified score function.

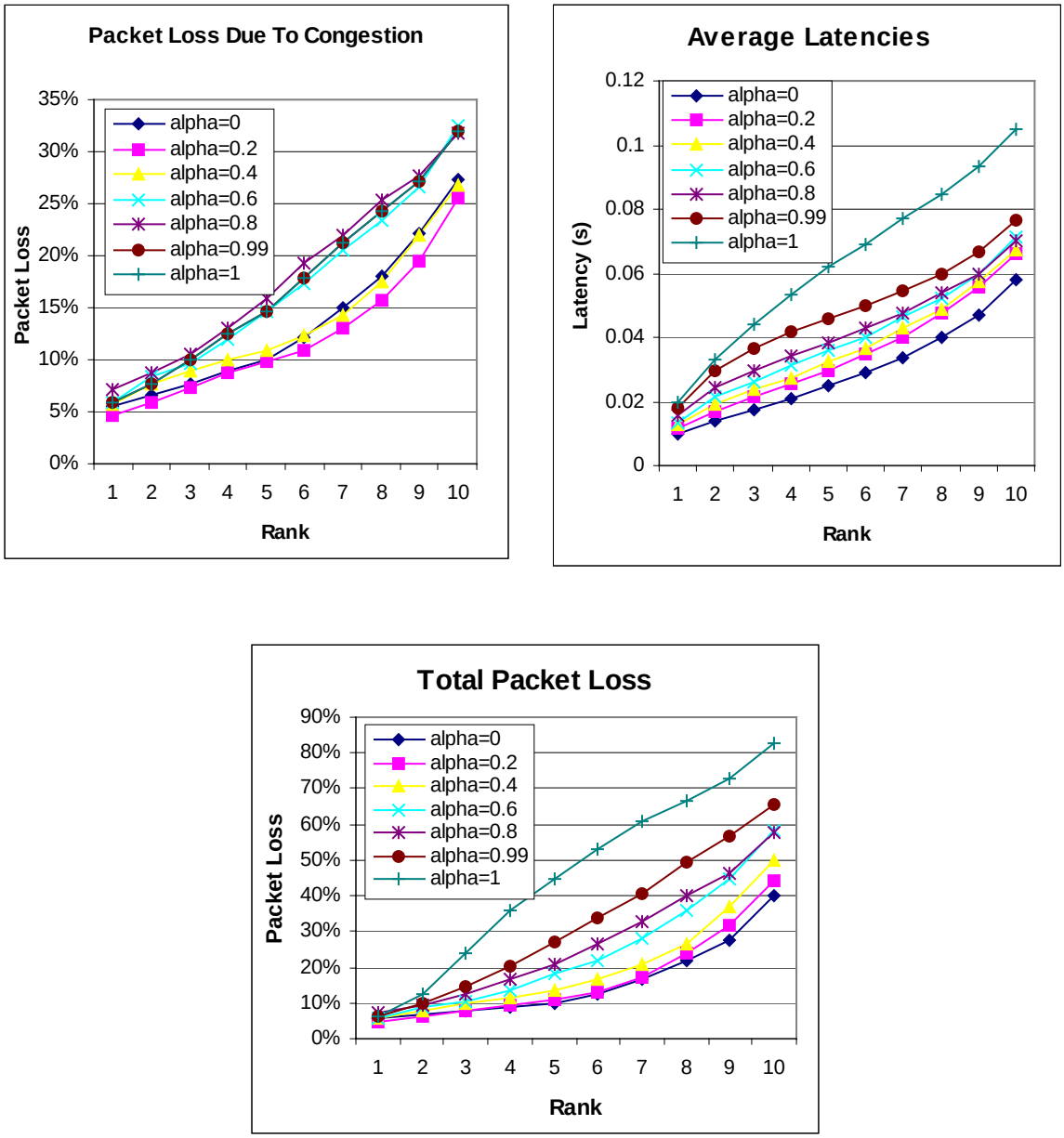


Figure 18: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted for original score function.

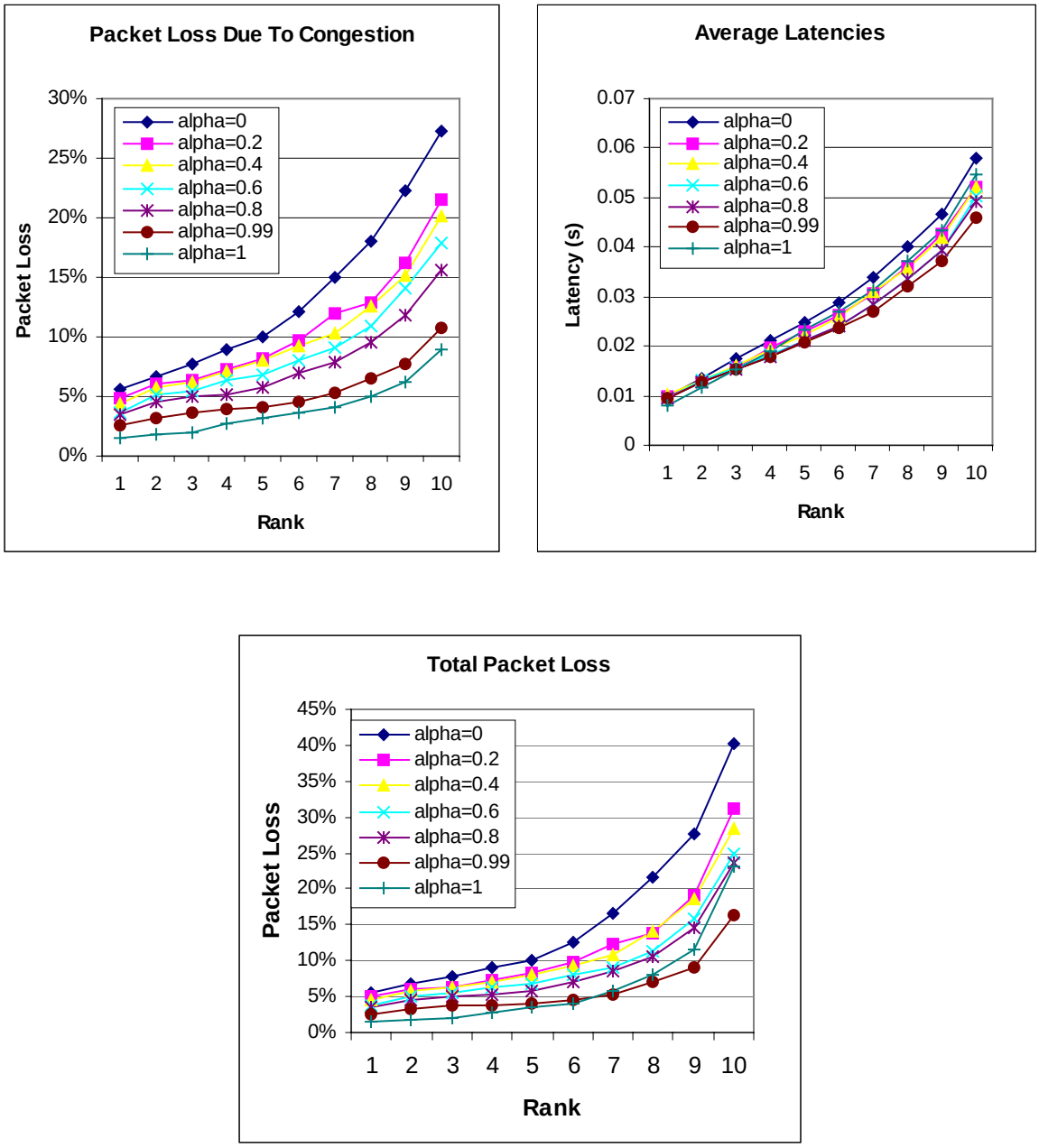


Figure 19: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted for modified score function.

Figure 20 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss for algorithm with all three enhancements (modified score function) vs

algorithm with bandwidth scalar modification switched off (original score function). Recall that for the original score function scenario, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.1. The total packet loss is lowest for $\alpha = 0$, where the trees go by latency and is nearly 40 %. For the modified score function, α =0.99 has the lowest packet loss of 16%. Thus the packet loss is decreased by 24 % for the worst rank by modifying the score function.

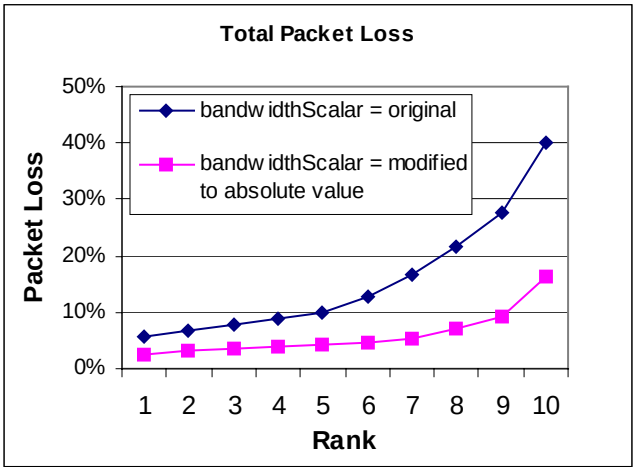


Figure 20: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss for original vs modified score function.

Single Source Nationwide Network

The simulations are run with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. Figure 21 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network for original score function. Figure 22 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network for modified score function.

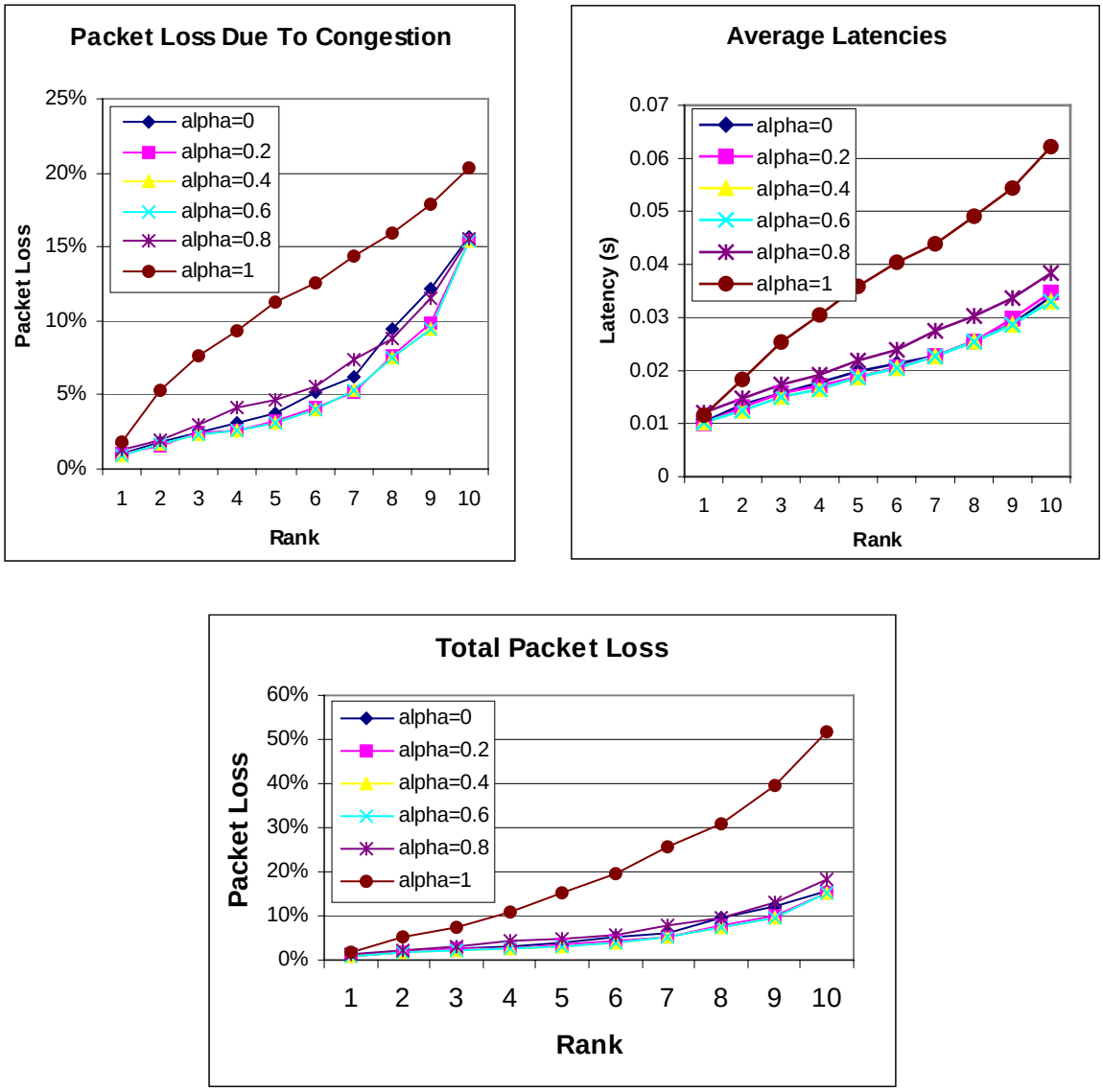


Figure 21: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted for original score function.

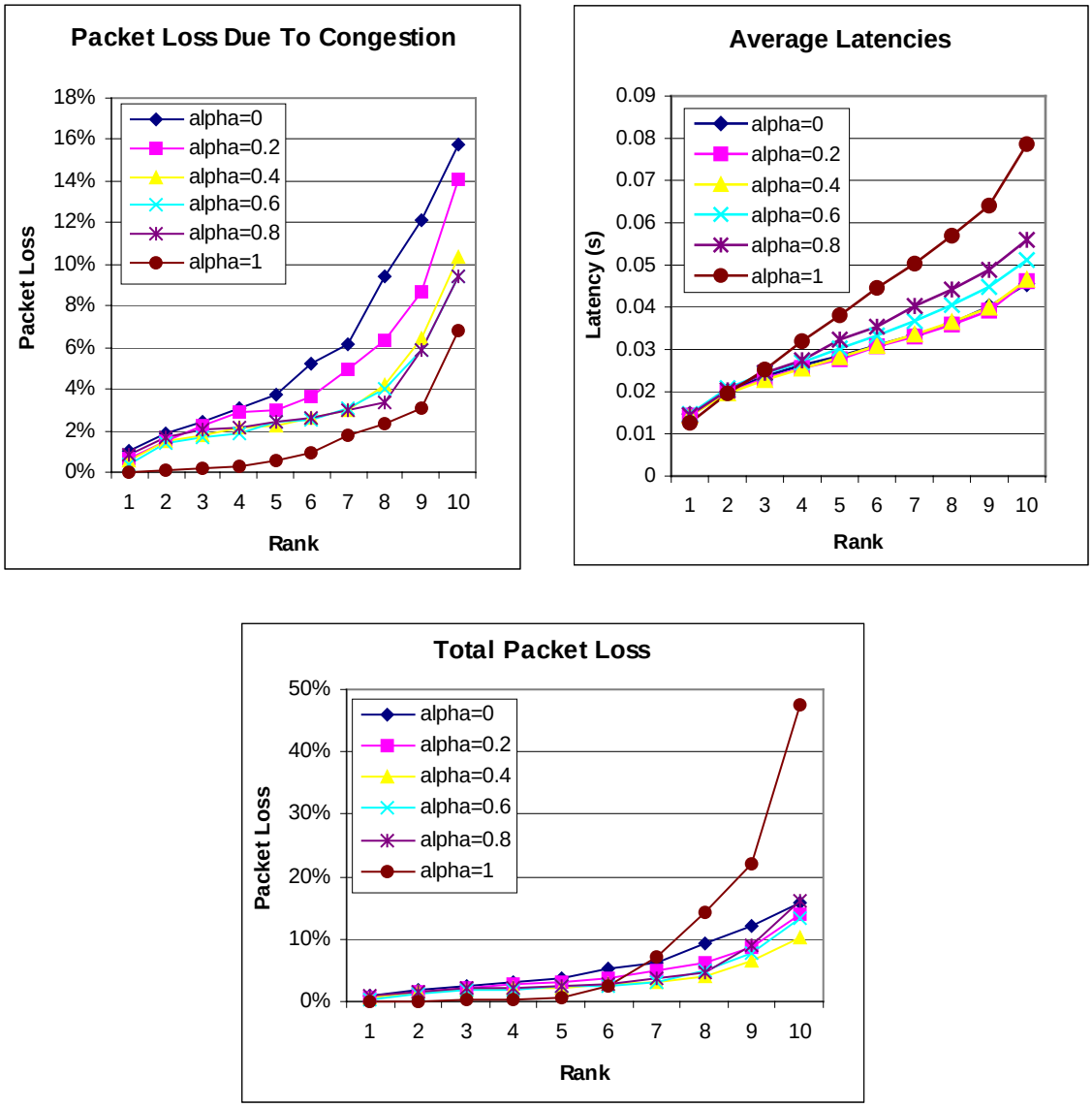


Figure 22: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted for modified score function.

Figure 23 displays the simulation results for single video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss for original vs modified score function. Recall that for the original score function scenario, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in

3.1. The total packet loss is lowest for $\alpha = 0.4$, and is nearly 15%. For the modified score function, $\alpha = 0.4$ has the lowest packet loss of 10%. Thus the packet loss is decreased by 5 % for the worst rank by modifying the score function.

