A Hybrid Approach to Differentiated Services Multicast

Joydeep Dhar

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science

Concordia University

Montreal, Quebec, Canada

August 2004

Your file Votre référence
ISBN: 0-612-94695-9
Our file Notre référence
ISBN: 0-612-94695-9

Abstract

A Hybrid Approach to Differentiated Services Multicast

Joydeep Dhar

Internet, the largest network of networks has evolved from a research-oriented network to one with a myriad number of commercial applications. Over the last several years, there has been an explosion in the introduction of new Internet technologies with high-end workstations being engaged in real-time and multimedia communications like video conferencing, IP telephony, streaming video broadcast, online gaming etc. Supporting both the legacy services like email, file transfer and real-time multimedia like video conferencing requires service differentiation of Internet traffic. Moreover, multipoint communication requires implementation of multicast services. Two of the emerging technologies for service differentiation in multipoint communication are Differentiated Services (DiffServ) and multicasting. Although the two technologies share complementary goals, the integration of the two technologies is a non-trivial issue due to three fundamental problems. The problems are the scalability of per-group state information, sender- versus receiver- driven QoS, and resource management.

The issues surrounding how to solve these problems provide the basis for this thesis. Edge Based Multicast (EdgeCast) – has been proposed here to satisfy the requirements for scalable DiffServ multicasting architectures. In addition it also presents a new join/leave protocol to support the EdgeCast architecture. Performance simulation has been done and compared with some already developed techniques. Finally, the whole work has been summarized and comments have been made on future applications of the architecture and several potential areas for future research.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Anjali Agarwal, for her continued support, guidance and encouragement during the period of this research. Her regular and constructive feedback on the topic always led me to the proper direction.

I always find that, words are not enough to express my love and gratitude to my parents. Without their love, patience and moral support I would never make it so far. Thank you for everything.

I also like to thank all my friends at Concordia University for making my days exciting during my stay in Montreal.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AF	Assured Forwarding
BB	Bandwidth Broker
BE	Best Effort
DiffServ	Differentiated Services
DM	Dense Mode
DSCP	DiffServ Code Point
DSMCast	Differentiated Services Multicast
ECN	Explicit Congestion Notification
EdgeCast	Edge Based Multicast
EEH	Encapsulated Edge Header
EF	Expedited Forwarding
FTP	File Transfer Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IIF	Incoming Interface
IP	Internet Protocol
ISP	Internet Service Provider
MBone	Multicast Backbone
MFC	Multicast Forwarding Cache
MPLS	Multi Protocol Label Switching
MRT	Multicast Routing Table
NRS	Neglected Reservation Subtree
OIF	Outgoing Interface
PHB	Per Hop Behavior
PIM	Protocol Independent Multicast
QoS	Quality of Services
RSVP	Resource Reservation Protocol
SGM	Small Group Multicast
SLA	Service Level Agreement
SM	Sparse Mode
SSM	Source Specific Multicast
TEH	Tree Encapsulation Header
VoIP	Voice over IP
WAN	Wide Area Network
XBB	Extended Bandwidth Broker

Chapter 1

INTRODUCTION

Internet, the largest network of networks has evolved from a research-oriented network to one with a myriad number of commercial applications. Over the last several years, there has been an explosion in the introduction of new Internet technologies supporting real-time and multimedia communications like video conferencing, IP telephony, streaming video broadcast, online gaming etc. Supporting both the legacy services like email, file transfer and real-time multimedia like video conferencing requires service differentiation of Internet traffic. Moreover there has been a push to point-to-multipoint communication from simple point-to-point communication as a method of saving bandwidth and time. All these have prompted the implementation of Quality of Services (QoS) sensitive multicast [1] in the Internet. Different architectures like IntServ-RSVP [2, 3], DiffServ [4, 5], and MPLS [6] have been proposed for QoS support. Protocols like DVMRP [7], PIM-DM/SM [8, 9] have been developed to support multicast. For QoS, Differentiated Services (DiffServ) model has received more attention because of its simplicity and scalability. As a result of increasing QoS multicast demand, the integration of DiffServ multicast seems to be the most promising. Although both are complementary technologies, the integration of the two is a rather difficult task due to architectural conflicts between multicasting and DiffServ principles- core statelessness for scalability versus per-group state information for efficiency.

The rest of this chapter is organized as follows. Section 1.1 describes the Internet communication from the connectivity perspective with brief descriptions of unicast, broadcast and multicast. Section 1.2 outlines the basic components required for multicast implementation. Next Section 1.3 outlines the concept of Quality of Services (QoS) and its importance in the future Internet. The subsections introduce the different QoS models. Section 1.4 outlines the problems that are the motivations of this thesis. Section 1.5 is all about how the thesis has been organized and which chapter discusses which issues. Finally, Section 1.6 is the summary of the current chapter.

1.1 From Point-to-Point to Point-to-Multipoint

The fundamental method of network communications is between two host computers, or unicasting. In a single session one computer transfers data to another computer. The one-to-one sessions can offer a great deal of control of the data traffic between the source and the receiver, allowing for acknowledgement of receipt, requests for transmission rate, and so on. But many internet applications involve one-to-many or many-to-many (multipoint) communications, where one or more sources are sending data to multiple receivers. For example in a corporate network all the employees may need to watch a webcast about the important company policy by the company CEO. This situation can be handled with point-to-point communication, but with a penalty of network bandwidth and time required to perform the job. On the other hand multipoint communication (point-to-multipoint and multipoint-to-point communication) is the best possible answer to serve this application efficiently.

Multipoint communication is possible in three different ways – *unicasting*, where a separate copy of data is delivered to each of the recipients; *broadcasting*, where a data packet is flooded to the whole network and all the recipients receive data even when some of them are not interested; and *multicasting*, where a single packet is addressed to all intended recipients and the network replicates packets only as needed.

1.1.1 Unicasting, Broadcasting and Multicasting

In multipoint unicasting, a source sends an individual copy of a message to each recipient. In such cases, the number of receivers is limited by the sender's bandwidth. Transferring a file from a File Transfer Protocol (FTP) [10] file server to a host computer is an example of unicasting; the data in the file is sent over the network from the server only to the host. But if five other people in the workgroup want to copy the same file to their own computers at the same time using FTP, the FTP server would have to send the file to each of the five recipients separately, using five times as much bandwidth as the single transfer. Even when the same data must be sent to more than one recipient at the same time, unicasting works well as long as the number of recipients is small.

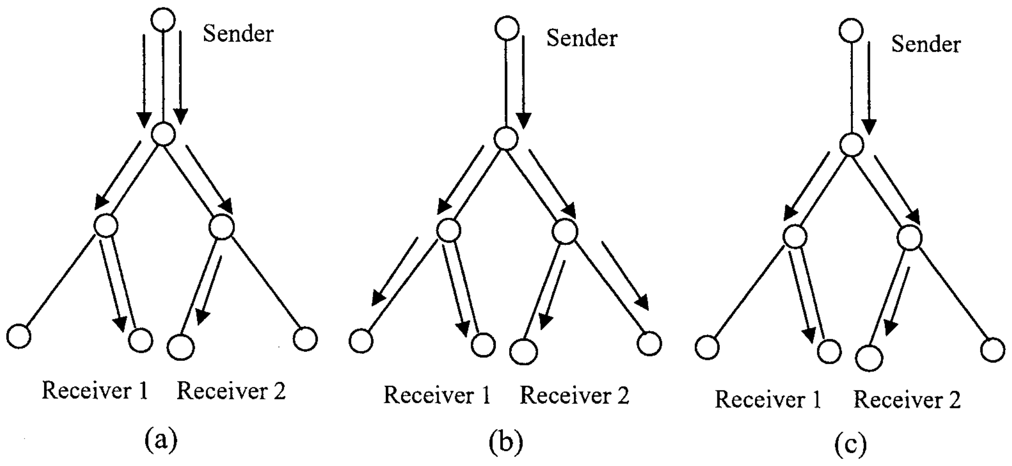


Fig 1.1 (a) Multiple Unicasting (b) Broadcasting (c) Multicasting

Broadcasting is at the other end of network spectrum. In this method the source sends a single copy of the message and all network nodes replicates the packet such a way that all the nodes in the network gets a copy of the message. The receiving nodes decide whether to accept or reject the message. Wide Area Networks (WAN)-based broadcasting has to depend on network devices like routers to duplicate packets and distribute them among the subnets compromising the WAN or internetwork. Broadcast traffic can quickly grow out of control at larger sites (e.g., greater than 300 or more networked devices) and reduce the bandwidth available to support mission-critical applications (and even general-purpose traffic). In worst-case scenarios, broadcast storm can effectively shutdown the network, since the broadcasts can monopolize all of the available bandwidth.

Multicasting falls between unicasting and broadcasting from the data flow perspective. Rather than sending data to a single host (unicasting) or to all hosts on a network (broadcasting), multicasting aims to deliver data to a select group of hosts, called the '*host group*'. The '*host group*' is defined by a specific multicast address – Class D address. Unlike, broadcasting, multicasting allows each host to choose whether it wants to participate in a multicast session. Once a host group is set up and the sender starts transmitting packets to the host group address, the network infrastructure takes on the responsibility for delivering the necessary data streams to all members of the group. Only one copy of a multicast message passes over any link (such as a router) in the network; copies of the message are only made when paths diverge at a router (e.g., the message is supposed to be passed on to another router as well as to a workstation attached to the current router), helping to conserve bandwidth.