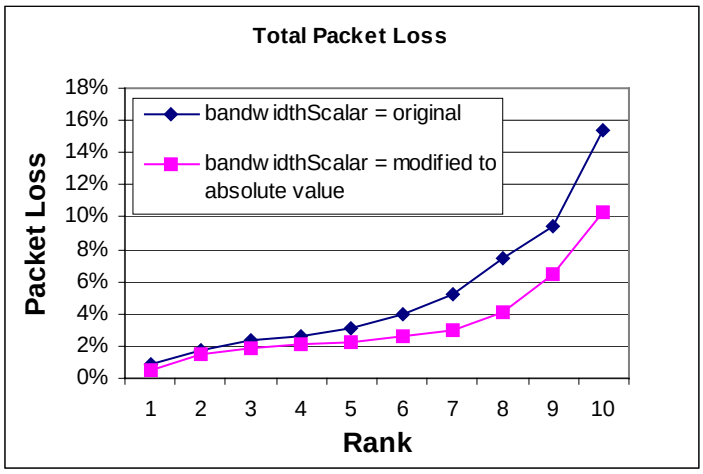


Figure 23: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss for original vs modified score function.

Multiple Sources Nationwide Network

Next the simulations are run with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 24 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network for original score function. Figure 25 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network for modified score function.

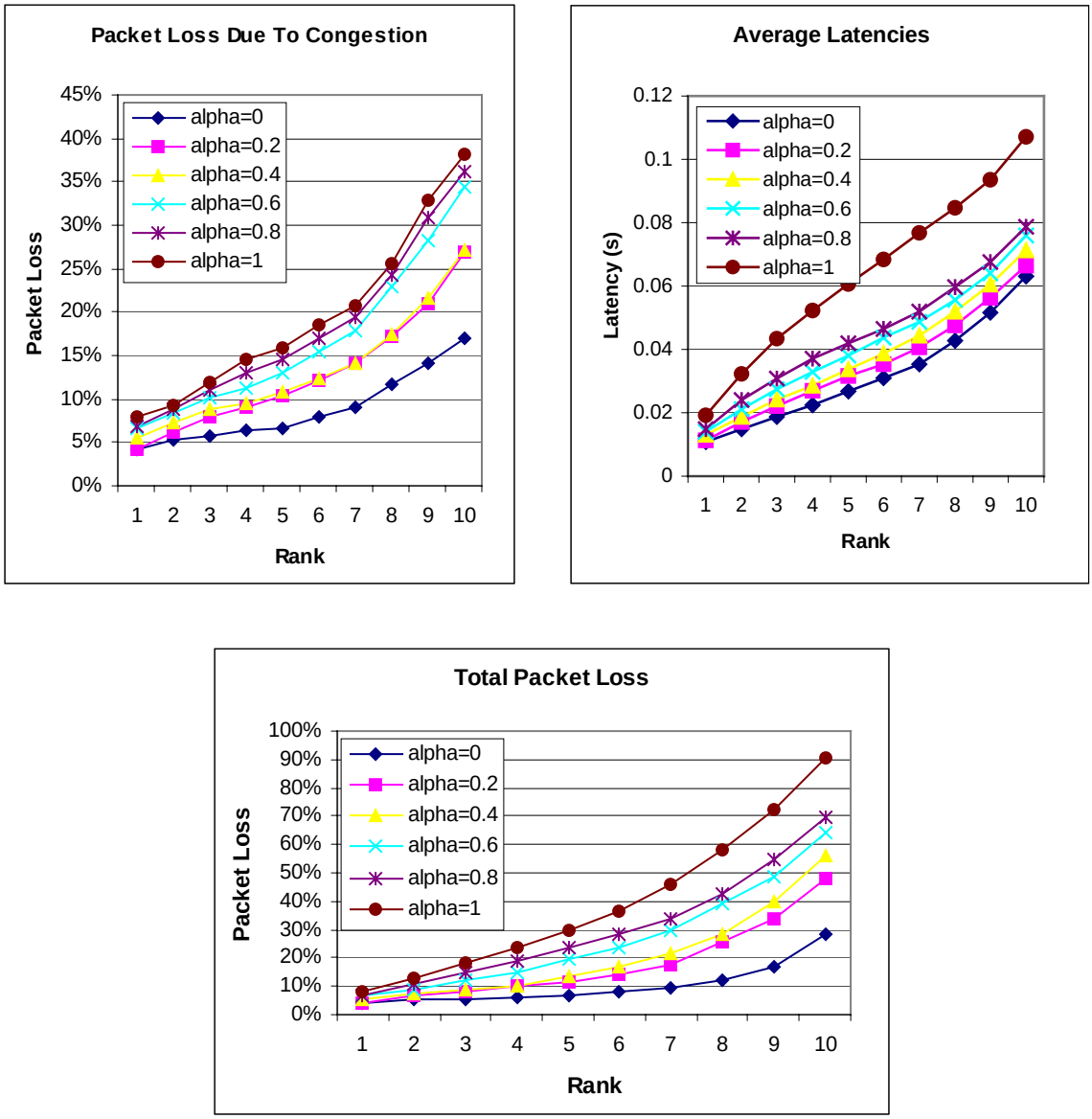


Figure 24: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted for original score function.

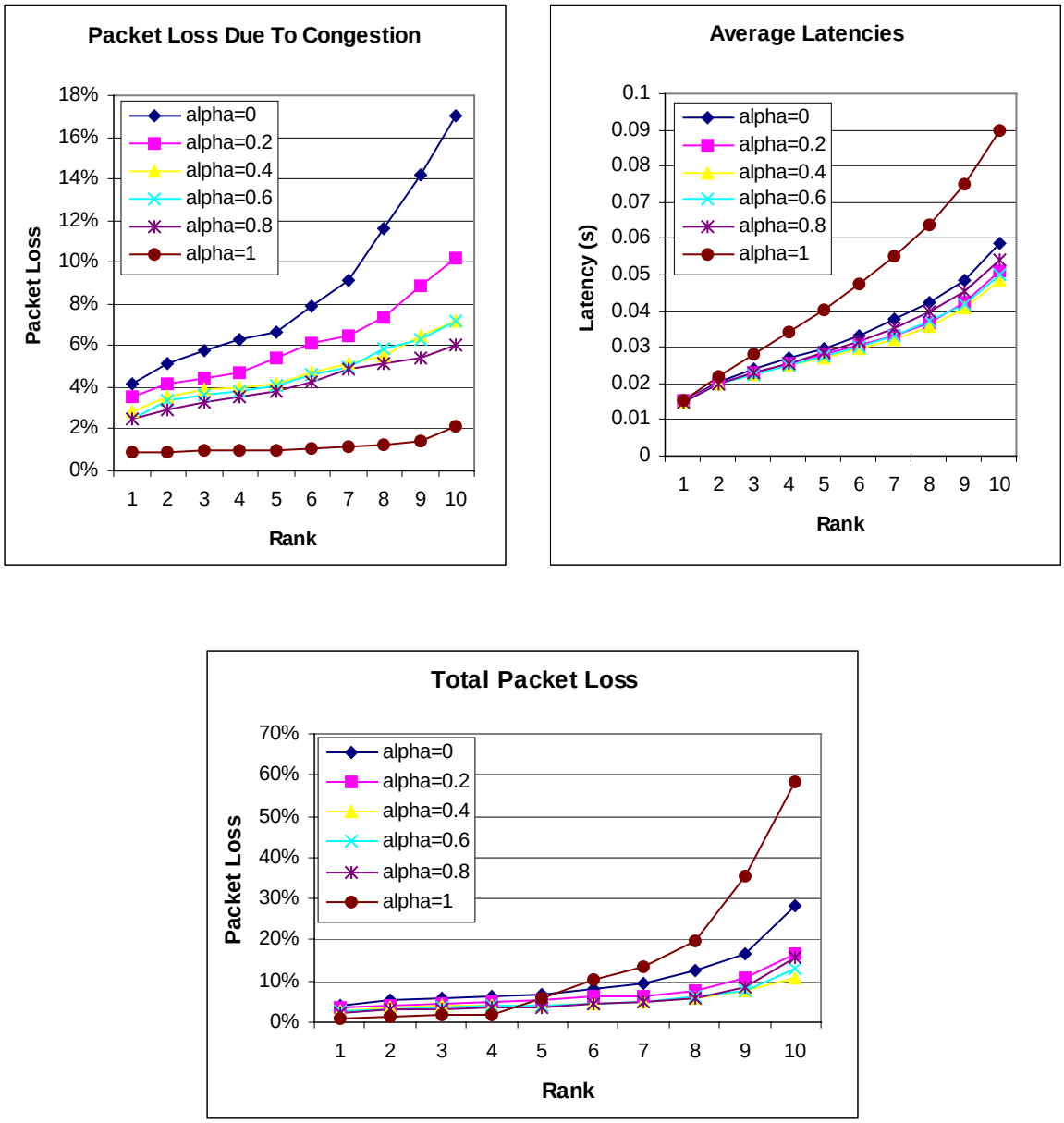


Figure 25: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted for modified score function

Figure 26 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss for original vs modified score function. Recall that for the original score function scenario, large number of packets are lost due to congestion in the network since

the algorithm is selecting trees with less available bandwidth on the links as described in 3.1. The total packet loss is lowest for $\alpha = 0$, where the trees go by latency and is nearly 29 %. For the modified score function, $\alpha = 0.4$ has the lowest packet loss of 11%. Thus the packet loss is decreased by 18 % for the worst rank by modifying the score function.

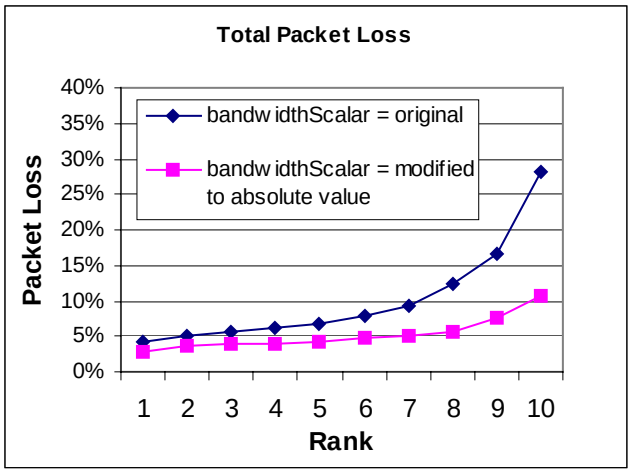


Figure 26: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss for original vs modified score function.

6.1.2 Randomizing selection of Configurations having same highest score

In order to study the effect of randomizing the selection of configurations having same highest score, the modified algorithm with all there enhancements is compared to the algorithm in which randomizing the selection of configurations having same highest score is switched off.

Single Source Statewide Network

The simulations are run with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. Figure 27 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network before randomizing the selection of the configurations having

same highest score. These results are compared with simulation results for single video source sending an mpeg stream across a diffserv statewide network after randomizing the selection of configurations having same highest score that are displayed in Figure 16.

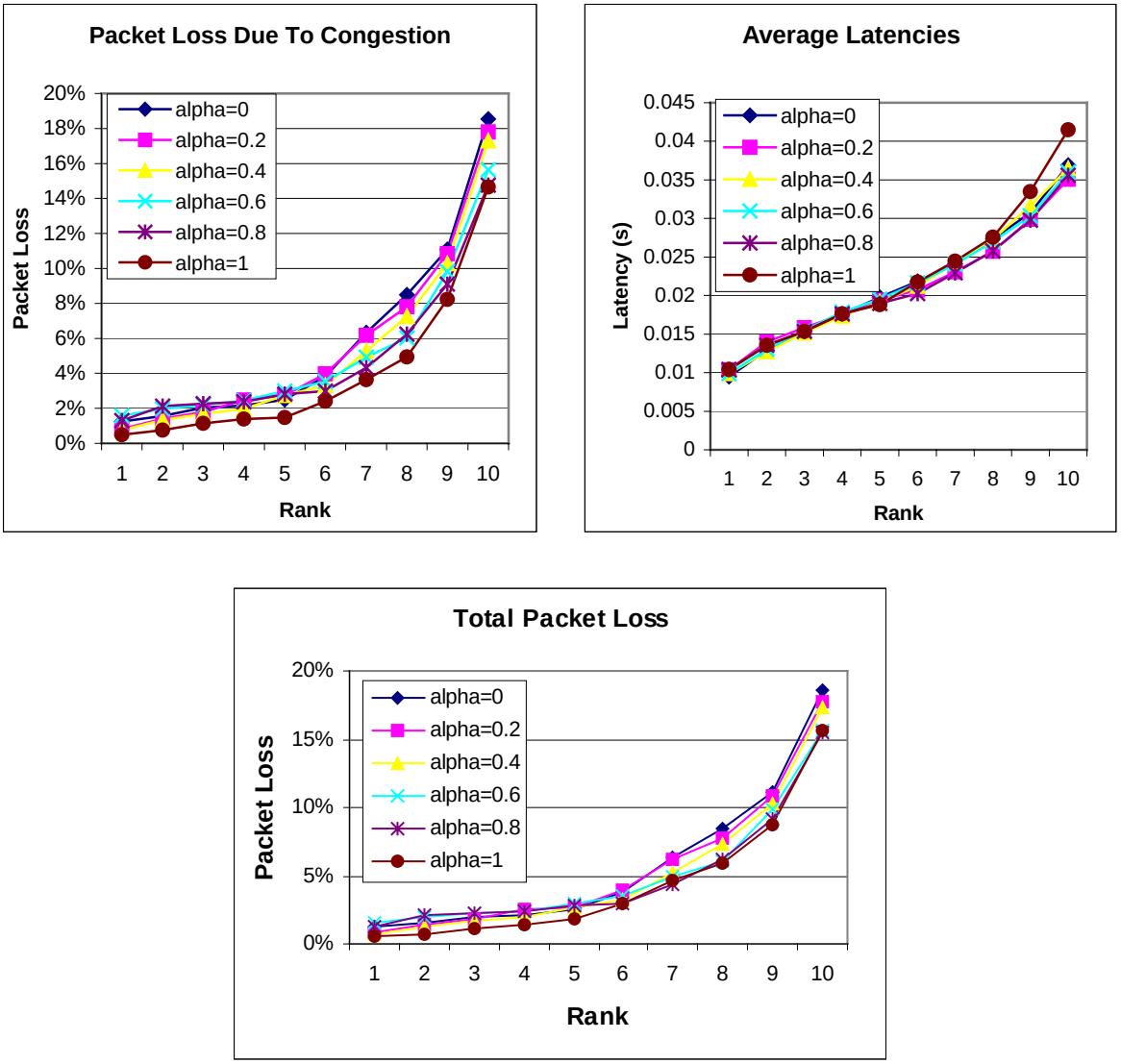


Figure 27: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted before randomizing the selection of configurations having same highest score.

Figure 28 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score. Recall that before randomizing the selection of trees with same highest score, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.2. The total packet loss is lowest for $\alpha = 1$ and is nearly 16 %. For the modified score function, $\alpha = 0.6$ has the lowest packet loss of 14.5%. Thus the packet loss is decreased by 1.5 % for worst rank by randomizing the selection of trees having same score for all values of α.

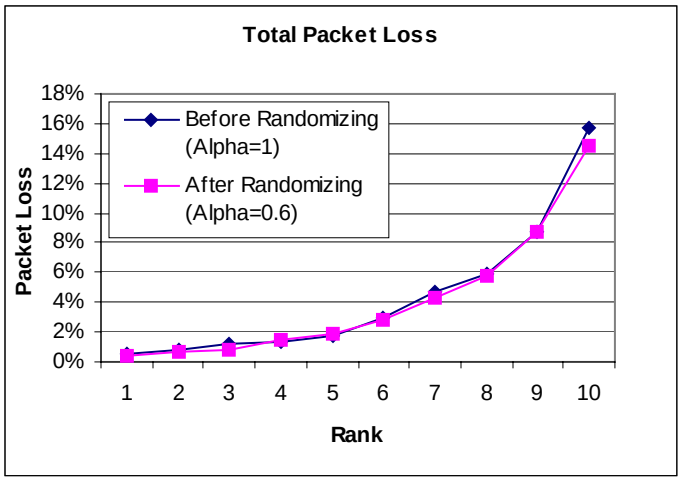


Figure 28: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score.

Multiple Sources Statewide Network

Next the simulations are run using Waxman statewide topology with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 29 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network before

randomizing the selection of configurations having same highest score. These results are compared with simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network after randomizing the selection of configurations having same highest score that are displayed in Figure 19.