1.2 The Basic Components of Multicasting

There are four major processes behind a successful implementation of multicasting:

- First, there is the definition of multicast host group, which is handled on IP networks by specially coded addresses called multicast addresses. These are the class D addresses in IPv4.
- Second, a mechanism is required for joining and leaving a multicast host group. This not only includes either sender- or receiver-based control of the group membership, but also the protocols used to transmit and manage the group membership information throughout the network, if necessary. Group membership is handled via Internet Group Management Protocol (IGMP) [11], which handles communications between routers and the Local Area Network (LAN) they serve.
- Complex IP networks cannot forward data without the assistance of routers, which form the third component of a multicasting system. A series of routing protocols have been specifically formulated to support multicasting, both to handle the duplication of multicast traffic as needed and, more importantly, to handle issues surrounding group management.
- Lastly, there are the application protocols for creating and managing the data that are distributed in a multicasting session.

1.3 QoS: Beyond Best Effort Services

Since its inception, Internet has come a long way within a fairly short amount of time. The technology that once started as a means of mere inter-device communication has now evolved into the principal source of information, entertainment, research and business. Different types of applications are required for serving different purposes. And all these different applications have different service constraints. E-mail exchange, file transfer applications and audio over Internet have to rely on faithful delivery than time-constraint delivery. On the other hand, in interactive applications – video conferencing and streaming video, for example – impose special restrictions on latency than accuracy. So far Internet had been providing best-effort (BE) services, which treats all the traffic streams same – try the best to deliver, no guarantee about delay and accuracy. But the Web and Internet together constitute a critical information, entertainment, and commerce infrastructure that are rapidly evolving from a BE service model to one with service differentiation provided for users, services and applications. This *service differentiation* (also referred to as Quality of Service (QoS)) is in the form of preferential treatment (using priorities, resource reservation, etc.) of one type of user/application traffic over another at the servers, proxies, and network elements that comprise the end-to-end infrastructure.

1.3.1 Major QoS Parameters

As we mentioned earlier, not all the applications have the same accuracy and delay constraints. For example, voice applications can tolerate more delay than video applications, but they require better accuracy. On the other hand, real-time video applications require reception with less delay than accurate reception. If few frames are lost in transmission still the receiver will be able to recognize the video properly, but reception with greater delay will affect the real-time aspects of the content. The major QoS parameters are accuracy, delay, and jitter.

- ***Accuracy*** is directly connected to the packet loss ratio. This is a major aspect that determines the quality of voice in packet networks. Each packet is assigned a header that identifies where the packet is going and contains the information for reassembly when the packet arrives at its destination. Unless the network is precisely matched to the peak traffic load, packets sometime fail to arrive at the destination. These lost packets affect the network services in different degrees depending on whether the application type is data transmission or voice/video distribution.

- ***Delay*** is the primary QoS measure for interactive multimedia applications. It is measured end-to-end across the packet network from the source, across the packet network, to the destination. The upper limit of delay that is tolerated by the International Telecommunication Union (ITU) is 150 ms [12]. In the conventional Public Switched Telephone Networks (PSTN), the largest part of the end-to-end delay is the propagation time of the transport medium [13]. Then there is processing delay which is much less than delays created by queuing and propagation. Processing delay includes the time taken for encoding and decoding speech, collecting the voice data into packets, etc. Buffering delay also adds up to affect the quality of multimedia and real-time packets. Buffers are used for queuing at routers and to control packet arrival time at the decoder, and data waits in the buffer for processing and propagation.
- ***Jitter*** is the variation in delay over time from point-to-point. If the delay of transmissions varies too widely in a 'Voice over IP' (VoIP) call, the call quality is greatly degraded. This is due to different queuing delays experienced by different packets. The amount of jitter tolerable on the network is affected by the depth of the jitter buffer on the network equipment in the voice path. The more jitter buffer available, the more the network can reduce the effects of jitter.

1.3.2 QoS Service Models

The Internet Engineering Task Force (IETF) has proposed many service models and mechanisms to meet the demand for QoS. Notably among these are the *Integrated*

Services/RSVP model, the *Differentiated Services* (DS) model, *MPLS*, *Traffic Engineering* [14] and *Constraint Based Routing* [15].

1.3.2.1 Integrated Services and RSVP

The Integrated Services model is characterized by resource reservation. For real-time applications, before data are transmitted, the applications must first set up paths and reserve resources. RSVP is a signaling protocol for setting up paths and reserving resources.

This model proposes two service classes in addition to *Best Effort Service*. They are: 1) *Guaranteed Service* [16] for applications requiring fixed delay bound; and 2) *Controlled Load Service* [17] for applications requiring reliable and enhanced best effort service. The routers, to be able to reserve resources for providing special QoS for specific user packet streams or flows, require flow-specific state in them. Because of the end-to-end resource and per-flow state reservation requirements the internetwide deployment of IntServ-RSVP is not feasible.

1.3.2.2 Differentiated Services

The primary goal of Differentiated Services (DiffServ) is to provide the benefits of QoS without the scalability limitations of IntServ. Rather than attempting to deal with QoS on a per-flow basis, the DiffServ model aggregates traffic with similar QoS requirements into classes of traffic. DiffServ eliminates two key weaknesses of the IntServ model - the setup and the maintenance of per-flow state information. In the DiffServ architecture, intelligence is migrated to the edge of the domain in order to keep the core of the network simple and scalable. Routers in a DiffServ domain are divided into two categories, core routers (simple and high speed) and edge routers (stateful and intelligent).

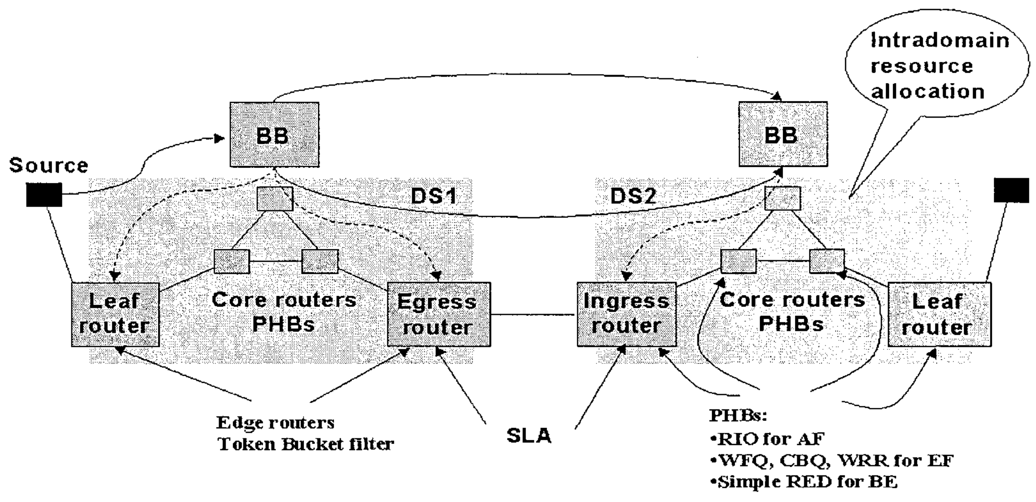


Fig 1.2 Two Adjacent DiffServ Networks

When a packet enters the network, it must first pass through an edge router. The edge router is responsible for policing and shaping all packets that pass onto the DiffServ network. Alternatively, the edge router may re-mark or drop a packet in the event that the flow or source has violated its respective Service Level Agreement (SLA). Once a packet enters the network, the packet is scheduled for transmission on the link according to the marking (DSCP - DiffServ Code Point) [4, 5] of the packet. In order to allow for gradual deployment, the DS (DiffServ) field is simply a semantic change to the original ToS field of IPv4 and the Flow ID field of IPv6, thus allowing DiffServ to operate without major changes to the Internet framework. The DS field is made up of six bits with the remaining two bits left over for protocols such as Expedited Congestion Notification (ECN [18]).

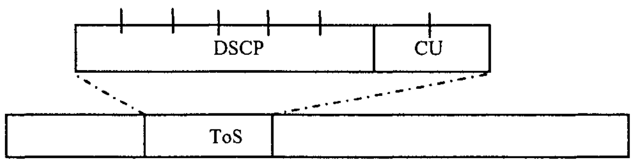


Fig 1.3 IPv4 Header and DSCP in Type of Service field

Core routers do not have per-flow state information and differentiate packets according to the marking of the packet. In contrast, edge routers are stateful entities that are responsible for policing and/or marking all packets according to an SLA between the source (other ISP, user, and company) and the domain, or between two domains. The DSCP of the packet defines the PHB or Per-Hop Behavior [4, 5] that should be applied to the packet for scheduling. The PHB defines how the packet will be scheduled as well as dealt with during congestion. In addition, certain PHBs may be rate-limited as well (such as Expedited Forwarding (EF) [19]) to prevent link starvation of lower classes by high priority traffic. Each core node in the network is independent and the underlying scheduling mechanism is left up to the manufacturer.

The IETF defined PHBs include an EF PHB for low-loss, low-delay traffic such as VoIP, Assured Forwarding (AF) PHB [20] which defines only the congestion priority of the traffic (low, medium, high). In addition, several other PHBs have also been proposed as IETF drafts [21, 22, 23].

1.3.2.3 Multi-Protocol Label Switching

Multi-Protocol Label Switching (MPLS) is a forwarding scheme where packets are assigned labels at the ingress of a MPLS-capable domain. Subsequent classification, forwarding, and services for the packets are based on the labels. Traffic Engineering is the process of arranging how traffic flows through the network. Constraint Based Routing is to find routes that are subject to some constraints such as bandwidth or delay requirement.

MPLS is the latest step in the evolution of multilayer switching in the Internet. It is an IETF standards-based approach built on the efforts of the various proprietary multilayer switching solutions. MPLS uses the control-driven model to initiate the assignment and distribution of label bindings for the establishment of label-switched paths (LSPs) [9]. LSPs are simplex in nature (traffic flows in one direction from the head-end toward the tail-end); duplex traffic requires two LSPs, one LSP to carry traffic in each direction. An LSP is created by concatenating one or more label switched hops, allowing a packet to be forwarded from one label-switching router (LSR) [9] to another LSR across the MPLS domain. An LSR is a router that supports MPLS-based forwarding.