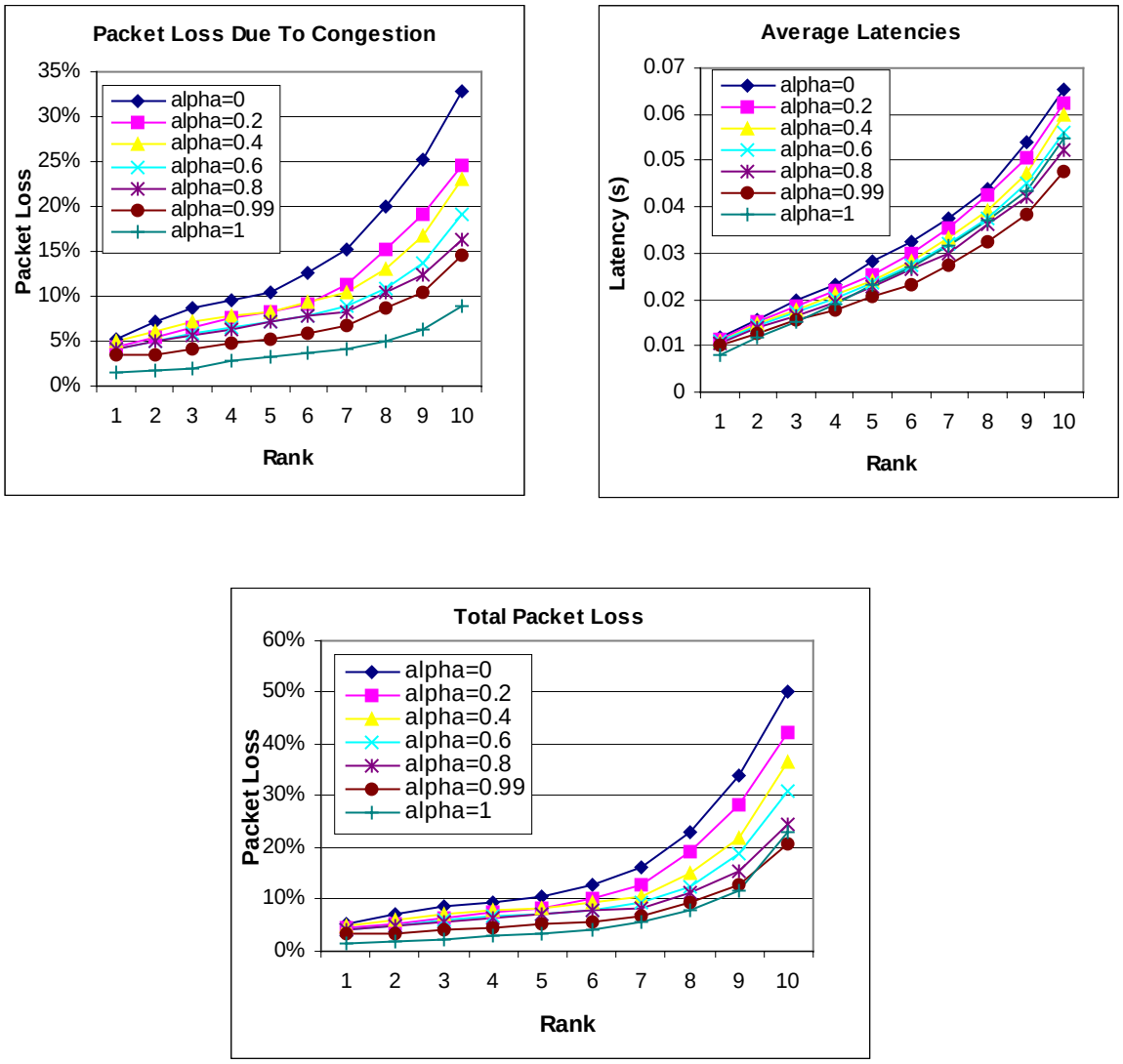


Figure 29: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted before randomizing the selection of configurations having same highest score.

Figure 30 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score. Recall that before randomizing the selection of trees with same highest score, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.2. The total packet loss is lowest for $\alpha = 0.99$ and is nearly 20.5 %. For the modified score function, $\alpha = 0.99$ has the lowest packet loss of 16%. Thus the packet loss is decreased by 4.5 % for the worst rank by randomizing the selection of trees having same score.

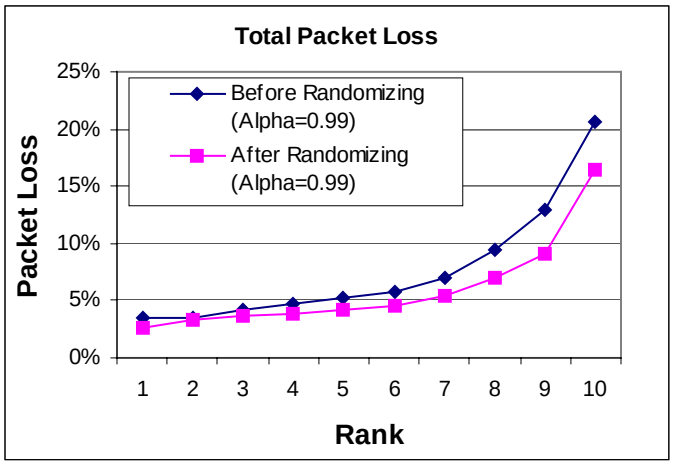


Figure 30: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score.

Single Source Nationwide Network

Next the simulations are run using Waxman nationwide topology with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. Figure 31 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network before randomizing

the selection of the configurations having same highest score. These results are compared with simulation results for single video source sending an mpeg stream across a diffserv nationwide network after randomizing the selection of configurations having same highest score that are displayed in Figure 22.

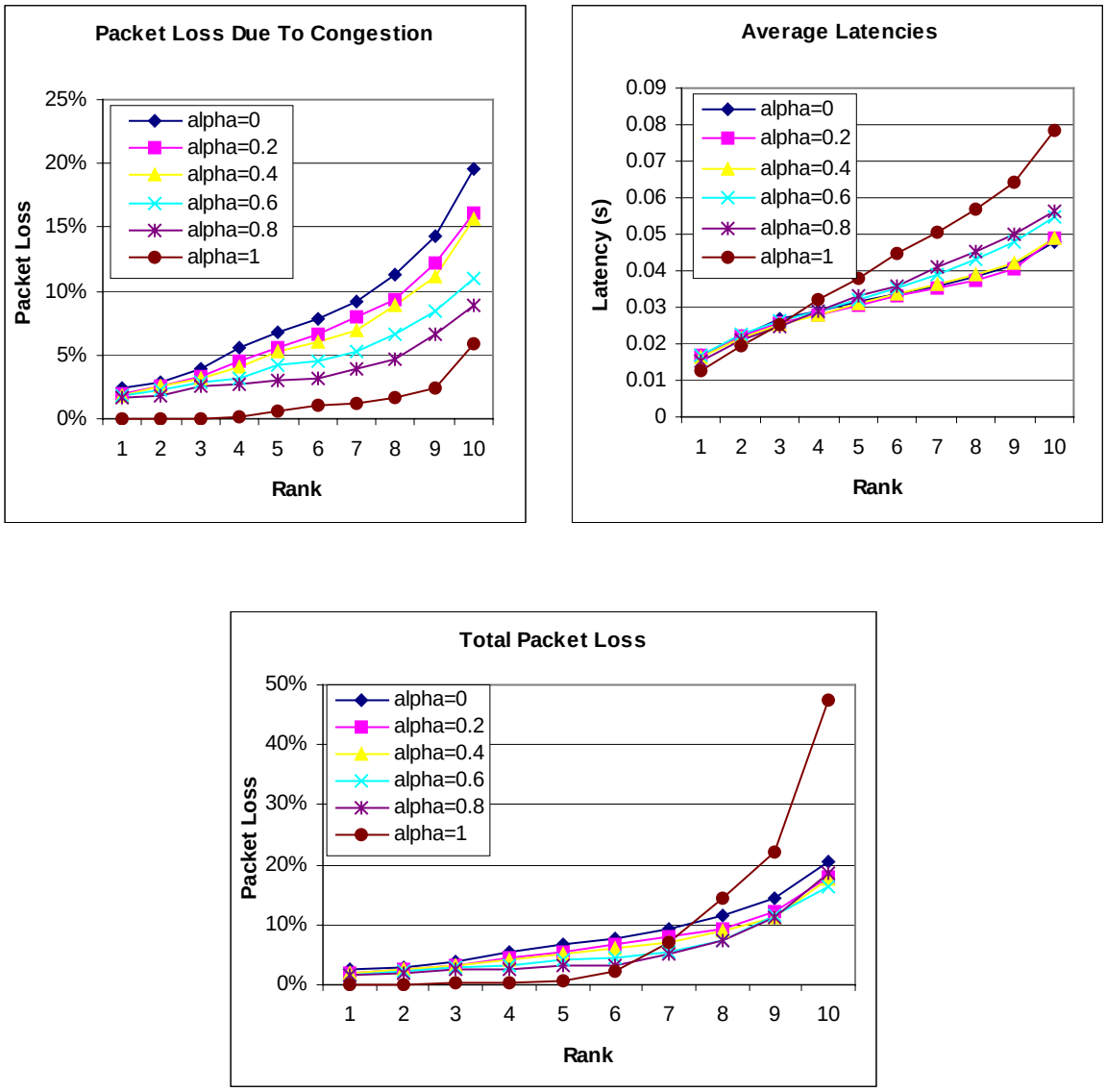


Figure 31: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted before randomizing the selection of configurations having same highest score.

Figure 32 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score. Recall that before randomizing the selection of trees with same highest score, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.2. The total packet loss is lowest for $\alpha = 0.6$ and is nearly 16 %. For the modified score function, $\alpha = 0.4$ has the lowest packet loss of 10%. Thus the packet loss is decreased by 6 % for the worst rank by randomizing the selection of trees having same score for all values of α.

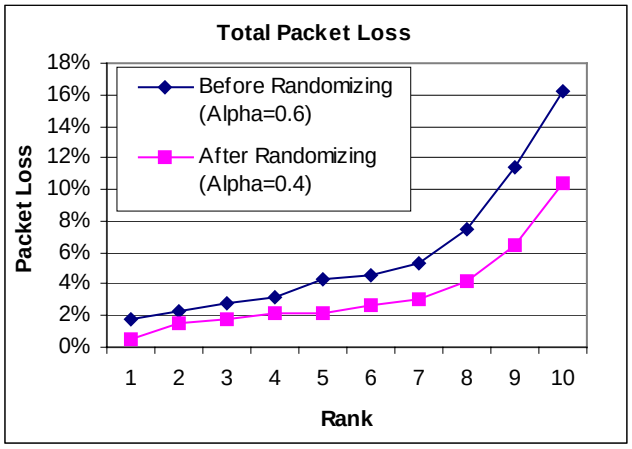


Figure 32: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score.

Multiple Sources Nationwide Network

Next the simulations are run using Waxman statewide topology with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 33 displays the simulation results for multiple

video sources sending an mpeg stream across a diffserv nationwide network before randomizing the selection of configurations having same highest score. These results are compared with the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network after randomizing the selection of configurations having same highest score that are displayed in Figure 25.

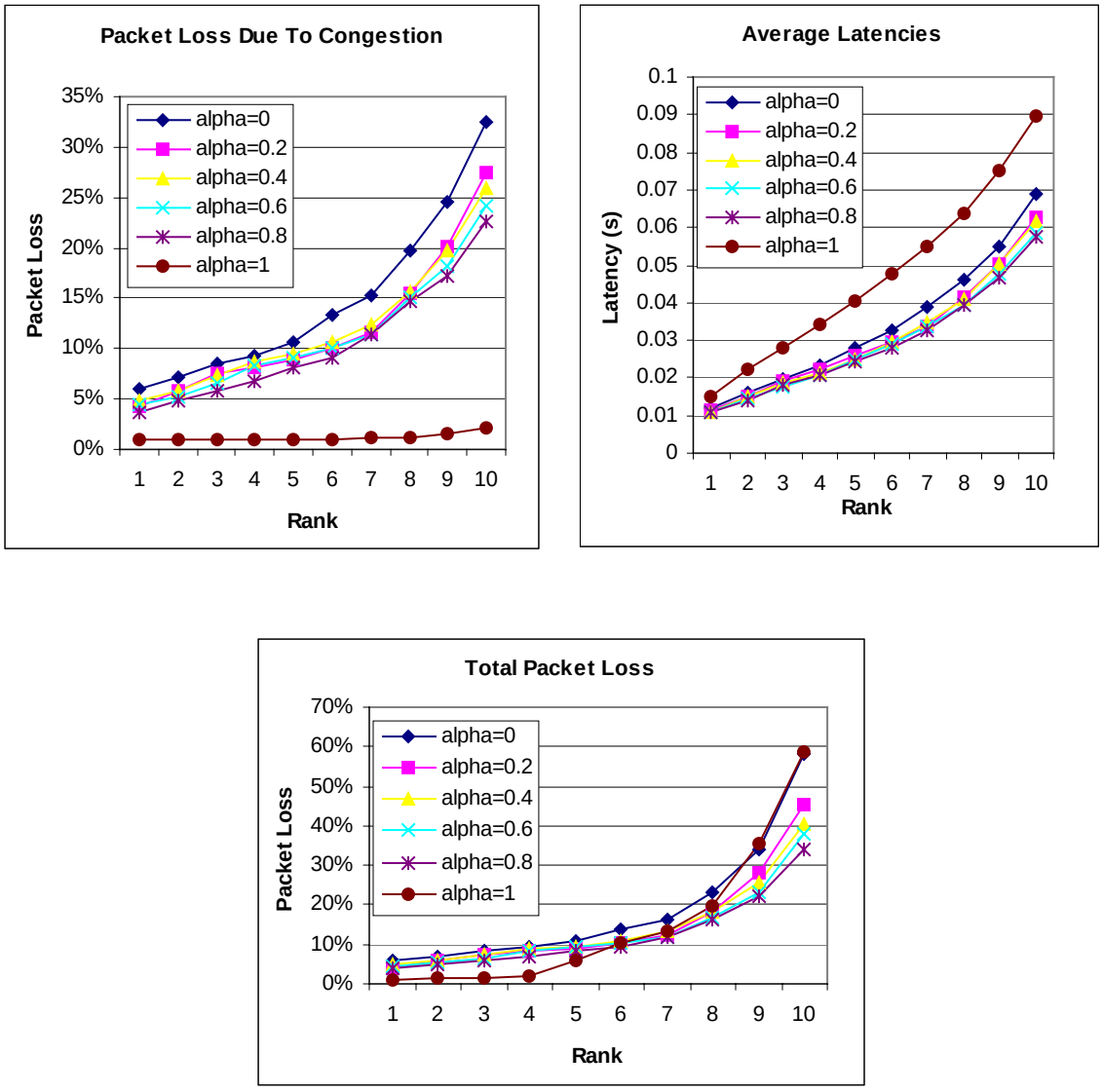


Figure 33: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depict the total packet loss before randomizing the selection of configurations having same highest score.

Figure 34 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss before and after randomizing the selection of configurations having same highest score. Recall that before randomizing the selection of trees with same highest score, large number of packets are lost due to congestion in the network since the algorithm is selecting trees with less available bandwidth on the links as described in 3.2. The total packet loss is lowest for $\alpha = 0.8$ and is nearly 34 %. For the modified score function, $\alpha = 0.4$ has the lowest packet loss of 11%. Thus the packet loss is decreased by 23 % for the worst rank by randomizing the selection of trees having same score.

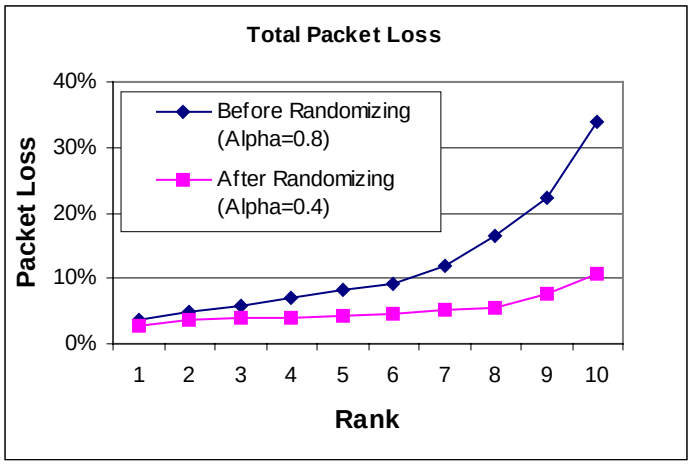


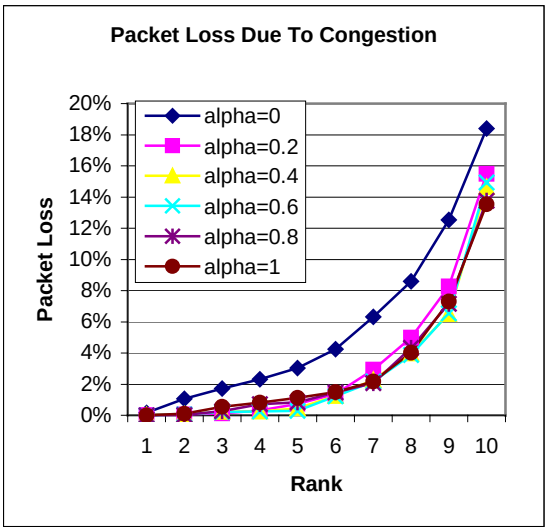
Figure 34: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss before and after randomizing the configurations having same score.

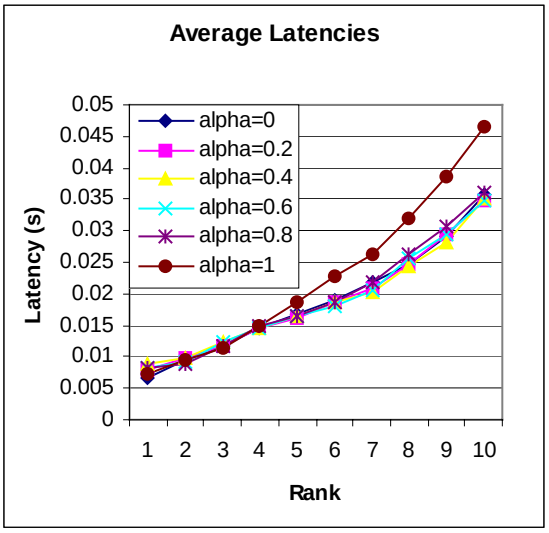
6.1.3 Information to calculate available bandwidth in a tunnel

In order to study the effect of how much information is used to calculate available bandwidth in a tunnel, the modified algorithm (using the modified link level information 'bitrate') with all there enhancements is compared to the algorithm with the modified link level information switched off (using the original link level information 'video + bitrate').