The MPLS control component centers around IP functionality, which is similar to proprietary multilayer switching solutions. However, MPLS defines new standard-based IP signaling and label distribution protocols, as well as extensions to existing protocols, to support multi-vendor interoperability. MPLS does not implement any of the ATM Forum signaling or routing protocols so the complexity of coordinating two different protocol architectures is eliminated. In this way, MPLS brings significant benefits to a packet-oriented Internet.

1.4 Problem Motivation

The important problem regarding the Internet today is, to provide the multi-level services to meet the user needs while keeping such implementations scalable to the billions of users present on the Internet. Two of the emerging technologies for the solution to this problem are DiffServ and multicasting. DiffServ proposes a model for QoS while multicasting proposes a model for group applications. Moreover the multicast states in the routers are dependent on the number of active (source, group). A potential solution to this problem is making the multicast states independent of the number of (source, group).

Also supporting heterogeneous QoS to meet group members' different requirements also has to be addressed. The issues surrounding integration of DiffServ and providing heterogeneous QoS are the motivations of this thesis.

1.5 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 outlines the problems of DiffServ multicast, the general approaches of QoS multicast, the background and related work and an overview of the proposed EdgeCast solution. Chapter 3 gives a complete architecture of EdgeCast including a new join/leave protocol description. It also discusses several considerations about EdgeCast implementation. Chapter 4 provides the theoretical analysis of EdgeCast and the simulation results showing the performance analysis. It also provides comparative analysis of EdgeCast with some other related approaches. Finally, Chapter 5 concludes the thesis by summarizing the architecture, addressing other applicable issues, and discussing future research directions.

1.6 Summary

This chapter introduced multicasting and QoS based Internet – the components and parameters of multicast and QoS. Brief description of multicast techniques, QoS models like DiffServ, IntServ-RSVP, and MPLS has also been provided. The problem motivation section introduced the motivation behind the thesis work. The complete thesis organization has also been described to outline the contents of the subsequent chapters.

Chapter 2

DIFFSERV MULTICAST PROBLEMS AND RELATED WORK

2.1 Introduction

Although multicasting is a good way of conserving bandwidth while transmitting data to multiple receivers, it still has its additional complexities that may need to be addressed. The successful deployment of multicasting on WANs requires that these issues be resolved. Moreover, deployment of multicast in a DiffServ network requires some other special issues to be resolved – state scalability, simple core, resource utilization and reservation, and heterogeneous QoS support. This chapter describes the primary conflicts between DiffServ and multicasting. Before that some fundamental issues regarding multicasting is outlined so that the DiffServ multicast problems can be explained in the light of those issues. The overview of existing solutions as well as the proposed solutions are categorized and examined.

Rest of this chapter is organized as follows. Section 2.2 outlines some important multicasting issues that need to be considered for implementation. Section 2.3 defines the conflicts between DiffServ and multicasting in the light of the previous section. Next, Section 2.4 introduces the three general classes of solutions to the DiffServ multicasting problem. Then, Section 2.5 presents existing solutions for the DiffServ multicasting problem and discusses the strengths and weaknesses of each approach. Following that, Section 2.6 describes a brief overview of the solution proposed in this thesis and its major properties and uniqueness.

2.2 Fundamental Issues in Multicasting

There are some fundamental issues [24] related to successful multicast operations which need to be considered prior to multicast implementation. As we know, receivers join a multicast group to receive data and when they are not interested in that group's data, they leave the group. So there should be a proper method of group management involved. Since multicast involves replication of packets, the efficient utilization of network resources is also of great importance. Another problem is managing multicast state scalability. All these issues are discussed below in brief.

2.2.1 Joining and Leaving a Group

Receiving multicast data requires that the interested receivers join a multicast group. The major aspect of multicasting is that the receivers who have expressed interest in receiving data – in other words, joined a multicast group will only receive data from the source of that group. When there is no need of receiving data from that group, it is required to leave that particular group.

It is difficult to have the complete knowledge about the active or available multicast sessions on an internetwork. Multicast Backbone (MBone) [25] developers experimented with applications that list multicast groups like a TV guide publication. New multicast sessions can also be announced using mailing lists, web pages or email in a secured environment. Receiver anonymity is an interesting point in multicast, that the sender does not have knowledge about the addresses of the receivers that have joined the host group. Senders only know the group address where the data needs to be sent. This host group member anonymity can sometimes lead to problems such as data delivery on the tree even when there is no receiver to receive [25].

Management of groups also gets complicated because of the fact that multicast group membership should be dynamic – group members should be able to join/leave the group in arbitrary fashion during the session. Although from the users' point of view this seems very much convenient, it naturally causes problems in tracking active members. Concerning the management of group membership of the receivers they serve, the routers bear the responsibility of updating each other. Traffic generated solely for managing group membership may not be large; the delays associated with leaving a host group can lead to wasted bandwidth [24]. Thus the dynamic nature of group membership can have a negative effect on network bandwidth.

2.2.2 Efficient Transmission of Multicast Traffic

Multicast involves packet replication and the degree of replication depends on the tree structure or how dense/sparse the group members are situated. Data transmission with the efficient use of network resources is therefore a very important aspect of multicast transmission. An important property of a good protocol design is the accommodation of efficient resource usage [12]. For unicast protocol this property can be accommodated fairly with ease. On the other hand, for multicasting, it is rather difficult to decide which resources should be optimized and trade-offs are required. Rather than along arbitrary paths multicast routing trees carry traffic to the designated receivers to minimize transmission costs through link sharing. For real-time multimedia applications there are some other additional factors such as data loss constraint, delay constraint and media heterogeneity [24].

The issues related to the integration and interaction of routing and resource reservation further complicates matters. When we think about QoS support, the success in building a multicast tree depends on having adequate resources at each router. Even for a static network this guaranteed resource reservation is not easy. Moreover when the network becomes dynamic (joining, leaving, changing groups) this issue becomes even more problematic since the involved network resources change as multicast tree changes.

2.2.3 Time-Sensitive Delivery of Multicast Traffic

The issues regarding QoS sensitive multicast transport rise as the use of real-time multimedia data rises. Naturally, it poses some special restrictions on the transmission capabilities of packet-switched networks like the Internet. Situations like simultaneous data delivery to multiple recipients are also in greater demand. Time sensitive data, such as stock quotes and other financial data need to be delivered to all recipients at the same time, or at least within a narrow window of time. The video and audio data require a different type of time sensitivity related to the temporal sequencing [24]. Only the maintenance of data sequencing during delivery is not enough; the synchronization issues between different, but related, data streams also need to be addressed. Even if a resource reservation protocol like RSVP is used to maintain a path with guaranteed quality of service, it is still likely that different paths will be chosen for audio and video streams, since the audio stream can tolerate a lower QoS than video stream [12].

2.2.4 Guaranteed Arrival of Multicast Traffic

Guaranteed delivery of data to a series of receivers is as important as the time constraint delivery. It is known that IP is a best-effort protocol and hence IP multicasting also becomes a best-effort service by default. This means that IP multicasting is unreliable by default. To ensure QoS, this means that a reliable protocol has to be utilized with IP multicasting in order to accommodate applications such as financial transactions or file transfers, where any loss of data is unacceptable [24]. There are many proposals for reliable multicast protocols currently under consideration by IETF, and some of these have already been used in commercial products, so routine reliable multicasting is merely a matter of time [12].

2.2.5 Scalability

Problem of scalability related to multipoint communication in the Internet is another major issue of concern. Lots of research activities are focused to eliminate or at least minimize this problem. The scalability problem related to multicast protocols arises from the use of groups or <source, group> tuple. Unlike broadcasting, multicasting delivers data selectively to a number of interested receivers and therefore, entire subnets might not have any receivers. It is the responsibility of the protocols to be able to construct the proper distribution topology and keep it updated. Considering the huge Internet topology and its increase day by day, the density of host group members distributed over the inter-network may vary widely from the topologically dense groups to very sparse groupings. Therefore, it is not a simple matter to keep the network infrastructure up to date [24].

Moreover, as the number of active multicast groups grows, the routers must store more information about the involved groups in the multicast network. This additional stored information consumes more resources and may have profound effect on the routers performing other network duties, such as unicast routing, QoS reservations and so on [23]. The multicast trees that are constructed by the most common routing protocols tend to be as close as possible to the tree of shortest-path routes. As the network dynamics changes, so does the shape of a distribution tree. Moreover, the distribution trees can be global and hence they can be subject to a high frequency of control traffic [24].

2.3 Differentiated Services Multicast Problems

Achieving scalability is the primary conflict between DiffServ and multicasting. Whereas DiffServ relies on only edge routers possessing intelligence and state information, multicasting relies on per-group state information throughout the entire network. Thus, when trying to integrate the two technologies, one is faced with two conflicting principles, core statelessness or simple core for scalability versus per-group state information for efficiency.

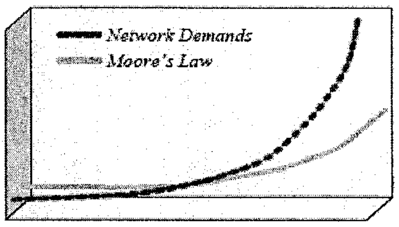


Fig 2.1 Network speed vs. bandwidth demands [26]

The research community is still divided on whether state information (storage and router complexity) or maximal network efficiency (bandwidth) is more important than the other [27]. Never-the-less, it is still arguable that, if bandwidth capacity can be increased, the state storage capacity can also be increased. This argument leads to some other basic issues to be considered. First, the growths of bandwidth capacity and demand are far outpacing the improvements in CPU performance (see Fig 2.1). This means, per-packet processing better kept to minimum which is extremely difficult with per-group state (also with per-flow state). Second, the bandwidth increase compounds the fact that it is the maintenance of state information (router complexity) and not necessarily storage that is the problem [27]. This exacerbates the problem that increased bandwidth may lead to additional multicast packets and hence, the amount of per-group or per-(source, group) state information/group dynamics would increase as well.