Single Source Statewide Network

Next the simulations are run using Waxman statewide topology with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. Figure 35 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network using the original link level information (video + bitrate). These results are then compared with the simulation results for single video source sending an mpeg stream across a diffserv statewide network using the modified information (bitrate) that are displayed in Figure 16.





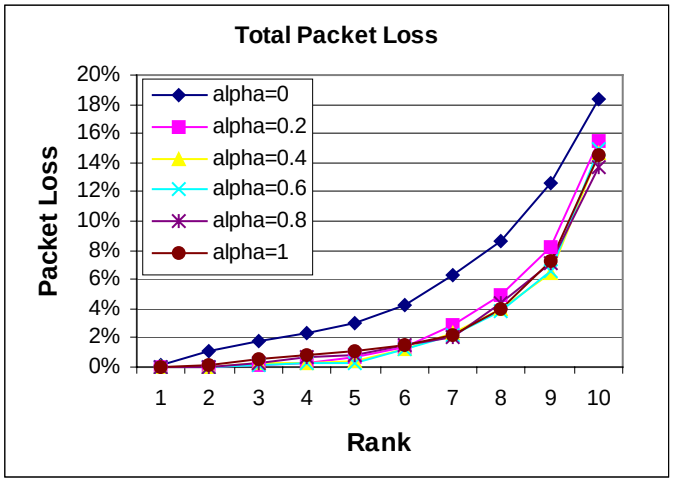


Figure 35: Single video source sending an mpeg stream across a diffserv statewide network. The results depict the total packet loss using original link level information (video + bitrate).

Figure 36 displays the simulation results for single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss using original vs. modified link level information. For the scenario where we use the link level information while calculating the average available bandwidth for a tree configuration, α =0.6 has the lowest packet loss of 13% for the worst rank. For the

scenario where we neglect the link level information while calculating the average available bandwidth for a tree configuration, the packet loss is lowest for $\alpha = 0.6$ and is nearly 14.5% for the worst rank. Recall from 3.3 that for the scenario where we do not use the link level information while calculating the total packet loss, the algorithm might not be able to differentiate between trees on the basis of available bandwidth to the link level and so might end up selecting trees with less available bandwidth that would then result in a slightly higher packet loss. Thus the packet loss can be reduced by 1.5 % if we store the link level information.

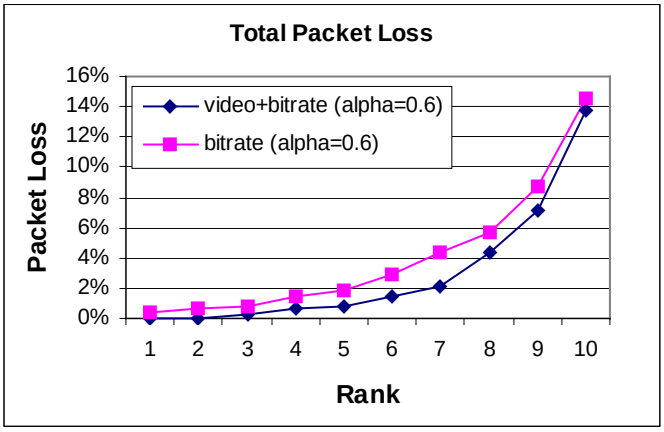


Figure 36: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss using original vs. modified link level information.

Multiple Sources Statewide Network

Next the simulations are run using Waxman statewide topology with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 37 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network using the original link level information (video + bitrate). These results are then compared with the

simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network using the modified information (bitrate) that are displayed in Figure 19.

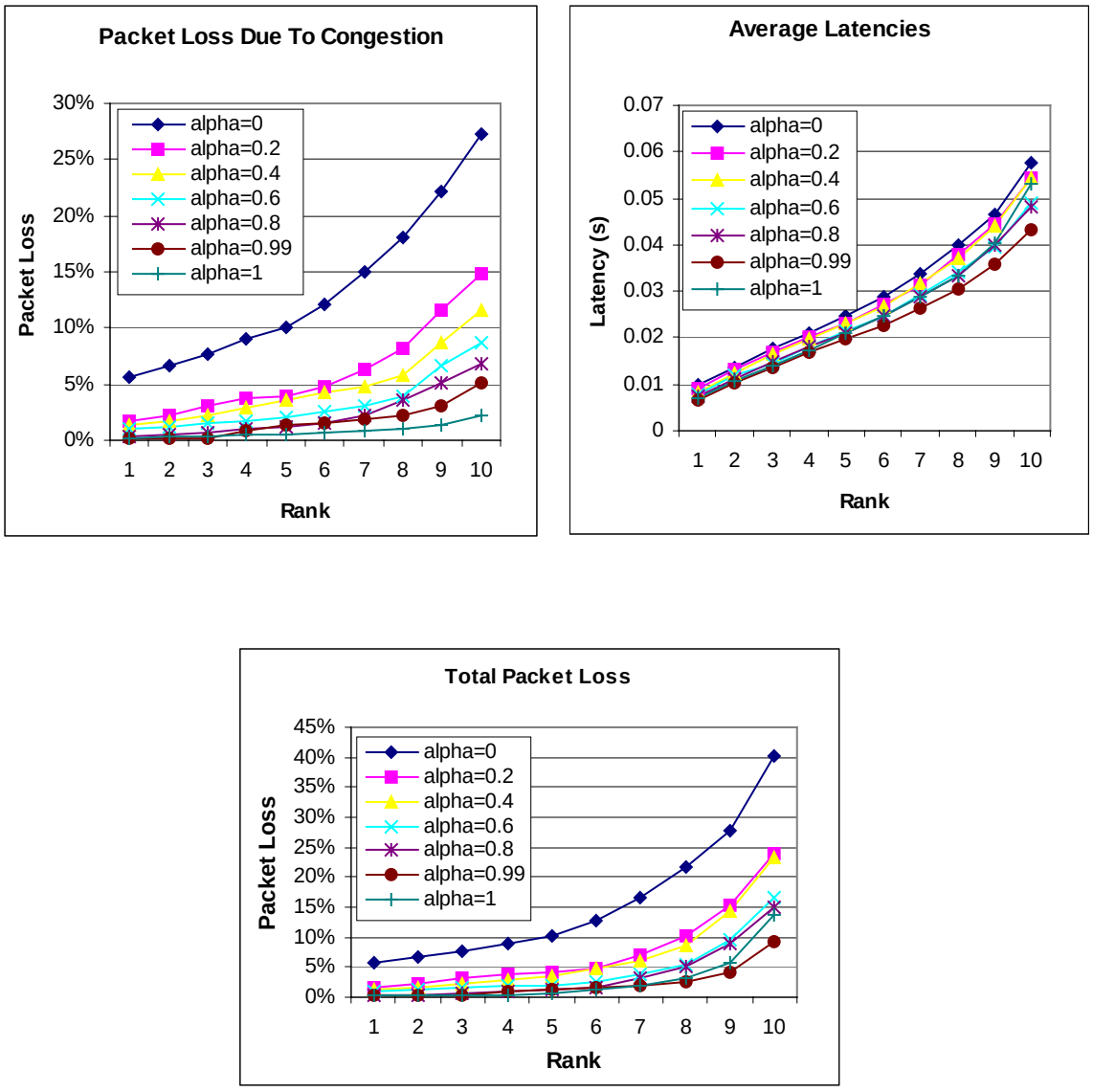


Figure 37: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depict the total packet loss using original link level information (video + bitrate).

Figure 38 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss using original vs. modified link level information. For the scenario where we use the link level information while calculating the average available bandwidth for a tree configuration, α =0.99 has the lowest packet loss of 9% for the worst rank. For the scenario where we do not use the link level information while calculating the total packet loss is lowest for α = 0.99 and is nearly 16% for the worst rank. Recall from 3.3 that for the scenario where we do not use the link level information while calculating the total packet loss, the algorithm might not be able to differentiate between trees on the basis of available bandwidth to the link level and so might end up selecting trees with less available bandwidth that would then result in a slightly higher packet loss. Thus the packet loss can be reduced by 7 % if we store the link level information.

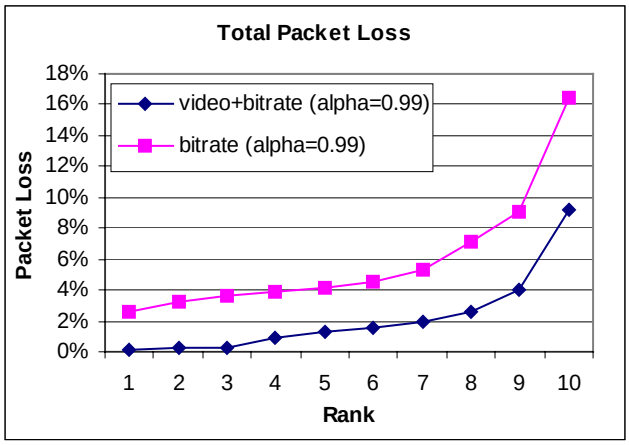


Figure 38: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted compare the total packet loss using the original vs. modified link level information.

Single Source Nationwide Network

Next the simulations are run using Waxman nationwide topology with single source involved in a videoconference. 11 nodes are involved out of which 1 is sending data to the other 10 destinations. Figure 39 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network using the original link level information (video + bitrate). These results are then compared with the simulation results for single video source sending an mpeg stream across a diffserv nationwide network using the modified information (bitrate) that are displayed in Figure 22.

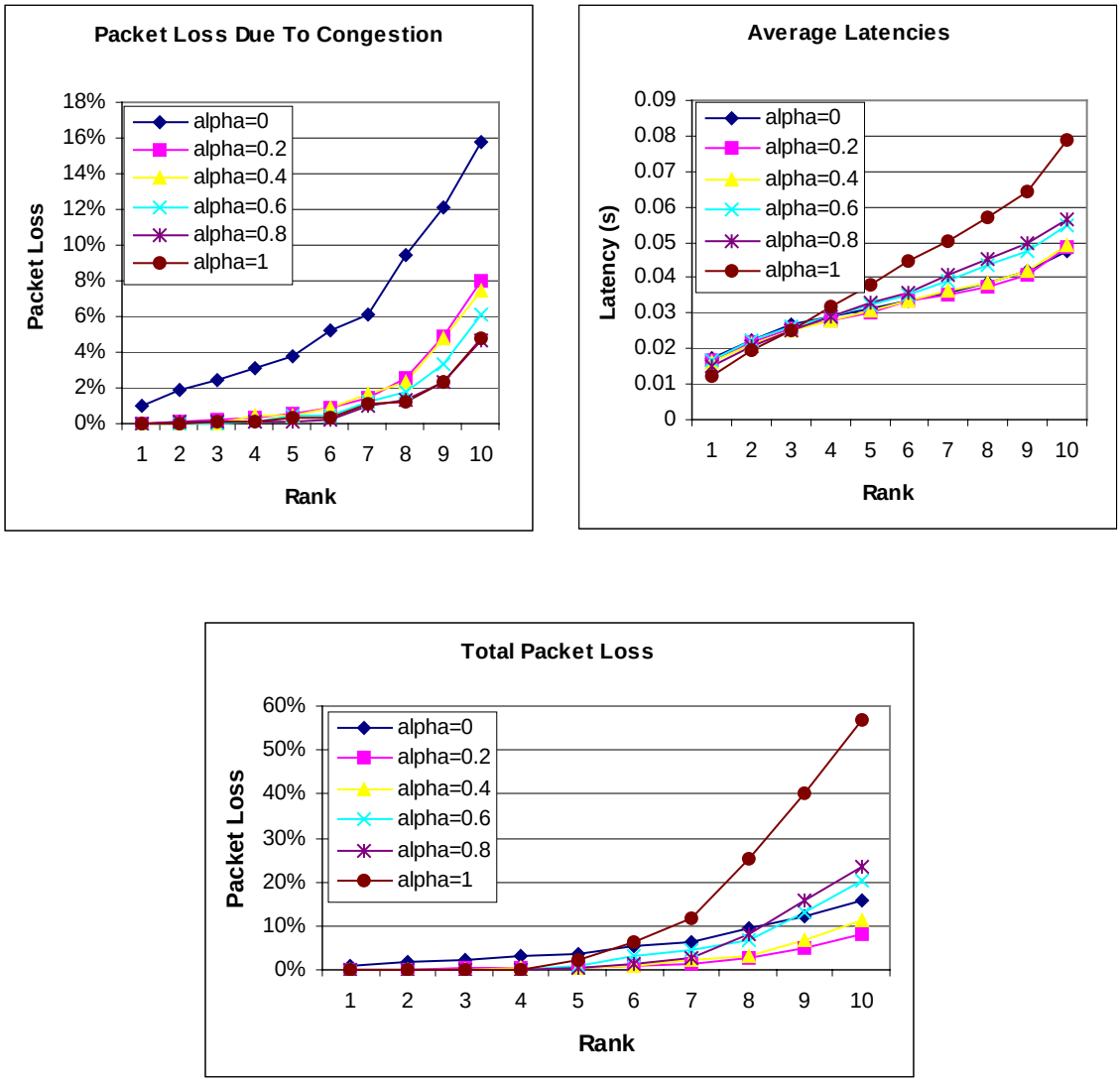


Figure 39: Single video source sending an mpeg stream across a diffserv nationwide network. The results depict the total packet loss using original link level information (Video + Bitrate).

Figure 40 displays the simulation results for single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss using original vs. modified link level information. For the scenario where we use the link level information while calculating the average available bandwidth for a tree

configuration, α =0.2 has the lowest packet loss of 8% for the worst rank. For the scenario where we do not use the link level information while calculating, the total packet loss is lowest for α = 0.4, where the trees and is nearly 10.5% for the worst rank. Recall from 3.3 that for the scenario where we do not use the link level information while calculating the total packet loss, the algorithm might not be able to differentiate between trees on the basis of available bandwidth to the link level and so might end up selecting trees with less available bandwidth that would then result in a slightly higher packet loss. Thus the packet loss can be reduced by 2.5 % if we store the link level information.

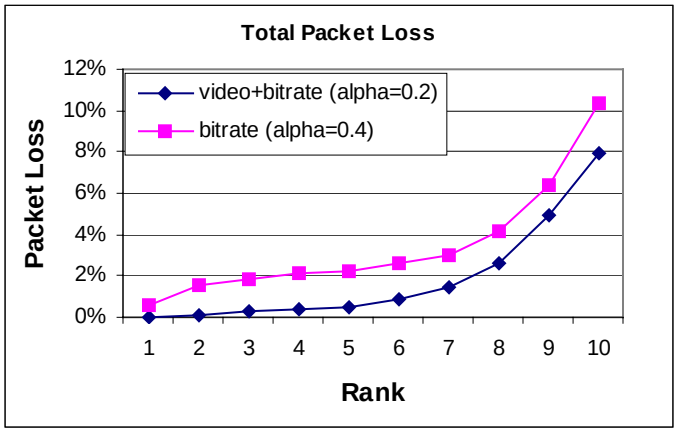


Figure 40: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss using original vs. modified link level information.

Multiple Sources Nationwide Network

Next the simulations are run using Waxman statewide topology with multiple sources involved in a videoconference. 11 nodes are involved out of which 4 are sending data to the other 10 destinations. Figure 41 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network using the original link level information (video + bitrate). These results are compared with the

simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network using the modified information (bitrate) that are displayed in Figure 25.

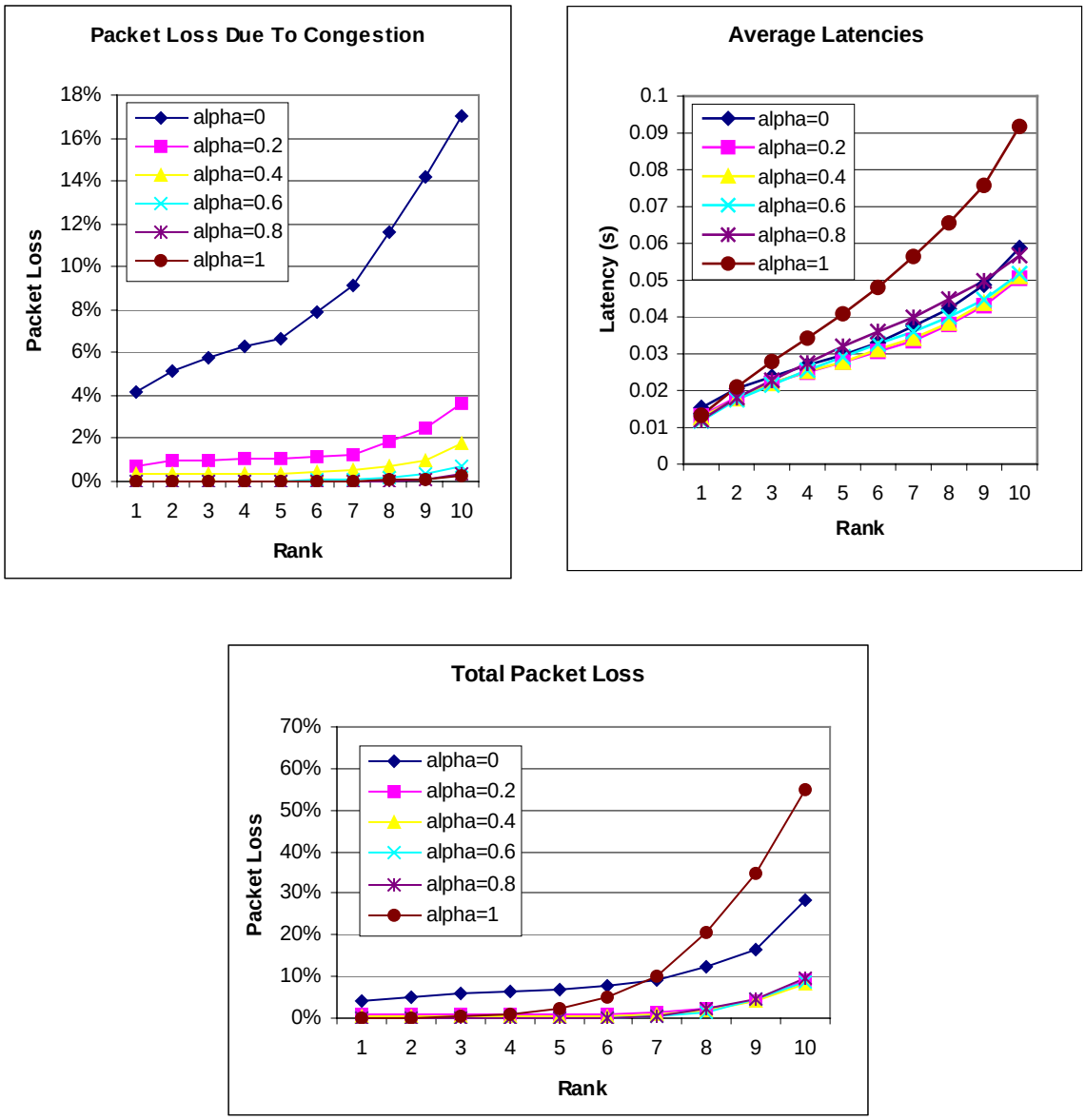


Figure 41: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depict the total packet loss using original link level information (Video + Bitrate).

Figure 42 displays the simulation results for multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss using original vs. modified link level information. For the scenario where we use the link level information while calculating the average available bandwidth for a tree configuration, α =0.4 has the lowest packet loss of 8% for the worst rank. For the scenario where we do not use the link level information while calculating the total packet loss is lowest for α = 0.4, where the trees and is nearly 11% for the worst rank. Recall from 3.3 that for the scenario where we do not use the link level information while calculating the total packet loss, the algorithm might not be able to differentiate between trees on the basis of available bandwidth to the link level and so might end up selecting trees with less available bandwidth that would then result in a slightly higher packet loss. Thus the packet loss can be reduced by 3 % if we store the link level information.

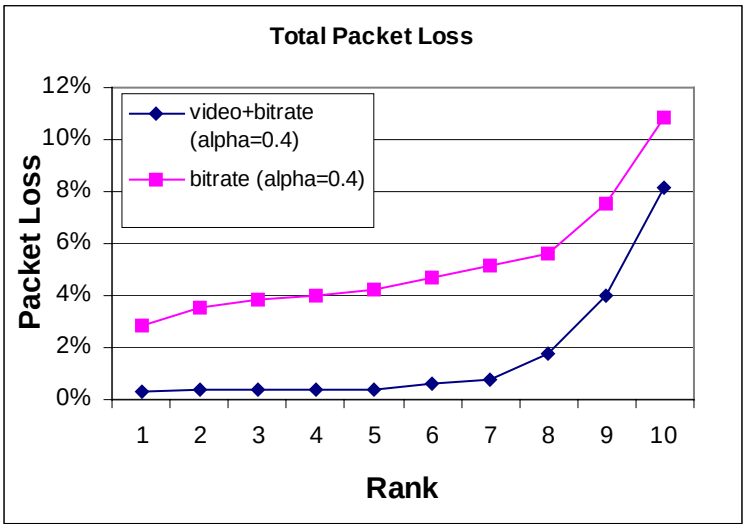


Figure 42: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the total packet loss using original vs. modified link level information.

6.1.4 Summary of Results

Table 5 displays the overall performance in terms of packet loss for different networks. The numbers indicate the percentage of increase (↑) or decrease (↓) in packet loss for after each modification.

Modification	Single Source Statewide	Multiple Source Statewide	Single Source Nationwide	Multiple Source Nationwide
Modifying the calculation for Bandwidth Scalar	3% ↓	24% ↓	5% ↓	18% ↓
Randomizing selection of configurations having same highest score	1.5% ↓	4.5% ↓	6% ↓	23% ↓
Information to calculate available bandwidth in tunnels	1.5% ↑	7% ↑	2.5% ↑	3% ↑

Table 5: Overall performance in terms of packet loss for different networks.

From the above table it can be observed that the first two modifications provide a good increase in performance for multiple source videoconferences. Considering the nature of videoconferences it is more likely that they are going to be multiple source conferences. So both the modifications are equally important to improve the video quality. The third modification slightly decreases the performance of the video in terms of video quality but by neglecting link level information while calculating available bandwidth puts less burden on edge routers and hence reduces the complexity and time for calculating the best-multicast tree. So it is a trade off between improving the quality of the video versus reducing the time and complexity of the algorithm and is a decision left to the ISP.

6.2 Summarizing the best choice of alpha for enhanced base algorithm

Following modifications are done to enhance the base algorithm –

1. Modifying the calculation for Bandwidth Scalar.
2. Randomizing selection of configurations having same highest score.
3. Information to calculate available bandwidth in tunnels.

After the enhancement of the base overlay algorithm to create efficient multicast trees, simulations were run over three different diffserv domains: Campuswide, Statewide and Nationwide for both single and multiple sources scenario to find out the best α. The results for campus wide and statewide networks are identical and hence only the results for state wide network are documented along with the nation wide network. In order to find out the best choice of α, simulations were run for different values of α from $\alpha = 0$ to $\alpha = 1$ in steps of 0.2.

Single Source Statewide Network

The packet loss due to congestion was maximum for $\alpha=0$ and is the least for $\alpha = 1$. The average latencies increased from $\alpha = 0$ to $\alpha = 1$. The best tree was created for $\alpha = 0.6$ with a maximum packet loss of 14.5% as shown in the Total Packet Loss Graph (Figure 16). The variation in total packet loss for $\alpha=0.2$ to $\alpha=1$ was not too much for the single source involved in a videoconference in a statewide network. Hence any value between 0.2 to 1 can be selected as α without affecting the quality of the video.

Multiple Sources Statewide Network

The packet loss due to congestion was maximum for α=0 and is the least for α = 1. The average latency was maximum for α =0. This was mainly because the packet loss for α = 0 was high indicating that there was heavy congestion. Hence the queuing delays were high which resulted in higher average latency for α = 0. The best tree was created for α = 0.99 with a maximum packet loss of 16% as shown in the Total Packet Loss Graph (Figure 19). The variation in total packet loss for different values of α was about 24 % for the multiple sources involved in a videoconference in a statewide network.

Single Source Nationwide network

The packet loss due to congestion was maximum for α=0 and is the least for α = 1. The average latencies increased from α =0 to α =1. The best tree was created for α = 0.4 with a maximum packet loss of 10% as shown in the Total Packet Loss Graph (Figure 22). The variation in total packet loss for different values of α was about 37% for the single source involved in a videoconference in a nationwide network.

Multiple Sources Nationwide Network

The packet loss due to congestion was maximum for α=0 and is the least for α = 1. The average latency was maximum for α =1. The average latency for α = 0 was higher than that for α = 0.2,0.4,0.6 indicating that there was heavy congestion. Hence the queuing delays were high which resulted in higher average latency for α = 0. The best

tree was created for $\alpha = 0.4$ with a maximum packet loss of 11% as shown in the Total Packet Loss Graph (Figure 25). The variation in total packet loss for different values of α was about 48 % for the multiple sources involved in a videoconference in a nationwide network.

Robustness of Algorithm with respect to alpha

It was observed that for a single source statewide videoconference the total packet loss was similar for all values of alpha except alpha = 0. This is mainly because for a statewide network average latency for the worst rank node is less than 45 ms. Hence there is not much packet loss due to delay. So the only packet loss that occurs in the network is due to congestion. The packet loss due to congestion is highest for alpha = 0. The packet loss decreases as alpha is increased from 0 to 1. Hence for a statewide network any value of alpha from 0.1 to 1 works fine. For multiple sources statewide videoconference there is more congestion in the network as multiple sources are sending video packets. Hence packet loss due to congestion decreases with increase in alpha from 0 to 1. At the same time packet loss due to delay is minimum for alpha = 0 to 0.99 and there is some packet loss when trees are selected only on the basis of available bandwidth (alpha =1). Hence the best alpha for this scenario is 0.99. So overall for statewide network alpha = 0.99 can be selected for constructing multicast trees for both single and multiple source videoconference.

For the nation wide network, the packet loss due to congestion decreases from alpha = 0 to alpha =1. At the same time the average latencies for worst rank nodes is more than 50 ms for alpha higher than 0.5 and hence there is some packet loss due to delay. Hence for nationwide network the total packet loss is lowest for alpha = 0.4. So

overall for nationwide network alpha = 0.4 can be selected for constructing multicast trees for both single and multiple source videoconference.

The best choice of alpha for enhanced base algorithm can be tabulated as shown in Table 5. The top number in each cell is the best choice of α for respective scenarios. The bottom numbers enclosed in brackets indicate the range of α for which the worst-case packet loss changes by no more than 5% of the total variation for the respective scenario i.e. indicates how robust the algorithm is with respect to alpha.

Scenario	Statewide	Nationwide
Single Source Best Effort class	0.6 (0 to 1)	0.4 (0 to 0.9)
Multiple Sources Best Effort Class	0.99 (0.99)	0.4 (0 to 0.9)

Table 6: Choice of best alpha for different networks.

Looking at the bottom numbers for each column in the above table it can be concluded that the algorithm is quite robust with respect to alpha. Choosing alpha to be around 0.9 is a good compromise for all traffic situations while losing only some performance in the multiple source/statewide case. For the multiple source/statewide case choosing alpha = 0.9 increases the packet loss by 8% than that for the best alpha value of 0.99.

6.3 Comparing with other DiffServ Multicast Technologies

The base overlay algorithm is enhanced with the following modifications -

1. Modifying the calculation for Bandwidth Scalar.
2. Randomizing selection of configurations having same highest score.
3. Information to calculate available bandwidth in tunnels.

After the base overlay algorithm is enhanced and the best alpha is identified for each network, simulations are run to compare the enhanced base overlay algorithm with other existing diffserv multicast technologies: DSMCast, EBM and Native Multicast.

DSMCast

The DSMCast described in [StM01] and [Str01] uses DVMRP to for the multicast trees. The multicast tree generated is similar to native multicast. All the routers are multicast capable. The multicast routing information is sent along with the data packets.

EBM

The EBM as described in [StB03] generates the multicast tree in two different ways.

1. EBM Simple – Each joining node appends itself to the nearest node in the multicast tree.
2. EBM Cluster – The multicast tree is formed using the Edge cluster algorithm.

Both DSMCast and EBM do not take into account the link bandwidths and latencies while calculating the multicast trees.

Statewide network single source

The simulations were run over statewide network for single source involved in the videoconference. There are eleven participants with one sender and ten receivers.

The tree formed with best value of α for the Enhanced Base Overlay Algorithm, was compared with DSMCast/ Native Multicast and the Simple and Edge Clustered EBM. From the graph for total packet loss as shown in Figure 43 it was seen that Simple EBM suffered the highest packet loss of 29%, which was 15% higher than our overlay algorithm. Next was EBM Cluster with a packet loss of 27%, which was 13% higher than our algorithm. DSMCast/Native multicast had the highest packet loss of 19%, which was 5% higher.

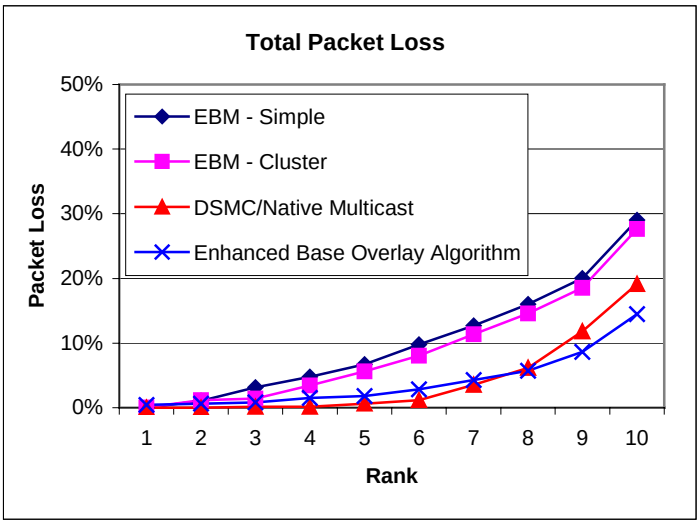


Figure 43: Single video source sending an mpeg stream across a diffserv statewide network. The results depicted for comparing total packet loss for different diffserv multicast technologies.

Statewide network Multiple Sources

Next simulations were run over statewide network for multiple sources involved in the videoconference. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers.

The tree formed with best value of α for the Enhanced Base Overlay Algorithm, was compared with DSMCast/ Native Multicast and the Simple and Edge Clustered EBM. From the graph for total packet loss as shown in Figure 44 it was seen that Simple EBM suffered the highest packet loss of 50%, which was 34% higher than our overlay algorithm. Next was EBM Cluster with a packet loss of 22%, which was 6% higher than our algorithm. DSMCast/Native multicast had the highest packet loss of 6%, which was 10% lower than our overlay algorithm.

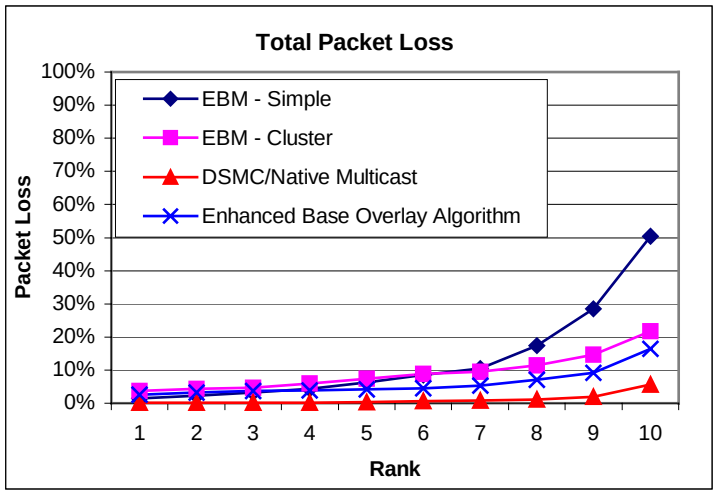


Figure 44: Multiple video sources sending an mpeg stream across a diffserv statewide network. The results depicted for comparing total packet loss for different diffserv multicast technologies.

Nationwide network single source

The simulations were run over nationwide network for single source involved in the videoconference. There are eleven participants with one sender and ten receivers.

The tree formed with best value of α for the overlay algorithm, was compared with DSMCast/ Native Multicast and the Simple and Edge Clustered EBM. From the graph for total packet loss as shown in Figure 45 it was seen that Simple EBM suffered the highest packet loss of 30%, which was 20% higher than our overlay algorithm. Next was EBM Cluster with a packet loss of 19%, which was 9% higher than our algorithm. DSMCast/Native multicast had the highest packet loss of 11%, which was 1% higher.

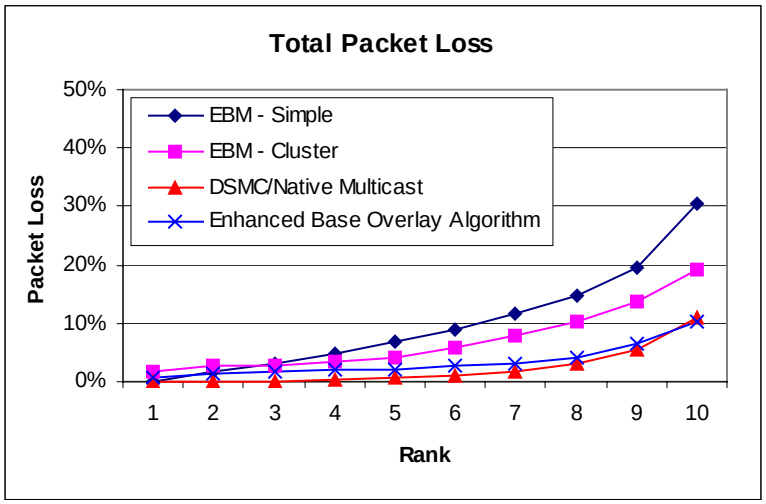


Figure 45: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted for comparing total packet loss for different diffserv multicast technologies.

Nationwide network Multiple Sources

Next simulations were run over nationwide network for multiple sources involved in the videoconference. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers.