It is to be noted that, the problem of scalability in multicasting is not necessarily unique to DiffServ. It is more important in DiffServ scenario is because of the fundamental principle of simplified core operation in DiffServ. The primary conflicts between DiffServ and multicasting [27] are summarized below:

- *Per-(source, group) state scalability*: The per-(source, group) state information of multicasting is not scalable to the rather simple DiffServ core. Thus, any solutions for DiffServ multicasting must address this fundamental issue.
- *Sender versus Receiver-driven QoS*: Since ingress router marks packets in a DiffServ network, the QoS provided is sender-driven. But multicasting is more a receiver-driven service since the group members join the group for data reception. Hence QoS request also comes from the receivers. As a result, appropriate mechanisms must be developed that bridge the gap between the sender-driven QoS of DiffServ and the receiver-driven QoS of multicasting.
- *Management of Resource Utilization:* In multicast service, unlike unicast, a packet may replicate into one or more copies in the network. Moreover the ingress node (entrance to the DiffServ domain) may not necessarily know the exact makeup of the multicast tree and the total QoS supporting capability of the network. Thus, appropriate signaling or management mechanisms must be introduced in order to manage the resource impact of multicasting on the network.

2.3.1 Per-(Source, Group) State Scalability

It is evident that the primary conflict between DiffServ and multicasting lies in the per-(source, group) state information required for multicasting. For each active multicast (source, group) across the domain, each router must maintain as well as manage the state information (group states, interfaces, timers etc.). When a large number of sources and groups are concurrently active in the network the maintenance of the required information leads the scalability problem.

In Source Specific Multicast (SSM) [28] the tuple (source, group) is used for group membership. Here the group identifier is expanded from the traditional IP multicast (Anycast) of only the group address (*, group). Let us consider an example where an individual web server employs SSM. Obviously this results in the multicast group address space expands significantly. We also consider the web server is set up to place each web object as an individual group. At first this may seem "far-fetched", such a scheme is not impossible to envision [27]. Let us also consider thousands of or all of the web servers on the Internet employed such a scheme. In such a case the traditional multicast imposes the problem of supporting and maintaining state information for millions of multicast groups [27]. The cost of such a huge storage and maintenance support in the core routers of the Internet would be huge. Thus, it is logical to think of some approach for DiffServ where the multicast states would be independent of (source, group) rather would depend on network topology (network edge to be precise). Such thinking provides the motivation of the work in this thesis and is addressed in detail in Chapter 3.

2.3.2 Sender versus Receiver-Driven QoS

Another conflict between DiffServ and multicasting arises from the nature of how the QoS is selected [24, 27]. Since the ingress router performs the traffic conditioning, marking and policing, QoS in DiffServ is provided through a sender-driven manner. But as we think of multicast, QoS is more receiver-driven since receiver asks for data from a specific group with a desired QoS. As only the receiver knows its own QoS requirements, the receiver is the only entity that can drive the appropriate QoS [27]. Hence, in DiffServ multicast, there exists a conflict in how QoS is driven.

If the sender-driven QoS is considered, then the receivers do not have much choice – either all of them subscribe to the same delivered QoS level for receiving data from a group or none of them join the group at all. This scheme may be the simplest but is completely incompatible with the future requirements and there is no flexibility at all. The other choice can be allocating individual QoS level requested by the receivers to the separate groups. This kind of scheme meets the QoS levels for the receivers but the bandwidth costs of each additional tree for the additional QoS levels will be well beyond the optimum level [27].

The most promising solution is to provide QoS in dynamic fashion where QoS level of the packet will change on the fly for supporting heterogeneous multicast receivers without requiring separate trees for each QoS level [22, 23, 27]. The unique aspect of this scheme is investigated in more detail in Chapter 3.

2.3.3 Management of Resource Utilization

Resource utilization is another aspect of conflict between DiffServ and multicasting. As mentioned earlier, such conflict is not unique with DiffServ multicast (traditional QoS sensitive IP multicasting also requires the resource management), it introduces several unique problems when multicasting is considered in DiffServ domain. In short, the conflict regarding resource management can be reduced to the problem of packet replication [29]. In unicast, one packet that goes in a router also comes out as a single packet. But in multicast, the packet that goes inside a router may be replicated in multiple copies. Hence, the resource consumption may multiply based on the degree of replication. The cost may vary between the cost of separate unicasts across the domain and the cost of a single unicast (no branching in the multicast tree) [30, 31]. The condition to allocate network resources efficiently is to properly quantify the resource utilization of a multicast group on the DiffServ domain and shape/police multicast packets at the ingress routers accordingly [27]. Although several architectures have studied the allocation of such resources for single sourced groups [32, 33, 34], the introduction of an architecture for both provisioning and policing of multiple sourced groups has yet to appear [27].

If the resource utilization for multicast traffic can not be quantified properly then a unique problem called Neglected Reservation Subtree (NRS) [22, 23] problem may occur. NRS problem can occur if member join messages are not approved for resource allocation before data is transmitted on the branch. The problem finds its basis in the distributed nature of the multicast tree and the separation of multicast routing/replication from the ingress-based policing/allocation of DiffServ [27]. While conditioning and policing, the edge router has no idea what is the shape of the tree and what is the degree of replication inside the core. Hence the edge cannot determine the resource requirements to deliver data through the tree. As a result, all the network resources along the path, for a new join request, must appropriately reserve resources for the new branch. This requirement incurs extra storage and processing overhead in the core routers which is not desirable from the DiffServ perspective.

2.4 Overview of DiffServ Multicast Approaches

In order to offer support for DiffServ multicasting, the solutions can be divided into three main classes, state-based [22], edge-based [27], and encapsulation-based [27]. The approaches are summarized in Table 2.1 and discussed in more detail later.

Table 2.1 Summary of DiffServ multicast approaches [27]

Type	**Replication Allowed**	**Replication Information**	**Strength (+)** ***Weakness (-)***
State-base	Everywhere	Per-group state in routing table	+ Most efficient B/W usage - *Scalability* - *State in DS core*
Edge-based	Edge routers	None Tunneling	+ Simple + Meets DS requirements - *Dense groups*
Encapsulation-based	Everywhere	Included in packet header	+ Scalable + Meets DS requirements - *Processing cost* - *Per-packet overhead*

2.4.1 State-Based Approach

The per-(source, group) information for each of the multicast (source, group)s is maintained in the routers of the DS domain. When a new receiver wishes to join or leave a multicast group, the core and edge routers in the path (of join or leave message) propagate and update their state information for the specific group. For a join request, the processing for resource reservation is handled at each router until the request reaches the appropriate point in the multicast tree. This approach is the equivalent of simply applying the traditional IP multicast model without distinction to the edge or core routers of the DiffServ domain.

Although very straightforward, this approach contradicts the fundamental concept of simple core routers in the DiffServ architecture. And definitely it has scalability problems because of the complexity of per-(source, group) state maintenance at all core routers. In addition, it also pushes the complexity of multicast protocols onto the core (control messages, resource reservation), thus violating the concept of simple core routers. This approach is viable only if the number of unique multicast group stays extremely small such as currently seen today [27].

2.4.2 Edge-Based Approach

There is another approach that restricts the location of multicast capable routers within the DiffServ domain. It does not allow all routers to perform packet replication and only the edge routers are allowed to be multicast capable and replicate packets as necessary. This approach is simple from the implementation point of view and is highly scalable but incurs performance degradation in terms of bandwidth consumption versus the state-based approach [27]. Other than simplicity, it is suitable for sparse groups where replication of packets at the core routers is less likely to occur compared to that of the dense groups. It is clear that this kind of solution keeps core routers entirely multicast unaware, thus requiring zero support for multicasting (replication or control messages). There is no additional implementation necessary in the core routers and the per-group statelessness of the edge-based approach complies with the DiffServ architecture.

2.4.3 Encapsulation-Based Approach

The next approach is very much different than the previous two - embed the multicast information within the packet itself. In this approach, the tree for the domain is encapsulated within the multicast packet as an extra header. The complete replication information is embedded inside the packet and it does not require per-group state information to be stored in the core. This fits extremely well within the DiffServ architecture. However, encapsulation does introduce several additional costs and overheads. It consumes additional bandwidth for each packet in the form of encoding the multicast tree and thus incurs the scalability concern with the group size. This scalability problem is reduced by encoding only the tree for the domain, not the entire end-to-end multicast tree [27]. The second factor of concern is the additional CPU cost incurred due to header processing that may be alleviated to a great extent through hardware support.

2.5 Existing Solutions and Related Work

Differentiated Services architecture has been proposed principally with unicast in mind that works on a '*push-model*' or is '*sender-driven*'. But multicast service is more a '*receiver-driven*' or '*pull-model service*'. Primarily, the DiffServ in multicast scenario was very much unexplored and all the attention was to the unicast paradigm. Recently there has been some work in this area of DiffServ multicast, which can be found in the references [22, 23, 27, 35-43].

In [22] the issue of DiffServ multicast is addressed that propose a solution to NRS problem through explicit Bandwidth Broker signaling. However, it did not address the issue of per-group state in the core. In an extended work [23] proposed to extend multicast routing table to include DSCP, so that branched traffic from this point downwards gets treatment according to the DSCP value in the multicast routing table (MRT).

DSMCast [27] approach gets rid of any multicast states in core by 'Header Encapsulation' in each packet. It also assumes the Ingress router has complete network and multicast tree knowledge. The complete tree including the required QoS at every downstream node is encapsulated in each packet by Ingress. As the multicast network grows header length also grows and the highly dynamic nature of multicast receivers clearly causes a problem for Ingress to change the 'bulky' DSMCast header information.