The tree formed with best value of α for the overlay algorithm, was compared with DSMCast/ Native Multicast and the Simple and Edge Clustered EBM. From the graph for total packet loss as shown in Figure 46 it was seen that Simple EBM suffered the highest packet loss of 68%, which was 58% higher than our overlay algorithm. Next was EBM Cluster with a packet loss of 24%, which was 14% higher than our algorithm. DSMCast/Native multicast had the highest packet loss of only 4%, which was 6% lower than our overlay algorithm.

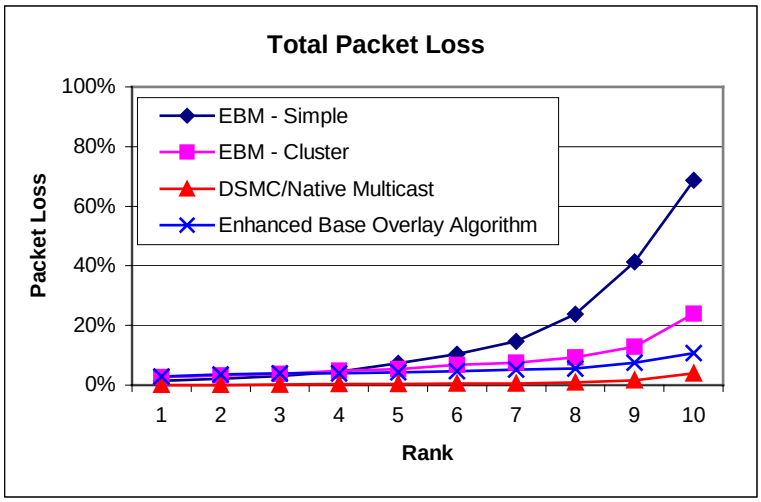


Figure 46: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted for comparing total packet loss for different diffserv multicast technologies.

6.4 DiffServ Extension of the base overlay Algorithm

Till now all the simulations run were for receivers getting same level of QoS i.e. best effort service. In this stage simulations are run after the enhanced base overlay algorithm is modified to support different receivers demanding different classes of QoS.

The simulations are run for a videoconference involving eleven participants. Two different scenarios are considered – single source and multiple sources and the simulations are run over the nationwide diffserv network. The destinations are grouped into two different QoS classes – EF (Expedited Forwarding) and AF (Assured forwarding).

First the simulations are run to show the performance improvement of implementing the Call admission control in the score function and to find out the best value of alpha. Next the simulations are run to decide whether to use average or minimum bandwidth in the score function. Finally simulations are run to compare the performance of diffserv overlay multicast with existing diffserv multicast technologies.

6.4.1 Implementing CAC in Score Function

Next simulations were run in order to find out the best choice of score function. The simulations were run with and without the CAC in score function over nationwide network for single source involved in the videoconference. There are eleven participants with one sender and ten receivers. The simulations are run for two different scenarios. In the first scenario all the destinations are demanding EF class and in the second scenario all the destinations are demanding AF class. Alpha was incremented for different values from $\alpha = 0$ to $\alpha = 1$ in steps of 0.1.

The CAC acceptance percentage was constant for all the values of alpha when CAC was implemented in the score function and was 97%. However when CAC was not in score function, the CAC acceptance percentage dropped from $\alpha = 0.5$ to $\alpha = 0$ and was 48% for $\alpha = 0$. Likewise the average cost for reserving EF and AF bandwidth dropped from $\alpha = 0.5$ to $\alpha = 0$ when CAC not implemented in the score. The average packet loss for AF class due to congestion was maximum for $\alpha = 0$ and is the least for $\alpha = 1$. The average packet loss for all values of alpha was higher when CAC was not implemented in the score function.

After analyzing the four graphs as displayed in Figure 47, it can be concluded that in order to improve the overall performance of the videoconference, CAC must be implemented in the score function. Although alpha around 0.2 results in lower EF cost and AF packet loss, the results are not highly influenced by the change in alpha. In order to reduce the complexity and running time of the algorithm, alpha can be eliminated from the score function.

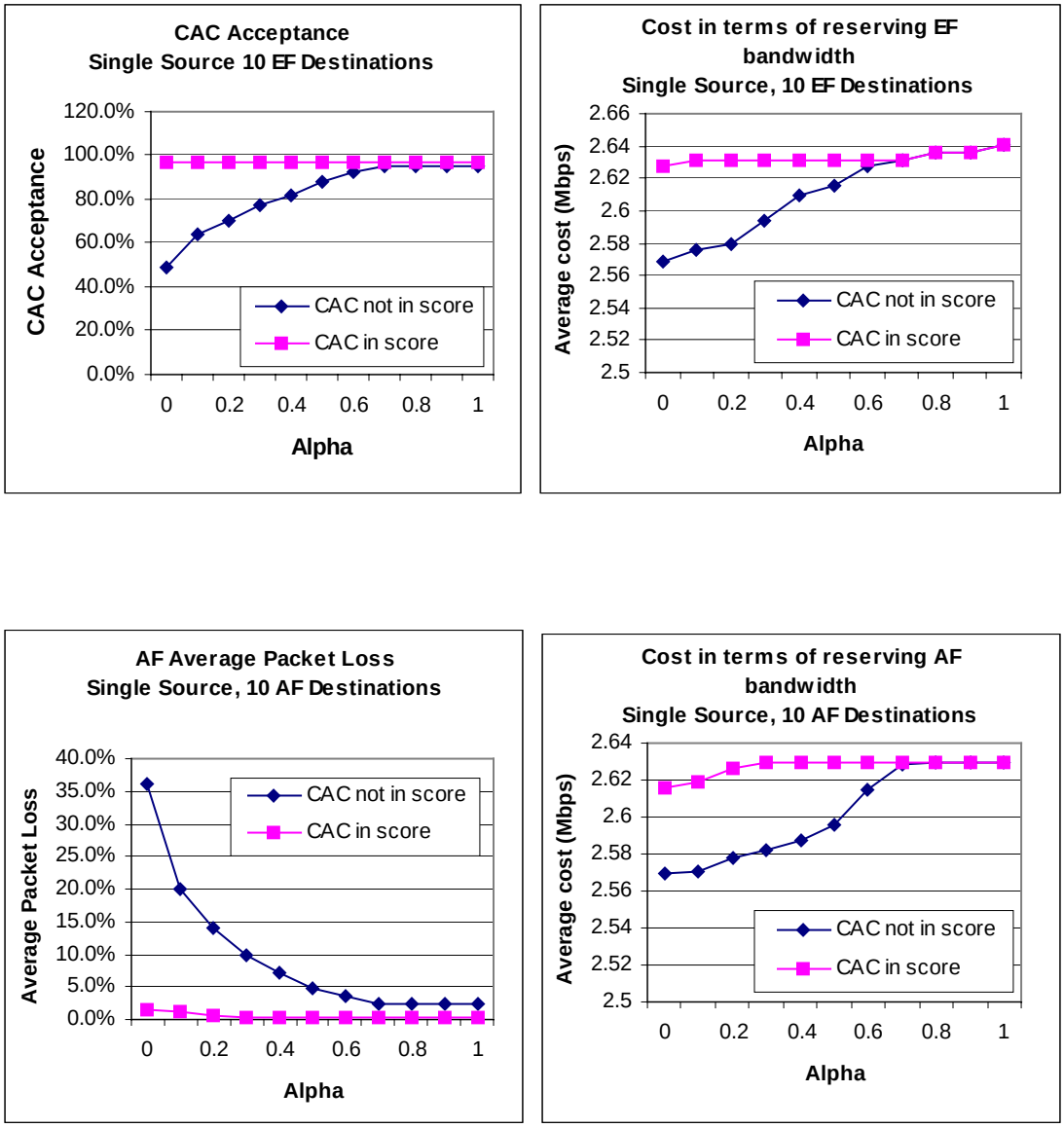


Figure 47: Single video source sending an mpeg stream across a diffserv nationwide network. The results depicted compare the performance metrics with and without implementing CAC in the score function.

6.4.2 Minimum vs Average Bandwidth in Score Function

Next simulations were run in order to find out the best choice of bandwidth used for calculating CAC in the score function. The simulations were run using average and minimum bandwidth for calculating the CAC in the score, over nationwide network for multiple sources involved in the videoconference. The minimum bandwidth of a tree configuration was calculated as the minimum of minimum available bandwidth on each tunnel in the multicast tree. The average bandwidth was calculated as the average of minimum available bandwidth on each tunnel in the multicast tree. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers. The simulations are run for two different scenarios. In the first scenario all the destinations are demanding EF class and in the second scenario all the destinations are demanding AF class. Alpha was eliminated from the score function.

The CAC acceptance percentage was 78% using minimum bandwidth while implementing CAC in score and reduced to 64 % when using average bandwidth. The cost for reserving EF and AF bandwidth was identical in both cases. However when CAC was implemented using minimum bandwidth, AF packet loss was 4.4% as compared to 5.1% for average bandwidth.

After analyzing the four graphs as displayed in Figure 48, it can be concluded that in order to improve the overall performance of the videoconference, CAC must be implemented in score function using minimum bandwidth.

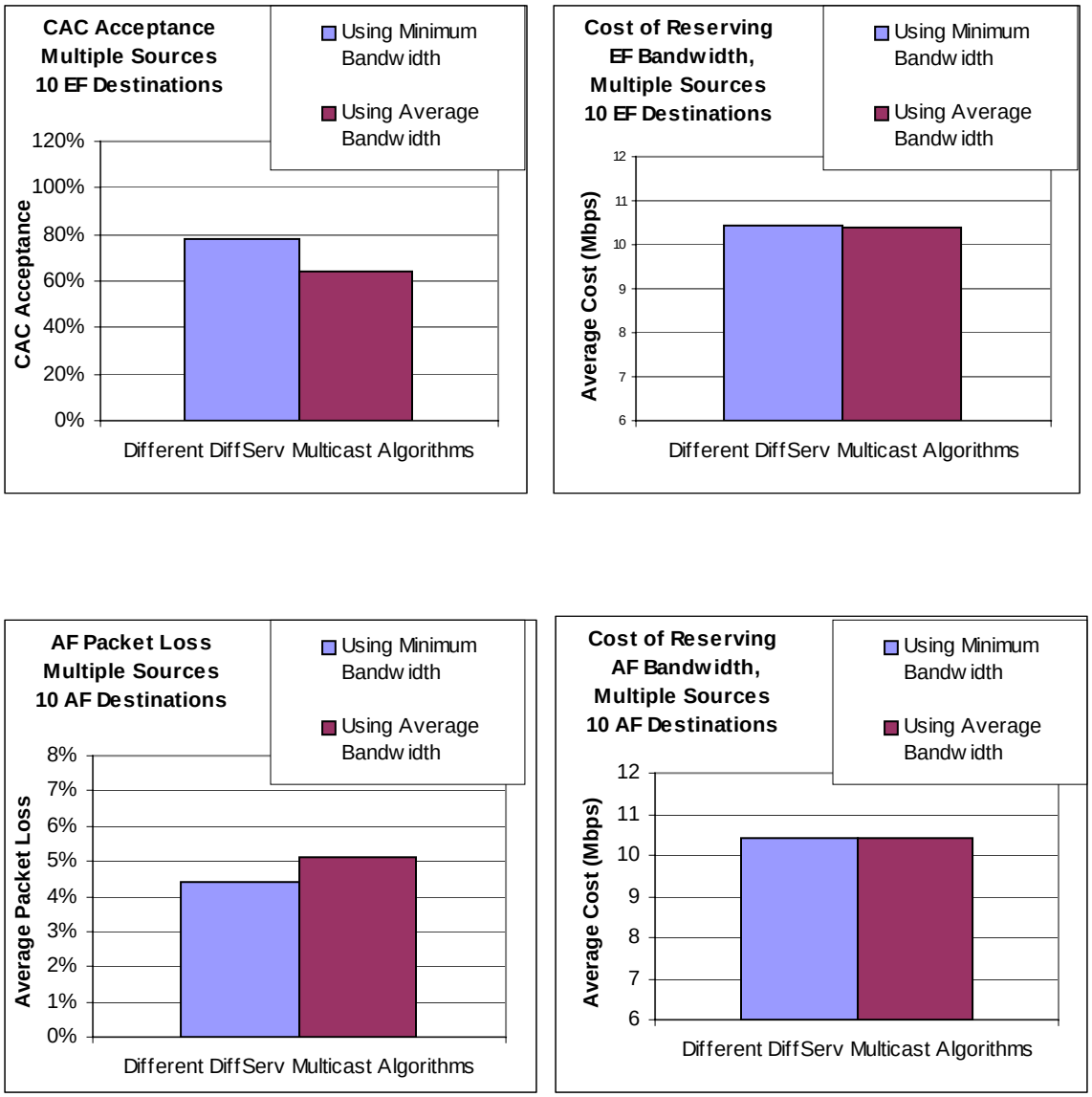


Figure 48: Multiple video sources sending an mpeg stream across a diffserv nationwide network. The results depicted compare the performance metrics using average and minimum bandwidth for calculating the CAC in score function.

6.5 Comparing with other DiffServ Multicast Technologies

The DiffServ Overlay Multicast is then compared with the existing diffserv multicast technologies: DSMCast, EBM and native multicast. Cost for reserving bandwidth in the network depends on the total number of stress on all the tunnels in the multicast tree and also on the extra header information sent along with the packet. The header information varies with different diffserv multicast technologies. The cost is calculated using the following –

$$Cost = \sum_{i=1}^{n} Stress_i * S$$

$$Stress_i = Number\ of\ hops\ in\ a\ tunnel_i * Bandwidth\ of\ Video$$

Type of DiffServ Multicast	Value of S
Native Multicast	S = 1
DSMCast	$S = (P_s + T_{eh}) / P_s$
EBM – Simple/Cluster	$S = (P_s + T) / P_s$
DiffServ Overlay Multicast	$S = (P_s + T) / P_s$

Table 7 – Scaling Factor for determining cost of reserving bandwidth in the diffserv network for different diffserv multicast technologies.

While calculating the cost the following parameters are used.

1. Bandwidth of video = 1Mbps
2. T_{eh} = Encapsulation header = (2 + k* 4) bytes
3. k = number of non-leaf nodes in the multicast tree
4. T = Tunneling header = 12 bytes
5. P_s = Packet Size = 1500 bytes

6.5.1 Single Source with destinations demanding EF class

Next simulations are run over a nationwide network for single source involved in the videoconference. There are eleven participants with one sender and ten receivers demanding EF class. Figure 49 compares the performance metrics for different DiffServ Multicast technologies. The CAC acceptance percentage was 97% for DiffServ Overlay Multicast. It was 67% for DSMCast/Native Multicast, 42% for EBM –Simple and 30% for EBM – Cluster. The cost for reserving EF bandwidth was 2.64 Mbps per EF destination for the DiffServ Overlay Multicast. It was 2.6 Mbps for DSMCast, 2.44 Mbps for Native Multicast, 2.83 for EBM –Simple and 3.22 Mbps for EBM – Cluster

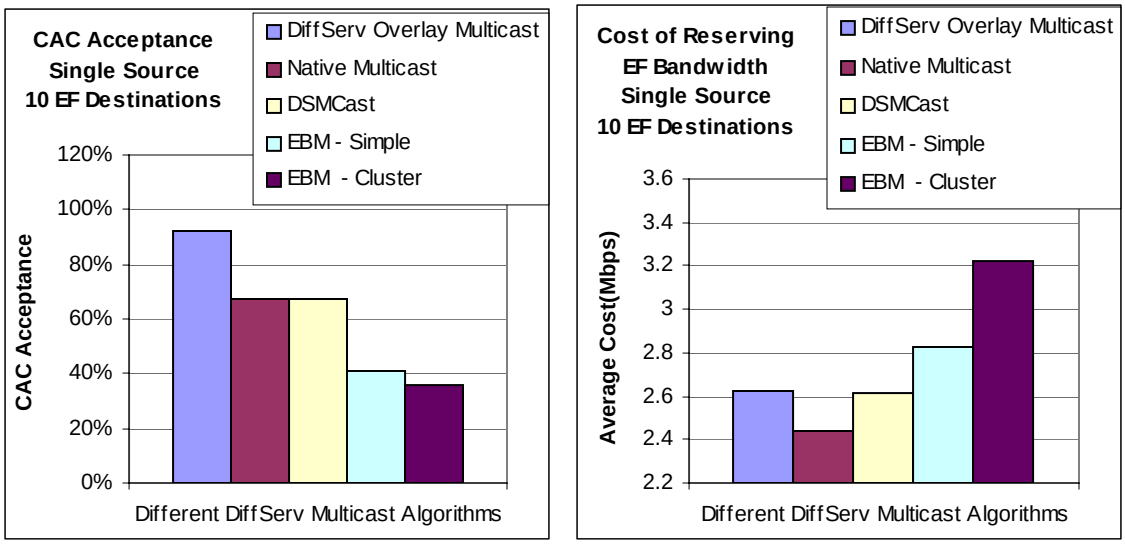


Figure 49: Single video source sending an mpeg stream across a diffserv nationwide network with destinations demanding EF class. The results depicted compare the performance metrics for different DiffServ Multicast technologies.

6.5.2 Single Source with destinations demanding AF class

Next simulations are run over a nationwide network for single source involved in the videoconference. There are eleven participants with one sender and ten receivers demanding AF class. Figure 50 compares the performance metrics for different DiffServ Multicast technologies. The Average packet loss for AF destinations was 1.5% for DiffServ Overlay Multicast. It was 5% for DSMCast/Native Multicast, 10% for EBM – Simple and 15% for EBM – Cluster. The cost for reserving AF bandwidth was 2.63 Mbps per AF destination for DiffServ Overlay Multicast. It was 2.6 Mbps for DSMCast, 2.44 Mbps for Native Multicast, 2.83 for EBM –Simple and 3.22 Mbps for EBM – Cluster.

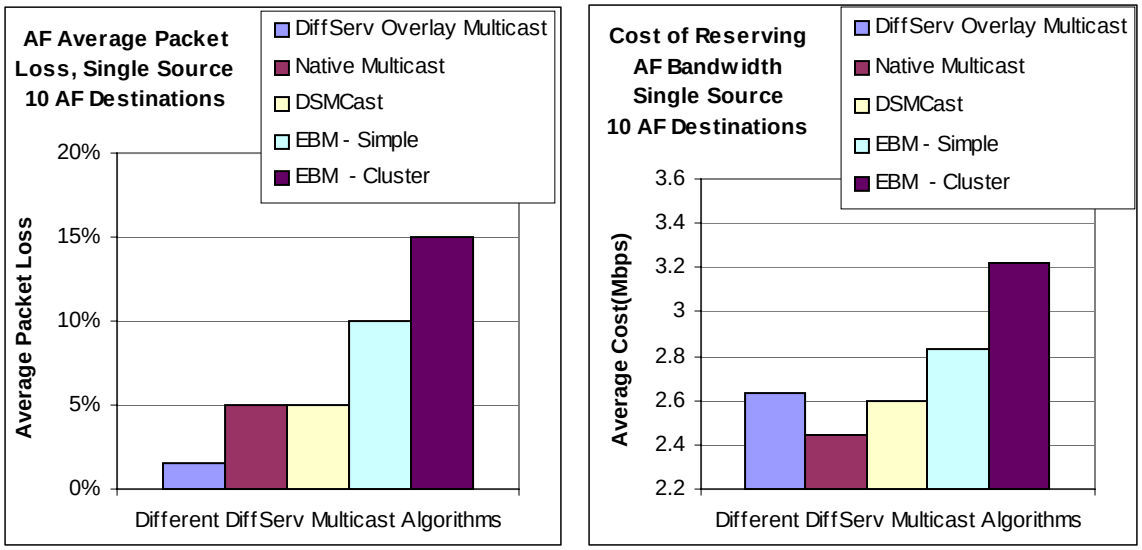


Figure 50: Single video source sending an mpeg stream across a diffserv nationwide network with destinations demanding AF class. The results depicted compare the performance metrics for different DiffServ Multicast technologies.

6.5.3 Single Source with destinations demanding mix

Next simulations are run over a nationwide network for single source involved in the videoconference. In this scenario destinations demand a mix of EF and AF class. The number of destinations demanding EF class is incremented from 0 to 10 in steps of 1. Figure 51 compares the performance metrics for different DiffServ Multicast technologies.

The CAC acceptance percentage was constant for EF = 3 to EF =10 destinations and was 97% for DiffServ Overlay Multicast. However the CAC acceptance percentage for DSMCast/Native Multicast, EBM-Simple and EBM-Cluster decreased as the number of EF destinations increased from 1 to 10. The average cost for reserving EF bandwidth for the overlay algorithm was nearly equal to DSMCast and slightly higher than Native Multicast for all values of EF destinations. The average packet loss due to congestion was equal to 1.5% for DiffServ Overlay Multicast. It was 5% for DSMCast/Native Multicast, 10% for EBM –Simple and 15% for EBM – Cluster. The average cost for reserving EF bandwidth for DiffServ Overlay Multicast was slightly higher than DSMCast for all values of EF destinations. This is mainly due to the difference in the construction of multicast trees for the two algorithms when there is a mix of traffic classes. For DSMCast a new AF node can be added anywhere in the existing multicast tree. However for DiffServ Overlay Multicast a new AF node can be added only according to the rules specified in 4.3.

After analyzing the four graphs displayed in Figure 51, it can be concluded that DiffServ Overlay Multicast provides a better success rate of establishing the videoconference while maintaining a low packet loss in order to improve the overall

performance of the videoconference. Also it has a lower cost for reserving EF and AF bandwidths compared to EBM-Simple and EBM –Cluster.

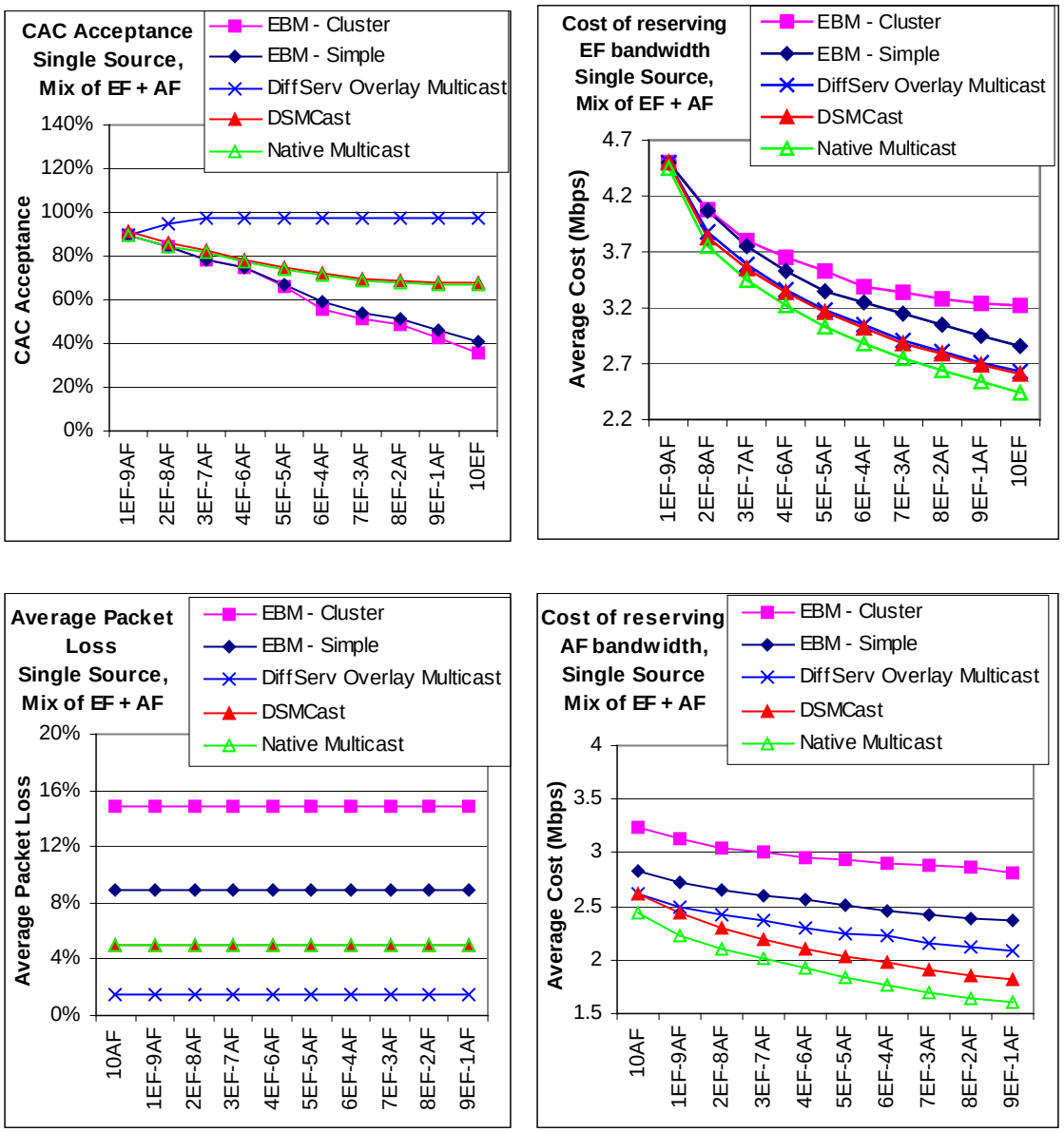


Figure 51: Single video source sending an mpeg stream across a diffserv nationwide network with destinations demanding a mix of EF and AF classes. The results depicted compare the performance metrics for different Multicast technologies.

6.5.4 Multiple Sources with destinations demanding EF class

Next simulations are run over a nationwide network for multiple sources involved in the videoconference. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers demanding EF class. Figure 52 compares the performance metrics for different DiffServ Multicast technologies. The CAC acceptance percentage was 77% for DiffServ Overlay Multicast. It was 67% for DSMCast/Native Multicast, 32.6% for EBM –Simple and 28% for EBM – Cluster. The cost for reserving EF bandwidth was 10.42 Mbps per EF destination for DiffServ Overlay Multicast. It was 10.44 Mbps for DSMCast, 9.76 Mbps for Native Multicast, 11.3 for EBM –Simple and 12.93 Mbps for EBM – Cluster.

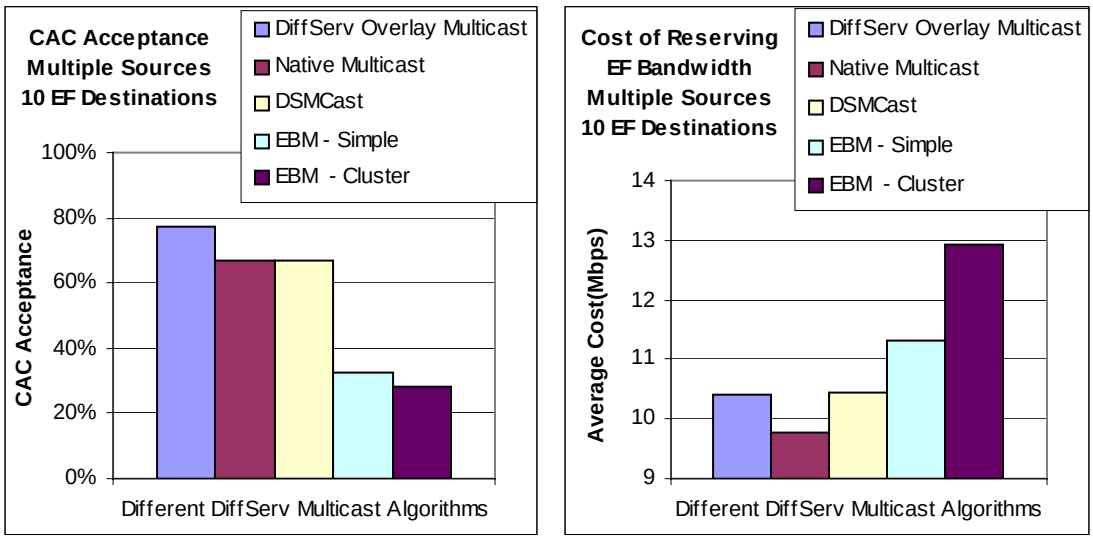


Figure 52: Multiple video sources sending an mpeg stream across a diffserv nationwide network with destinations demanding EF class. The results depicted compare the performance metrics for different DiffServ Multicast technologies.

6.5.5 Multiple Sources with destinations demanding AF class

Next simulations are run over a nationwide network for multiple sources involved in the videoconference. There are eleven participants with four senders. Each sender sends video packets to the other ten receivers. Figure 53 compares the performance metrics for different DiffServ Multicast technologies. The Average packet loss for AF destinations for DiffServ Overlay Multicast was 4.4%. It was 1% for DSMCast/Native Multicast, 20% for EBM –Simple and 22% for EBM – Cluster. The cost for reserving AF bandwidth was 10.42 Mbps per AF destination for DiffServ Overlay Multicast. It was 10.44 Mbps for DSMCast, 9.76 Mbps for Native Multicast, 11.3 for EBM –Simple and 12.93 Mbps for EBM – Cluster.

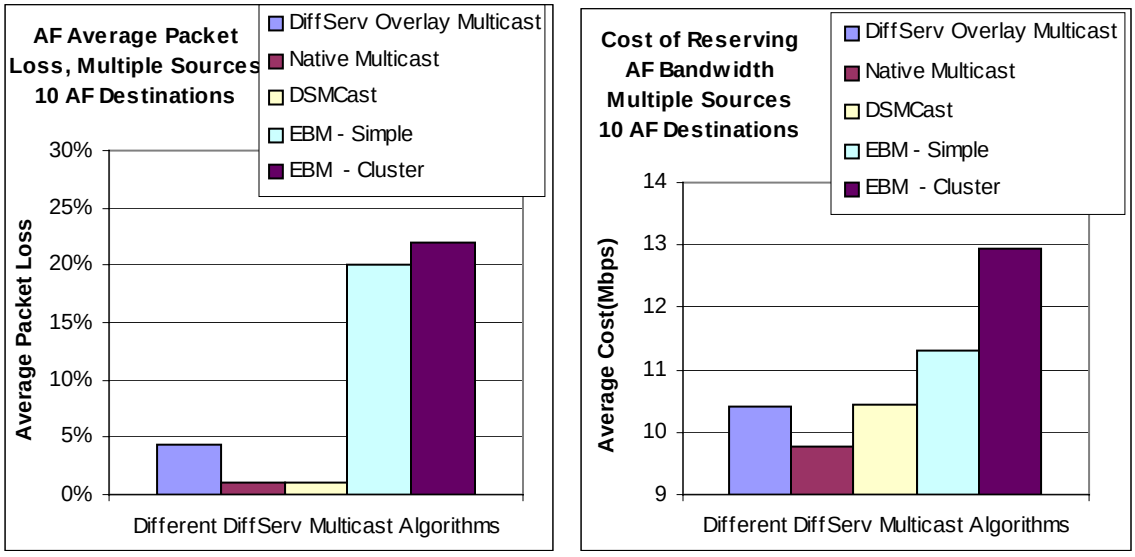


Figure 53: Multiple video sources sending an mpeg stream across a diffserv nationwide network with destinations demanding AF class. The results depicted compare the performance metrics for different DiffServ Multicast technologies.

6.5.6 Multiple Sources with destinations demanding mix

Next simulations are run over a nationwide network for multiple sources involved in the videoconference. In this scenario destinations demand a mix of EF and AF class. The number of destinations demanding EF class is incremented from 0 to 10 in steps of 1. Figure 54 compares the performance metrics for different DiffServ Multicast technologies.

The CAC acceptance percentage for all the algorithms decreases as the number of EF destinations increased from 1 to 10. The CAC acceptance for DiffServ Overlay Multicast is higher compared to other algorithms. The average cost for reserving EF bandwidth for DiffServ Overlay Multicast was nearly equal to DSMCast and slightly higher than Native Multicast for all values of EF destinations. The average packet loss due to congestion was equal to 4.4% for DiffServ Overlay Multicast. It was 1% for DSMCast/Native Multicast, 20% for EBM –Simple and 25% for EBM – Cluster. The average cost for reserving EF bandwidth for the DiffServ Overlay Multicast was slightly higher than DSMCast for all values of EF destinations. This is mainly due to the difference in the construction of multicast trees for the two algorithms when there is a mix of traffic classes. For DSMCast a new AF node can be added anywhere in the existing multicast tree. However for diffserv overlay multicast a new AF node can be added only according to the rules specified in 4.3.

After analyzing the four graphs displayed in Figure 54, it can be concluded that diffserv overlay multicast provides a better success rate of establishing the videoconference while maintaining a low packet loss in order to improve the overall

performance of the videoconference. Also it has a lower cost for reserving EF and AF bandwidths compared to EBM-Simple and EBM –Cluster.