In [35] a DiffServ-Aware Multicast (DAM) scheme is proposed which includes three features: Weighted Traffic Conditioning (WTC), Receiver Initiated Marking (RIM) and Heterogeneous DSCP Headers Encapsulation (HDHE). WTC 'counts' the admitted multicast traffic as multiple unicast traffic while conditioning the traffic aggregate at the edge routers. RIM takes care of Heterogeneous QoS requirements and HDHE provides a means to ensure the fairness among the multicast and non-multicast flows. We see that DAM is clearly a sender-driven approach more than a receiver-driven approach. Another problem identified is that the network edge must have the complete network and multicast traffic branching knowledge. Otherwise neither the WTC table can be created nor can the HDHE be performed. Moreover HDHE scheme will suffer from same negative effect of extra header burden and corresponding more bandwidth consumption as in DSMCast.

In [36] another DSM approach is described known as M-DS (Multicast DS). It operates on two inter operable limited branching techniques: Edge Router Branching and Limited Core Branching, as appropriate within each DS domain. Edge Router Branching disallows multicast branching points in the core of any domain and moves all the complexities to the edge of the domain – which is very much conforming to DS architecture. Whereas Limited Core Branching allows a limited number of core routers to act as branching points. Here we find that the core routers have to take part in the signaling activity in Edge Router Branching technique when an on-tree core determines that it is the branching point and communicate that information to BB. Considering the DiffServ proposal in [4] the cores should be free from any kind of control functions. In case of Limited Core Branching the BB searches for special core routers as branching points whose functionalities will be enhanced to be similar to that of Ingress router. This leaves the network core routers to be able to switch to enhanced as Ingress any time needed, since any core can be found to be potential branching point which again makes the core routers complex.

G. Bianchi et al, in [37] describes another novel approach of DSM with QoS consideration called QUASIMODO. It works based on Gauge and Gate Reservation with Independent Probing (GRIP) [37, 38]. The resource availability along a new QoS path is verified via the probe based approach GRIP and then QoS is maintained by marking replicated packets with a special DSCP value before forwarding them on the QoS path. QUASIMODO provides Heterogeneous QoS solutions as inherited from the proposals in [22, 23]. It also differentiates the core routers to be multicast and non-multicast routers based on branching and non-branching points. Only the branching or multicast routers keep MRTs. This approach is built on top of the existing multicast routing protocol (PIM) and allows support for both BE and QoS users as well as upgrading a BE user to a QOS one. Multicast replication and packet forwarding is provided by the PIM protocol operation and stored in MRTs in the multicast routers.

RIMQoS [39] is a receiver-initiated multicast protocol with multiple QoS constraints. It is a source-tree based approach assuming the availability of link-state information and QoS unicast routing protocol. RIMQoS is intended for multicast applications with a single source. It delegates most computing complexities to the receiver node (i.e., router). The receiver router does the route computation and other nodes usually only do some simple message processing. It can also support heterogeneous multicast. The join complexity and related computation on demand depends on the underlying unicast protocol.

In [40] another DiffServ multicast approach called QMD is described where the control and data plane functions are separated. The edge routers do the control plane functions and only a set of on-tree routers (key nodes) maintain multicast routing states and forward multicast data. Here also QMD assumes that the edge routers have the knowledge of domain topology which is not a very favorable assumption in the highly dynamic multicast network especially when the network is large enough. QMD is also sender-driven in the way that the QoS path from the source to destination is computed by the Ingress using a modified Dijkstra algorithm QMD-DIJKSTRA. It also does not specify how an edge router will get the IP address of the other end Ingress router. There are other QoS based multicast routing protocols like [41, 42] but all of them are based on per-flow reservation. All the on-tree routers have to maintain the resource reservation states of multicast groups that make the protocols impractical to be deployed in DS domains.

The literatures in [44 - 48] discuss about the problems and solutions to multicast state scalability and state reduction. In [44] a scheme called Dynamic Tunnel Multicast (DTM) is proposed to eliminate the unnecessary multicast forwarding states. Based on the observation that, when the members of a group are sparsely located, the distribution tree of the group is likely to contain long, un-branched paths – DTM utilizes dynamically established tunnels on un-branched links of the multicast distribution tree. After dynamic tunnels are established, usually only the root node, branching nodes and leave nodes of the original multicast distribution tree need to maintain state information about the group.

The un-branched nodes are bypassed by the tunnel and do not have to know about the group since the packets sent to the group are forwarded in a unicast fashion between tunnel end points.

In [45] the authors have presented a technique that can be used to aggregate multicast forwarding states. They have considered only *perfect* forwarding state aggregation; that is aggregation that does not change the distribution of multicast traffic. An interface-centric data structure model has been presented which allows aggregation of ranges of multicast addresses in the forwarding table and aggregation can be performed independent of the number of interfaces on a router. Depending on the future high end router/switch architectures with parallel processors the authors assume each interface be associated with its own copy of an input filter and an output filter where each filter yields a pass-or-fail answer for a given (source, group). When a packet arrives on an interface it must first pass the input filter. If it passes then the output filter of every other interface is independently checked to verify whether the packet should be sent through that interface. Results yield that aggregatability is significantly higher for clustered receivers. It is greatest on the interfaces which are busiest and hence need it the most.

Aggregated Source Specific Multicast (ASSM) is discussed in [46, 47] for both the legacy network and the DiffServ supported MPLS network. ASSM is targeted to intra-domain multicast provisioning. The key idea is that, instead of constructing a tree for each individual multicast session in the core network (backbone), multiple multicast sessions are forced to share a single aggregated tree. Using this concept of aggregated multicast, [47] proposes Aggregated QoS Multicast (AQoSM) that can support QoS multicast scalably and efficiently in DiffServ supported MPLS networks. Data packets from different groups are multiplexed on the same distribution tree, called aggregated tree and each data packet of each group is encapsulated and travels on the aggregated tree.

The NARADA protocol [48] is a system designed for video distribution. It uses the idea of a mesh to densely connect the videoconference participants. Simultaneously, and algorithm similar to the DVMRP is run and source rooted shortest path (reverse) trees are constructed using the links established in the mesh. Here the end systems implement all multicast related functionality including membership management and packet replication. Here, an end system participating in the multicast group communicates via an overlay structure. It maintains the stateless nature of the network by requiring end systems, which subscribe only to a small number of groups, to perform additional complex processing to any group.

The work in [48] proposes an edge-node multicast overlay to create a tree capable of performing point to multipoint distribution of real-time video with a hard upper bound on end to end delay. The tree source node will iterate through the possible tree topologies and choose the configuration with the highest score where the score is based on the latency and bandwidth of each configuration. By moving the tree configuration, replication and network monitoring to the edge routers instead of end systems, the trees can be constructed using the knowledge of overlay link bandwidths and latencies. All tree construction and content distribution is done without the use of native IP multicast.

2.6 Proposed Solution

Although the existing solutions address different aspects of the conflicts between DiffServ and multicasting, none of them address anything that can make the DiffServ multicasting independent of active multicast (source, group). In the next chapter, this thesis proposes an architecture that addresses the problems of DiffServ and multicasting by making the multicast operation based on topology rather than (source, group) states. Chapters 3 and 4 describe the architecture and provide comparative analysis by simulation.

The architecture, EdgeCast (Edge Based Multicast), proposes a hybrid solution that combines the header encapsulation and internal router states. The egress edge router information that is included in the multicast distribution tree is added as an extension header at the ingress (entrance) to the DiffServ domain. The header - Edge Encapsulation Header (EEH), allows the ingress router to explicitly specify the egress routers that the packets must take to reach the group destinations. A heterogeneous QoS extension allows EdgeCast to use a single tree while still meeting the heterogeneous QoS needs of the egress points in the multicast tree.

The primary strength of this architecture is that it addresses all the conflicts of DiffServ multicast with performance close to that of the state-based approach. However, the key weakness of the approach is that it does introduce additional overhead as well as requiring additional hardware for processing the EdgeCast extension header.

2.7 Summary

This chapter introduced some key aspects of DiffServ multicast. Also an overview of general multicast issues has been provided to identify the DiffServ multicast problems with greater clarity. In the light of the multicast issues the problems of DiffServ multicast have been described. An overview of the methods of providing DiffServ multicast has been given from the general perspective. Later, a brief discussion regarding the related

DiffServ multicast approaches and their pros and cons have been provided. A hybrid approach to DiffServ multicast – EdgeCast, has been introduced as a potential solution to multicast problem in DiffServ domain.

Chapter 3

EDGECAST: A HYBRID APPROACH TO DIFFERENTIATED SERVICES MULTICAST

3.1 Introduction

In this chapter, an approach for DiffServ multicasting, EdgeCast, is proposed which is a combination of state-based and encapsulation-based approaches. The primary goal of EdgeCast is to address the issue of per-(source, group) state information in the core. By utilizing the per-edge (egress) state information instead of per-(source, group) state information in the core, the routers' state scalability problem is made independent of the number of active *(source, group)*. Instead, the number of states is made dependent on active egress edge points. Moreover, the list of egresses is added to the packets passing through the ingress as an extra header that is used to make replication and forwarding decisions. This encapsulated header has also the provision of providing heterogeneous QoS to the multicast group members. Each of these issues is discussed in this chapter along with the basic mechanics of group dynamics (member join/leave).

The rest of this chapter is organized as follows. The section 3.2, describes the fundamental case for a hybrid approach versus the existing solutions. Next, Section 3.3 outlines the components of EdgeCast architecture and details the required modifications and extensions to the current multicast components. The description of how multicast transport service is delivered including details of member join/leave is provided. Finally, Section 3.4 summarizes the entire EdgeCast architecture and its components.

3.2 A Case for Hybrid Approach

Unlike the traditional *(source, group)* based approach of IP multicast, the hybrid approach can solve the state scalability problem by making the multicast states independent of active *(source, group)*. This approach is hybrid because it works on utilizing both the multicast states and an extra encapsulated packet header. The encapsulated header carries the list of egresses involved in the transport of the multicast packets. The packets are replicated and forwarded not based on (source, group) states, rather based on egress edge states. Although this hybrid approach does incur some performance penalties, it solves some severe problems of DiffServ multicast. The benefits of this approach can be summarized as follows:

- *No (source, group) states*: The hybrid approach gets rid of state maintenance based on (source, group) and allows measure of worst case scalability scenario based on domain topology.
- *Hardware support*: The additional processing related to the hybrid EdgeCast can be reduced to substantial level through the addition of hardware support.
- *Resource management*: A central entity like BB is able to easily monitor the resources being consumed by the multicast tree with the aid of the edge routers
- *Heterogeneous QoS support*: Rather than requiring separate trees for each level of QoS, 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.
- *Deployment:* EdgeCast deals with domainwide deployment and instead of requiring end-to-end support; the transport nature of this hybrid approach reduces the scope of

the deployment problem to a domainwide problem whose operation is transparent to the external Internet.

However, there are also several weaknesses to the approach which must also be considered as well.

- *Overhead*: The addition of encapsulated header to the packets incurs extra overhead in bandwidth and processing.
- *Increased search cycles*: The header needs to be searched and matched to the Multicast Forwarding Cache (MFC) multiple times for making the replication and forwarding decisions.
- *Edge Scalability*: Edge routers must maintain per-(source, group) state information. Moreover, the ingress router also has to maintain an extra table for holding egress list per source. Although these routers are better suited for such information, this is a non-trivial problem which is inherent to multicasting.

3.3 EdgeCast – The DiffServ Multicast Approach

The primary goal of the EdgeCast architecture is to address the problem of state scalability. It also solves the problems of heterogeneous QoS support. Although the hybrid approach can address all these issues, the use of encapsulation and replacing *(source, group)* core state with egress states introduce several challenges that must be addressed. To start, the first challenge is to develop an efficient packet header that reflects the multicast tree and is optimized for fast hardware processing. Furthermore, a new domainwide Join/Leave protocol must be designed to support the edge based state maintenance in the core. Since the general trend in multicast is towards SSM, the EdgeCast is designed with a view to supporting SSM.

Hence, it is the motivation of this chapter to address these challenges in the EdgeCast architecture. The architecture can be subdivided into the following components – the Extended Bandwidth Broker (XBB), the egress edge states in the core, Encapsulated Edge Header (EEH), Registration and Query, member dynamics, packet transport, and heterogeneous QoS support. Table 3.1 lists the components and a brief description of their functions.

Table 3.1 EdgeCast System Components and Their Functions

Components	**Functions**
XBB	The XBB takes part in the Registration and Query operations and also it does the QoS verification. It is a central entity that acts something like a "*Domain Manager*".
Edge States	The core routers store the egress edge states and this happens during the join process.
EEH	The EEH is the most important part of EdgeCast and is an extra header that keeps the required egress entries for the packet multicast.
Registration and Query	Through Registration the XBB gets proper knowledge about the source and Query ensures that the source is active and the requested QoS can be met.
Member Dynamics	Member dynamics deals with the new member join and active members leaving the group.
Packet Transport	Packet transport mechanism describes how the packets are replicated and forwarded based on the EEH information in the packet.
Heterogeneous QoS Support	This provides the mechanism to support heterogeneous requirements of different members of a group

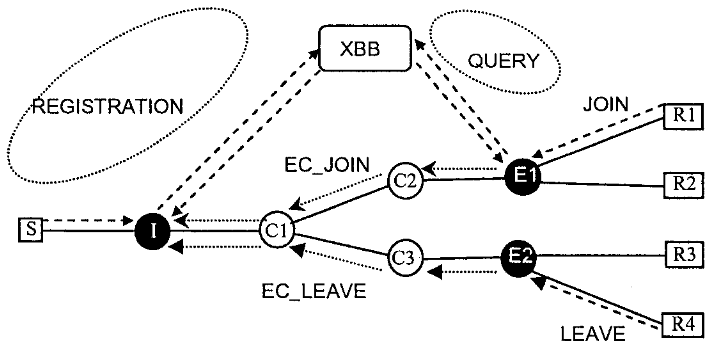


Fig 3.1 EdgeCast System Model

Fig 3.1 shows a schematic of an EdgeCast system model. It consists of a single source (S), an ingress edge router (I), two egress edge routers (Ex), three core routers (Cx), four receivers (Rx) and an XBB. A multicast source first registers itself to the XBB through the registration process. Because of the registration XBB has complete knowledge about the group's source. A receiver willing to join group G (source S) first sends a join request (JOIN). Based on the receiver's request, the egress initiates query operation (QUERY_REQ) and communicates with the XBB. If the source is active and QoS can be guaranteed the egress gets positive acknowledgement (QUERY_ACK) about the query. This initiates the propagation of EdgeCast specific join (EC_JOIN) message towards the source. This EC_JOIN message creates proper egress states in the core routers. The data from source first enters the ingress (I). It performs the EEH construction and traffic conditioning, policing and marking. All the passing packets get their proper EEHs inserted by the ingress. When a packet reaches the core, its EEH is checked for the entries in the core MFCs. Based on this lookup the replication decision is made and packets are forwarded through the proper next hop interfaces. This is how the packets are multicast through the core. When a packet reaches the egress the first thing done is to remove the EEH header from the packet. Then it is forwarded to the proper receivers based on its (source, group) information.

We provide the details of each individual component in the next few sections.

3.3.1 Extended Bandwidth Broker

DiffServ architecture was proposed to provide QoS to the Internet traffic. To provide QoS the prior reservation of network resources is a major concern. If required resources cannot be reserved for a traffic session then QoS cannot be guaranteed. To manage resource reservation in the DiffServ network the Bandwidth Broker (BB) has been introduced [50, 51]. BB can be configured with organizational policies, keeps track of the current allocation of marked traffic, and interprets new requests to mark traffic in light of the policies and current allocation. A BB has a policy database that keeps the information on who can do what when and a method of using that database to authenticate requesters.

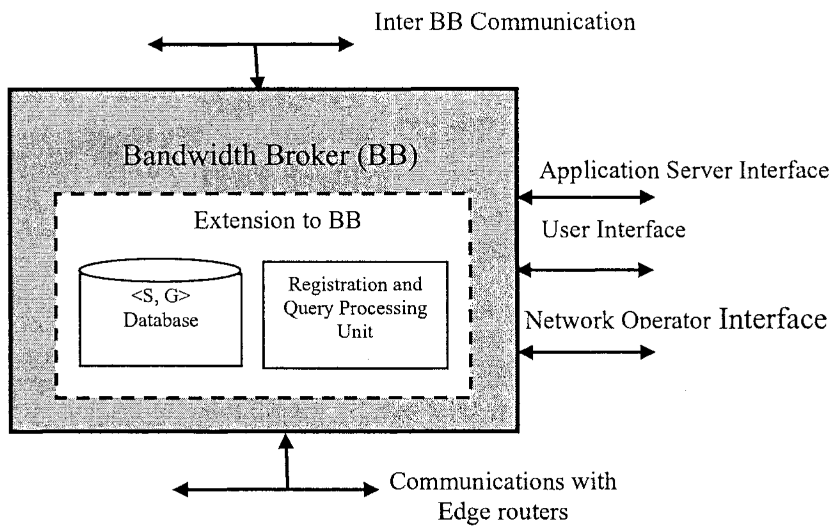

Inter BB Communication
Bandwidth Broker (BB)
Extension to BB
<S, G> Database
Registration and Query Processing Unit
Application Server Interface
User Interface
Network Operator Interface
Communications with Edge routers

Fig 3.2 Extended Bandwidth Broker Architecture with storage and processing unit

In the EdgeCast it is assumed that BB will handle the resource management and so the core BB functionality is not discussed here. We propose an extension to the BB architecture and functionality to handle the '*Registration*' and '*Query*' operations. Fig 3.2 shows the architecture of proposed Extended Bandwidth Broker (XBB). Here only the parts inside the dashed rectangle are extended as compared to the BB architecture figured in [51]. The extension includes the 'Registration and Query processing unit' and a data structure to store the <S, G> information for each set of <S, G> registration. Assuming there is a single source per group in the domain, there is no need to specify the source in the join request, only the group and QoS are specified. This database of XBB can readily supply the address of the source corresponding to the multicast group. Another reason for source registration is the centralized security. A receiver will receive data only from the authentic source registered with the XBB so that no malicious source sends data to the group members. The receiver sends JOIN request to the egress and the egress initiates query process with the XBB for the source authentication and QoS verification. The XBB performs the desired authentication and/or supplies the source address (in case of single source per group) and QoS admission notification after performing necessary processing.

3.3.2 Egress States in Core Routers

EdgeCast – as the name specifies, utilizes the edge (egress to be more specific) points of the multicast distribution tree to deliver multicast packets to the group receivers. In SSM the active *(source, group)* or *(S, G)* pair is termed as an individual channel. The network layer service provided by SSM is a "*channel*", identified by an SSM destination IP address (G) and a source IP address (S). An IPv4 address range has been reserved by Internet Assigned Numbers Authority (IANA) [52] for use by the SSM service. The channel subscription is supported by version 3 of the IGMP protocol [53] for IPv4. SSM provides receiver applications with a "channel" abstraction, in which each channel has exactly one source and any number of receivers. When a receiver subscribes to an *(S, G)* channel, it receives data sent only by the source S.

The network layer service provided by EdgeCast is by the egress edge value corresponding to the end point edge router of the tree. This edge value can be the part of the original IP address of the edge or an assigned *ID* value corresponding to an edge. If *ID* is used then there must be a system to map the unique *ID* values to the corresponding edges. And the XBB can be used to centrally manage that *ID* mapping for a domain. This thesis assumes the last 16 bits of the IP address is used for edge value. In DiffServ, we know that the edge routers are classified into ingress and egress, based on the direction of data flow. All the traffic generated from source must pass through the ingress and to reach the group destinations the packets must pass through the egress routers.