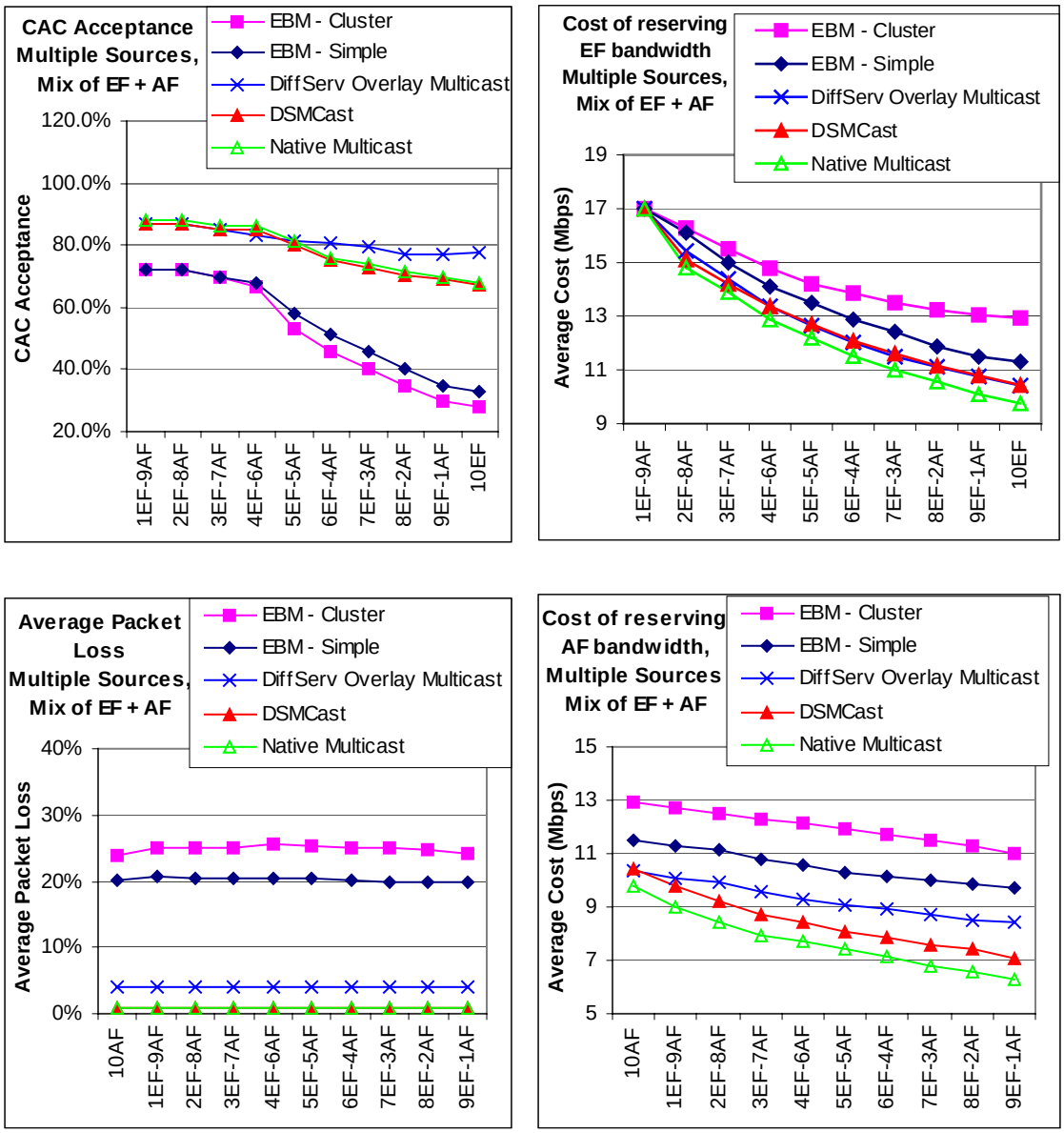


Figure 54: Multiple video sources sending an mpeg stream across a diffserv nationwide network with destinations demanding a mix of EF and AF classes. The results depicted compare the performance metrics for different Multicast technologies.

6.6 Final Summary of simulation results

Simulations were run to determine three different sets of results

1. To study the performance of the base overlay algorithm after enhancements.
2. To determine the value of parameter α to create best overlay multicast tree.
3. To compare the enhanced overlay algorithm (DiffServ Overlay Multicast) with other existing diffserv multicast technologies.

First set of results compared the performance of original vs. the enhanced base overlay algorithm. For the original base algorithm, it was observed that the algorithm was selecting trees with less available bandwidth on the links. Hence large numbers of packets were lost due to congestion in the network. After enhancements the modified base overlay algorithm selects trees with more available bandwidth on the links. Also the modified algorithm randomizes the selection of tree configurations having identical highest scores. Thus the total packet loss is reduced which in turn betters the video quality. From the simulation results it can be observed that the total packet loss for single source statewide videoconference can be reduced by 2 % for the worst rank by modifying the base overlay algorithm. For a multiple source statewide videoconference the total packet loss can be reduced by 21 % for the worst rank by modifying the base overlay algorithm. Similarly for single and multiple sources videoconference in a nationwide network the total packet loss can be reduced by 6% and 38% respectively for the worst rank by modifying the base overlay algorithm.

The second set of results determined the best value of alpha for each type of scenario. It was observed that for a single source statewide videoconference the total packet loss was similar for all values of alpha except alpha = 0. This is mainly because for a statewide network average latency for the worst rank node is less than 45 ms. Hence there is not much packet loss due to delay. So the only packet loss that occurs in the network is due to congestion. The packet loss due to congestion is highest for alpha = 0. The packet loss decreases as alpha is increased from 0 to 1. Hence for a statewide network any value of alpha from 0.1 to 1 works fine. For multiple sources statewide videoconference there is more congestion in the network as multiple sources are sending video packets. Hence packet loss due to congestion decreases with increase in alpha from 0 to 1. At the same time packet loss due to delay is minimum for alpha = 0 to 0.99 and there is some packet loss when trees are selected only on the basis of available bandwidth (alpha =1). Hence the best alpha for this scenario is 0.99. So overall for statewide network alpha = 0.99 can be selected for constructing multicast trees for both single and multiple source videoconference.

For the nation wide network, the packet loss due to congestion decreases from alpha = 0 to alpha =1. At the same time the average latencies for worst rank nodes is more than 50 ms for alpha higher than 0.5 and hence there is some packet loss due to delay. Hence for nationwide network the total packet loss is lowest for alpha = 0.4. So overall for nationwide network alpha = 0.4 can be selected for constructing multicast trees for both single and multiple source videoconference.

The third set of results compared the performance of DiffServ Overlay multicast with existing diffserv multicast technologies. For single source videoconference where all the nodes demand best effort service, DiffServ Overlay Multicast had the lowest total packet loss compared to DSMCast, Native Multicast and both the versions of EBM. When multiple sources are involved in a videoconference, both DSMCast and native multicast had lower packet loss than DiffServ Overlay Multicast. But this is mainly due to the fact that both DSMCast and Native Multicast use DVMRP as the routing algorithm, which allows packet replication in the core. But Native multicast is not scalable, as multicast routing information has to be maintained in the core, which defies the DiffServ principle of maintaining a stateless core. DSMCast does maintain a stateless core but it packs all the multicast routing information in the packet header. Hence the core routers have to perform extra effort to process and extract this information from the packet header. This again defies the DiffServ principle of keeping the core routers simple. DiffServ overlay multicast uses the overlay algorithm to tunnel the packets between edge routers. Thus DiffServ Overlay Multicast keeps the core routers simple and stateless at the expense of slightly higher packet loss for best effort service. The two versions of EBM also use overlay multicasting technology but they result in higher packet losses than DiffServ Overlay Multicast.

Thus DiffServ Overlay Multicast proves to be an efficient balance in terms of minimizing packet loss for receivers demanding best effort service and at the same time maintaining the rules of the DiffServ network for implementing a statewide and nationwide videoconference compared to other existing DiffServ multicast technologies.

For a videoconference where all the nodes demand EF service, DiffServ Overlay Multicast had the highest CAC acceptance percentage compared to other existing DiffServ multicasting technologies. This is mainly due to the fact that DiffServ Overlay Multicast uses the enhanced base overlay algorithm, which constructs trees taking into account the available bandwidth on the links. Thus in case of congestion the algorithm is able to come up with alternate routes thus increasing the connection rate. Also DiffServ Overlay multicast has a lower cost of reserving EF bandwidth per node for the videoconference compared to the EBM.

DiffServ Overlay Multicast also provides a good balance in terms of minimizing average AF packet loss per node and at the same time keeping the cost for reserving AF bandwidth low compared to the other existing DiffServ technologies.

Chapter 7 Conclusion & Future Work

This chapter summarizes the material presented and contributions of this thesis. In addition, the chapter also points out possible future research directions.

7.1 Conclusion

In order to successfully implement videoconference over IP network, we need to provide Quality of service. This can be achieved by implementing the videoconference over a DiffServ network. This requires integration of two technologies – DiffServ and IP Multicast and solving the associated problems arising due to the integration. This thesis proposes one such solution to effectively implement a videoconference in a DiffServ domain.

DiffServ Overlay Multicast provides a solution for the problems associated with DiffServ Multicast. The multicast tree is constructed only using the edge routers in the network, thus keeping the DiffServ core routers simple. It also uses the concept of Videoconferencing Assistant (VA) while constructing the multicast trees. The VA with the help of bandwidth broker is able to easily monitor the resources being consumed by the multicast tree. Hence the VA can check and reserve resources before joining a new receiver to the existing multicast tree. This helps in solving the neglected reservation subtree problem. DiffServ overlay multicast also provides support for multiple senders involved in the videoconference and different receivers demanding different level of QoS for the same multicast group.

Apart from solving the problems arising from integration of IP multicast with DiffServ, thesis also proposes a multicast routing algorithm that supports traffic engineering. All the edge routers constantly monitor network resources for evaluating the bandwidth and total stress to other edge routers in the domain. Edge routers also transmit this information to the ingress router to help it determine the best overlay tree for a particular multicast group. After all valid configurations are generated a score is assigned to each configuration according to the score function described in 4.2.5. The score is based taking into account the total stress and minimum available bandwidth for each configuration. Hence the DiffServ Overlay Multicast generates trees taking into account the available link bandwidth and stress on the links. Hence it finds alternate paths when there is congestion thus increasing the call admission success ratio and minimizing the average total packet loss for AF class. In addition the stress parameter in the score function also minimizes the cost for reserving EF and AF bandwidth in the diffserv network.

DiffServ Overlay Multicast is also compared with other existing diffserv multicast technologies. For a single source videoconference, DiffServ Overlay Multicast has a higher CAC acceptance ratio of 97%, which was 30% higher than its closest competitor. It was 67% for DSMCast/Native Multicast, 42% for EBM –Simple and 30% for EBM – Cluster. Also DiffServ Overlay Multicast had lower cost for reserving bandwidth than EBM and DSMCast. The cost for reserving EF bandwidth was 2.64 Mbps per EF destination for the DiffServ Overlay Multicast. It was 2.6 Mbps for DSMCast, 2.44 Mbps for Native Multicast, 2.83 for EBM –Simple and 3.22 Mbps for EBM – Cluster. Native

IP Multicast performed the best in terms of cost needed for reserving bandwidth but it does not provide a scalable solution in the DiffServ network. Finally DiffServ overlay multicast also had the lowest AF packet loss compared to other DiffServ Multicast technologies. The Average packet loss for AF destinations was 1.5% for DiffServ Overlay Multicast. It was 5% for DSMCast/Native Multicast, 10% for EBM –Simple and 15% for EBM – Cluster.

For multiple source videoconference there were similar results. DiffServ Overlay Multicast had the highest CAC acceptance ratio of 77%. It was 67% for DSMCast/Native Multicast, 32.6% for EBM –Simple and 28% for EBM – Cluster. The cost for reserving EF bandwidth was 10.42 Mbps per EF destination for DiffServ Overlay Multicast. It was 10.44 Mbps for DSMCast, 9.76 Mbps for Native Multicast, 11.3 for EBM –Simple and 12.93 Mbps for EBM – Cluster. The Average packet loss for AF destinations for DiffServ Overlay Multicast was 4.4%. It was 1% for DSMCast/Native Multicast, 20% for EBM – Simple and 22% for EBM – Cluster.

After analyzing and comparing the results, it can be concluded that DiffServ Overlay Multicast provides a better success rate of establishing the videoconference while maintaining a low packet loss in order to improve the overall performance of the videoconference. Also it has a lower cost for reserving EF and AF bandwidths compared to EBM-Simple and EBM –Cluster. Thus it provides a perfect balance of video quality and at the same time reducing the cost for setting up the videoconference.

7.2 Future Work

This dissertation has proposed a solution for implementing videoconference over a DiffServ network. The work has opened up an interesting area for future research work.

Using MPLS for explicit routing

DiffServ Overlay Multicast comes up with an overlay multicast tree to connect all the nodes involved in the videoconference. Overlay multicast tree consists of set of tunnels between the edge nodes. These tunnels are created using IP-IP tunneling [Sim95]. In IP-IP tunneling, the original IP packet is wrapped by an IP header. The encapsulation technique is fairly simple. An outer IP header is added before the original IP header. The outer IP header Source and Destination identify the endpoints of the tunnel. The inner IP header Source and Destination identify the original sender and recipient of the packet. But this form of tunneling does not have explicit control over the path of the entire tunnel. The tunnel path is the shortest path between the source and the destination of the tunnel. Instead of using IP-IP tunneling, MPLS can be used to set up tunnels and route packets for the overlay tree.

Multiprotocol Label Switching (MPLS) [RoV01] has emerged as an important technology for the Internet. It represents the convergence of two fundamentally different approaches: datagram and virtual circuit. IP forwarding in the Internet is traditionally based on datagram model: routing protocols are used to pre-calculate the paths to all destination networks by exchanging routing information and each packet is forwarded independently based on its destination address. But this means that the source has no control over the path to the destination. For the purpose of traffic engineering it would be

ideal if the source had explicit control over the entire path to the destination. MPLS provides a mechanism for explicit control over routing paths.

MPLS uses this technique of Label Switching. Label Switching uses a short, fixed length label inserted in the packet header to forward packets. When a Label Switched Router (LSR) receives a labeled packet, it uses the incoming label in the packet header to find out the next hop and also the corresponding outgoing label. With label switching the path that a packet traverses through is called a Label Switched Path (LSP). LSP has to be set up before it can be used for label switching. For explicitly routed LSPs that require QoS guarantees, the IETF MPLS working group has proposed CR-LDP [AnD01] and RSVP-TE [AwB01] as two protocols that support such functions.

The future research would be to implement DiffServ Overlay Multicast using MPLS for routing of packets in the overlay tunnels between the different nodes involved in the videoconference.

References

[AdN02] A. Adams, J. Nicholas, and W. Siadak, “Protocol Independent Multicast - Dense Mode (PIM-DM): “, Protocol specification (revised)," *IETF Internet Draft draft-ietf-pim-dm-new-v2-01.txt, Feb. 2002.*

[AnD01] L. Anderson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, “LDP Specification”, *January 2001.*

[AwB01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, “RSVP-TE: Extensions to RSVP for LSP Tunnels”, *December 2001.*

[Ber98] Y. Bernet et al., “A Framework for Differentiated Services”, *Internet draft <draft-ietf-Diffserv-framework-00.txt>, May 1998.*

[BeW04] R. Bless and K. Wehrle, “IP Multicast in Differentiated Services Network”, *RFC 3754, April 2004.*

[BlB98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An Architecture for Differentiated Services”, *RFC 2475, Dec. 1998.*

[BrJ03] J. Brooks and M. Jurczyk, “Creating edge-to-edge multicast overlay trees for real time video distribution”, *Internet Computing '03 / International MultiConference in Computer Science, Vol.1, pp. 371-377, Las Vegas, Nevada, June 2003.*

[BrC94] R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: An Overview”, *Internet RFC 1633, Jun. 1994.*

[BrC98] R. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroklaski and L. Zhang “Recommendations on Queue Management and Congestion Avoidance”, *Internet RFC 2309, Apr. 1998.*

[Bro03] J. Brooks “Edge-To-Edge Multicast Overlay Trees for Real Time Video Distribution”, *MS-Thesis presented to University of Missouri – Columbia, Dec. 2003*

[BrZ97] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, “Resource Reservation Protocol (RSVP) -- Version 1 Functional Specification”, *RFC 2205, Sept. 1997.*

[Cis01] “DiffServ – The Scalable End-to-End QoS Model”, *white paper, Cisco Systems Inc, 2001.*

[Dee89] S. Deering “Host Extensions for IP Multicasting”, *RFC 1112, Aug. 1989.*

[FeH98] P. Ferguson and G. Huston “Quality of Service: delivering QoS on the Internet and corporate networks”, *John Wiley & Sons. Inc, 1998.*

[HeB99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, “Assured Forwarding PHB Group”, *RFC 2597, June 1999.*

[JaN99] V. Jacobson, K. Nichols, and K. Poduri, “An Expedited Forwarding PHB”, *RFC 2598, June 1999.*

[LiM02] Z. Li and P. Mohapatra, “QDM: QoS-Aware Multicasting in DiffServ Domains”, *in Proc. of Global Internet Symposium, 2002.*

[Moy94] J. Moy, "MOSPF: Analysis and Experience", *RFC 1585, Mar. 1994.*

[NS2] Network Simulator Version 2 (NS-2), http://www.isi.edu/nsnam

[RoV01] E. Rosen, A. Vishwanathan, R. Callon, "Multiprotocol Label Switching Architecture", *RFC 3031, January 2001.*

[ShP97] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service", *RFC 2212, Sept. 1997.*

[Sim95] W. Simpson, "IP in IP Tunneling", *RFC 1853, October1995.*

[StM01] A. Striegel and G. Manimaran, "DSMCast: A scalable approach to DiffServ multicasting", *in Proc. of ICC 2001, Helsinki, Finland, June 2001.*

[Str01] A. Striegel and G. Manimaran, "A scalable protocol for member join/leave in DiffServ Multicast", *in Proc. of Local Computer Networks (LCN), Tampa, Florida, Nov. 2001.*

[StB03] A. Striegel, A. Bouabdallah, H. Bettahar and G. Manimaran, "EBM: A new approach for scalable DiffServ multicasting", *in Proc. Network Group Communications (NGC), Munich, Germany, Sept 2003.*

[Wan01] Z. Wang, "Internet QoS – Architectures and mechanisms for Quality of Service", *Lucent Technologies, 2001.*

[Wro97] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", *RFC 2211, Sept. 1997.*

[Wax88] B. M. Waxman, "Routing of Multipoint Connections", *IEEE Journal of Selected Areas in Communication, vol.6, no. 9, pp. 1617–1622, December 1988.*

[WaP88] D. Waitzman, C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol", *RFC 1075, November 1988.*

[ZeC96] E. Zegura, K. Calvert and S. Bhattacharjee, "How to model an Internetwork", *proceedings of IEEE Infocom '96, San Francisco, CA.*