To accomplish the multicast data transport based on egress values, it is required to store the egress entries in the routers that are part of the multicast distribution tree. The states are stored the same way as the SSM *(S, G)* states are stored. During join process the JOIN message carries the associated egress/*ID* value from egress to the source. This value is compared to the existing entries in the on-tree routers and stored if necessary. A major difference between the traditional join process (JOIN) and the EdgeCast join process (EC_JOIN) is that the process does not stop at the branching node (the node already having the specific *(S, G)* entry); rather it is transported to the ingress connected to the multicast source. The necessity of this procedure will be justified later. Although the very first join corresponding to a particular egress must reach the source.

EdgeCast proposes an extension to the functionality of BB so that the BB is capable of identifying the correct source associated with the new join to the multicast *(S,G)*. EdgeCast currently deals with intra-domain situation and the inter-domain case is out of the scope of this thesis. Towards the source the EC_JOIN message makes egress entry (if required) in the routers. Thus an egress entry in the router utilizes a single address space of 16 bits instead of double address space required for SSM *(S,G)* entry – 64 bits for IPv4 and 256 bits for IPv6. This helps to save total memory required to store the same number of egress states than *(S,G)* states in the routers. When an incoming EC_JOIN message initiates a new egress entry it also stores the outgoing interface (OIF) corresponding to the join's incoming interface (IIF) – same as the traditional approach.

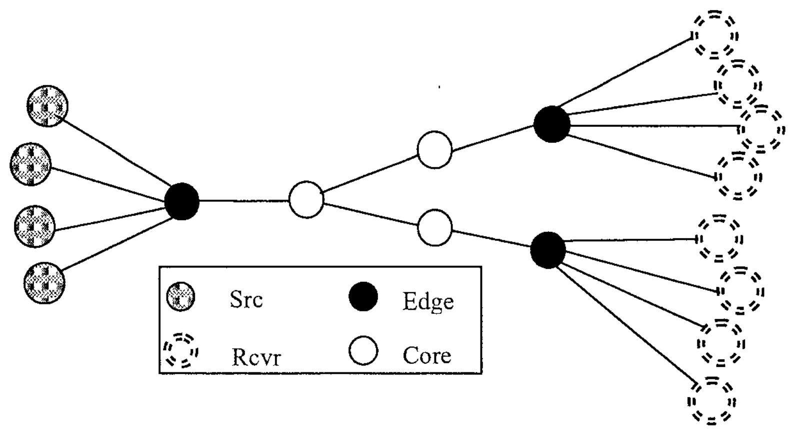


Fig 3.3 A DiffServ multicast network with 4 sources/groups and 8 members

Let us consider the network of Fig 3.3. Here, an ingress is connected to four sources and each of two egresses is connected to four receivers. Let four sources serve four individual groups. If four receivers of either egress join four groups, using SSM will require the first core router after the ingress to store four different entries corresponding to four individual (source, group) combinations. The other two cores, each will also need to store four entries since data of four (source, group)s pass through each of them. On the other way it is also clear that the multicast data basically utilize two egress routers from the ingress. In that case using EdgeCast will require the first core router to store only two entries corresponding to the two egresses. And the other two cores, each will use a single entry of the egress it is connected to. The states also need to be refreshed after some specific timeout period. But we have not considered this in the thesis and is left as a matter of future research.

3.3.3 Encapsulated Edge Header

The previous section described the router states that are used to replicate and forward multicast packets in the domain. But there must be some kind of transport mechanism that utilizes the states. For SSM, the packets are transported by comparing the *(source, group)* address in the IP header to the *(S, G)* entries in the MFC. Unlike SSM, EdgeCast packet forwarding is based on egress edge entries. Within the IP header there is no room to accommodate the extra egress entries. Hence EdgeCast provides that transport mechanism by adding an extra header to each of the packets that pass through the ingress. That special header is the *EdgeCast Encapsulated Header* (EEH). It carries the list of all the egress edges associated to the multicast packet distribution tree.

A multicast packet may need to be delivered to 10 different receivers and the they may be connected to 5 different egress edges. In that case there will be 5 edge entries associated to the packet. Now the question arises about the location of the EEH. One choice could be utilizing the IP '*Options*' field. But IP '*Options*' are a rare occurrence [27] and makes the processing slower. Another choice is in between layer 3 and layer 4 headers. This location for header insertion has been suggested in [27] and solves the slow processing problem related to using '*Options*'. The ingress router looks up the IP header for the specific multicast group *(S, G)* and inserts the EEH between the layer 3 (IP) and layer 4 (UDP, etc.) headers of the packet. This location is chosen as it preserves the standard IP routing and DiffServ classification. The EEH is only added at the ingress of the domain and removed at the egress and it carries all the associated egress from the ingress to the appropriate egresses through the domain.

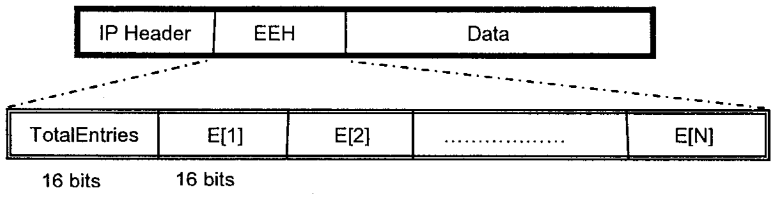


Fig 3.4 EdgeCast Edge Encapsulation Header (EEH)

The EEH is the sequence of bits that are responsible for routing and replicating the packet across the core of the DiffServ domain. The makeup of these bits is faced with two potentially conflicting goals: optimal bandwidth usage versus processing speed.

The EEH consists of two types of fields - the *TotalEntries* field and the *E[i]* field. The fields are a fixed length in order to allow for easier processing by the hardware mechanism. Fig 3.4 shows the layout of the EdgeCast EEH. The first field, the *TotalEntries* field, is a 16 bit field that gives the total number of egress entries present in the EEH itself. For each different egress involved in the multicast distribution tree, an entry will be present. The next portion of the EEH is the egress edge information. For now there is no *'option'* field present in the EEH configuration. EdgeCast assumes a space of 16 bits for each egress entry should be enough to handle the total number of edge entries in a single DiffServ domain. This 16 bit space can support maximum 2^{16} edge/ID values. For now, the information required to support heterogeneous QoS is not considered.

3.3.4 Registration

The successful operation of EdgeCast also includes 'Registration' of the sources. Sources perform the registration operation with the help of ingress router and the XBB. A node acting as a multicast source first registers itself as a '*Source*' to a multicast '*Group*'. The steps are as follows:

- Source sends a REGISTER_REQ <S,G> message to the ingress router.
- The ingress forwards this request to the XBB and an entry of <S,G> is made in the XBB database. This process ensures that S is a valid and active multicast source S for group G.

- After successful registration process the XBB sends acknowledgement REGISTER_ACK<S,G> to notify the requesting ingress of its <S,G> registration.
- After receiving REGISTER_ACK<S,G> the ingress will start a timer associated with the <S,G> pair. We consider this process from ingress to XBB to be a *'soft process'* to ensure that the <S,G> information is properly updated.
- When the corresponding timer expires, the REGISTER<S,G> message is again sent to the XBB. What would be the optimum duration of the timer is a matter of future research and is not considered in this thesis.
- When a source no longer wants to remain a multicast source, it sends a UNREGISTER<S,G> message to ingress. Ingress notifies XBB of that action and then both XBB and ingress update their databases.

3.3.5 Query

Receivers perform the 'Query' operation with the help of egress router and the XBB. This makes the coordination much better among different EdgeCast network components and also helps to maintain multicast security and QoS verification in a centralized fashion. Query operation is invoked when a receiver sends a request to join a multicast group. The egress router that receives the join request invokes the Query operation for this join. Since EdgeCast has been proposed to support QoS, there must be a way to specify the QoS requests both in the join and the query requests. In other words, the QoS signaling has to be specified. A look at the current protocols reveals that specifying QoS by the JOIN message is a non-trivial problem. If we consider PIM then the member join request is specified by the IGMP Report message. The IGMP Report message can specify the *Source* and *Group* the receiver wishes to join. But there is no way of specifying the

QoS request in the JOIN message. We propose an extension to the original IGMP Report message. A look at the Report message format reveals that there is enough '*reserved*' space in the message which is currently unused and are ignored by the routers. This '*reserved*' space is a good place where the QoS request can be embedded. This QoS information will be used by the EdgeCast EC_QUERY message as explained later in this section.

Another way is using RSVP to specify QoS. But RSVP is mainly used with IntServ. Using IntServ/RSVP is not regarded a good choice for internetwide deployment because of IntServ per-flow specification. Moreover, end-to-end resource reservation by RSVP is not feasible enough for ever expanding internet. Rather, the research community favors keeping IntServ/RSVP as stub domains and DiffServ as the core domain. This QoS information within RSVP messages will be mapped into the EdgeCast Query message.

In either of the cases as a choice for join, the egress router performs the required mapping and transformation by extracting the *Source, Group, QoS* information from the original JOIN message. The egress later sends the EC_JOIN message towards the source including *Source, Group, QoS* and *Egress identity.*

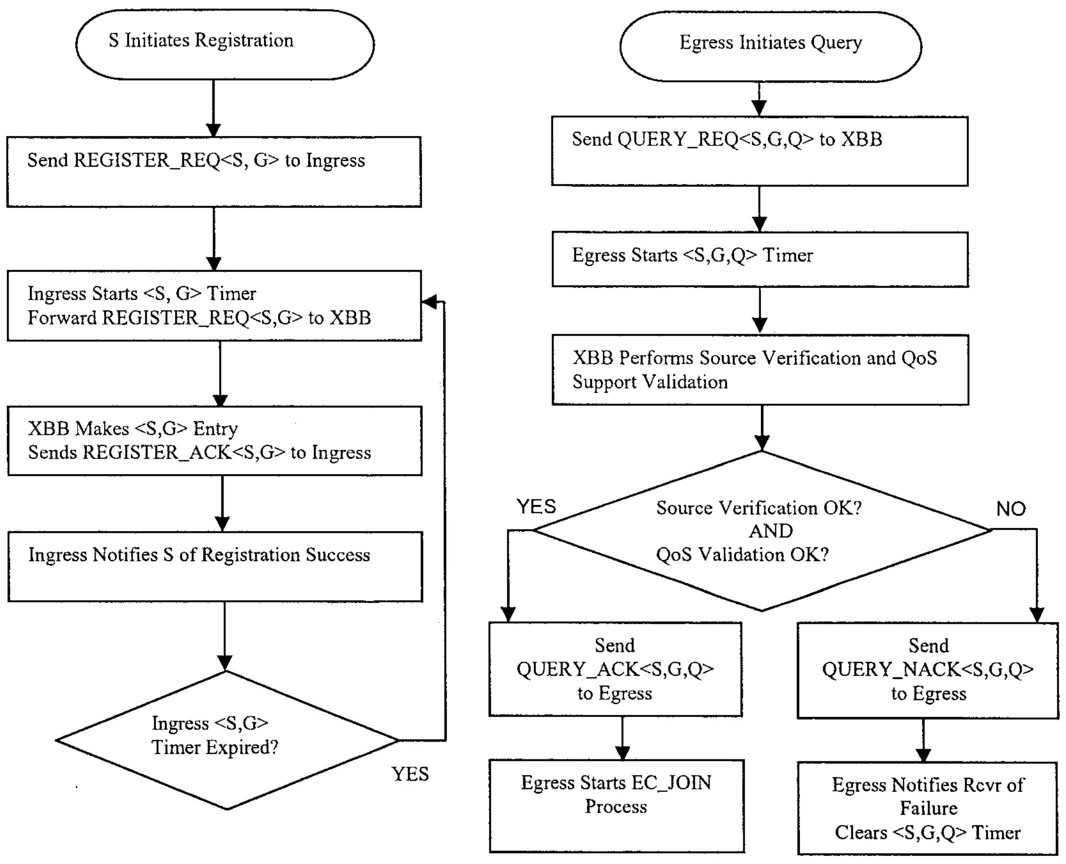
The egress then invokes the Query process. The query steps are as follows:

- A receiver willing to join a group with a specific QoS sends a JOIN request with the source S, group G and QoS Q to the egress.
- The egress then starts a timer corresponding to this request and sends a QUERY_REQ<S,G,Q> message to the XBB for approval. These parameters are the values extracted from the PIM JOIN request.

- The XBB then performs the *(S,G)* entry lookup corresponding to the request.
- If the source is 'active' and QoS can be guaranteed then the XBB sends a QUERY_ACK<S,G,Q> message to the requesting egress notifying the positive feedback. Otherwise, the XBB sends a QUERY_NACK<S,G,Q> to the egress notifying the negative feedback about the join request.
- If egress receives a positive feedback then the EC_JOIN message is sent towards the source S. If negative feedback is received then the egress notifies the receiver of its inability to make the join for the time being. The EdgeCast join process is described later in detail.

For SSM kind of join request it may seem redundant to check and verify the source in the XBB since source address is already known to the receiver. But this query process takes care of another problem in multicasting – to verify whether the source is active at that instance. Sometimes it is a problem of multicast that the receiver asks for joining a source that may not be '*active*' at that time. Due to the '*soft*' registration process the status of the source is always updated to the XBB. So, a positive acknowledgement from the XBB also confirms that the source is active. For the ASM-like join, this query process also solves the '*source anonymity problem*' without the help of other external protocols.

Fig 3.5 shows the flowcharts of 'Registration' and 'Query' operations. Also, Table 3.2 lists the related messages and their parameters for these operations.


S Initiates Registration
Send REGISTER_REQ<S, G> to Ingress
Ingress Starts <S, G> Timer
Forward REGISTER_REQ<S,G> to XBB
XBB Makes <S,G> Entry
Sends REGISTER_ACK<S,G> to Ingress
Ingress Notifies S of Registration Success
Ingress <S,G> Timer Expired?
YES
Egress Initiates Query
Send QUERY_REQ<S,G,Q> to XBB
Egress Starts <S,G,Q> Timer
XBB Performs Source Verification and QoS Support Validation
Source Verification OK? AND QoS Validation OK?
YES
NO
Send QUERY_ACK<S,G,Q> to Egress
Send QUERY_NACK<S,G,Q> to Egress
Egress Starts EC_JOIN Process
Egress Notifies Rcvr of Failure
Clears <S,G,Q> Timer

(a) (b)

Fig 3.5 (a) Registration and (b) Query Operations Flowchart

Table 3.2 Registration and Query Messages and Their Parameters

Message	Parameters
REGISTER_REQ	Source, Group
REGISTER_ACK	Source, Group
UNREGISTER	Source, Group
QUERY_REQ	Source, Group, QoS
QUERY_ACK	Source, Group, QoS
QUERY_NACK	Source, Group, QoS

3.3.6 Member Dynamics

Unlike DVMRP, PIM-DM there is no periodic flood-and-prune in the EdgeCast network. Unless explicitly a join has been established there is no data flow. And, a receiver stops getting multicast traffic after explicitly leaving a group. This keeps the network free from unnecessary flow of traffic and corresponding network management.

After providing the underlying transport service the next step is to solve the problem of how join/leave operations are performed. Whereas the *(S,G)* state-based approach simply relied on individual nodes propagating JOIN/LEAVE messages towards the multicast tree, the replacement of *(S,G)* state information with egress state information makes the control packet processing a bit different. The multicast group may be present anywhere in the global Internet, thus necessitating a search mechanism for the multicast group. Several schemes such as MSDP (Multicast Source Discovery Protocol [56]), MBGP (Multicast Border Gateway Protocol) [57, 58] have been developed to address this. EdgeCast is primarily concerned with source specific model of multicast. As EdgeCast relies on XBB for *(S, G)* authentication, it can also act as a source address provider to the join process. In that case the operation can be done without the help of MSDP etc. When a receiver sends a JOIN request the egress performs the '*query*' operation with the XBB to reveal the source's '*status*'. It may happen that the source is not active at the time of join request. In that case the join operation is discarded although there is a source for the requested group – but inactive.

Member Join

A JOIN request is sent to the egress edge router by the receiver, as usual, including a source/group identifier *(S,G)*. The egress router must then forward the packet to the appropriate ingress node for multicast source. Hence the join problem can be summarized as follows:

For a join to source S_x and group G_x from node R_y that arrives at egress edge (E_{Egress}), forward the join towards the S_x so that the QoS constraint of the request is met.

Since EdgeCast works based on egress edge value - not based on *(S,G)*, there must be a way to allocate the egress to the message for join. If only the edge address (partly) is used then there is no problem, but if a unique ID per edge model is used, the XBB is the best possible entity to allocate and manage this edge-to-ID matter. And moreover, since DiffServ is developed for supporting QoS, the join request first must go through a verification process to find out whether the requested QoS level can be supported. Here it is assumed that the XBB is capable of deciding on whether the requested QoS can be guaranteed. How this is done, is out of the scope of this thesis.

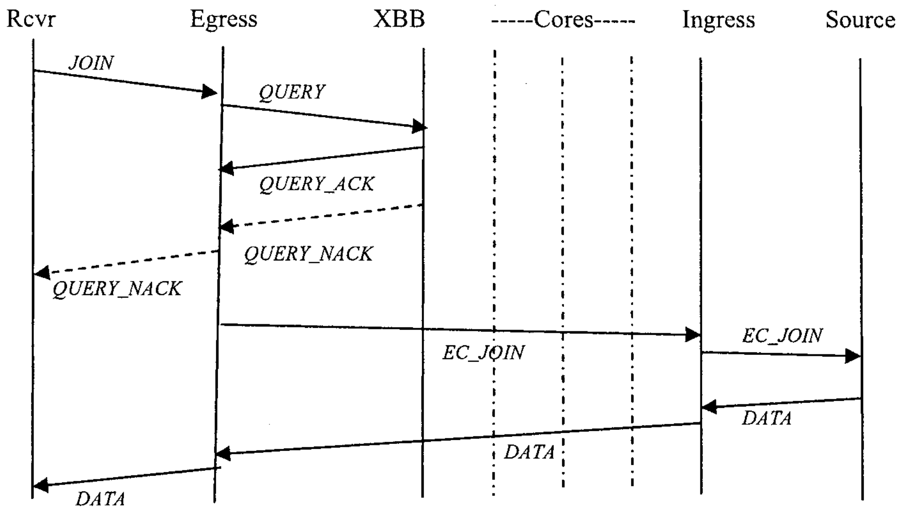


Fig 3.6 EC_JOIN Protocol Timeline

Before sending the EdgeCast join - EC_JOIN, the egress initiates a '*query*' process by sending a QUERY_REQ message including the *(S,G)* and the requested QoS. The XBB then performs multiple tasks like QoS verification, verifying the source address and status (active/inactive), and assigning a unique edge ID for the join (used only if the edge-to-ID assignment model is used and QoS verification turns positive). The XBB sends a QUERY_ACK<S,G,Q> response to the egress. Otherwise it sends a QUERY_NACK<S,G,Q> indicating that the requested join can not be made now. In such a negative response the egress may decide either to join the group with the default BE or abandon the join for the time being. Here it is assumed that the EC_JOIN message is only initiated if the egress receives positive acknowledgement from the BB, otherwise the join request is discarded. This EC_JOIN message is different than the original JOIN received by the egress. The egress extracts the required information from the original JOIN and creates EdgeCast specific join message, EC_JOIN, which includes the egress identity.

The next hop core router receives the EC_JOIN and checks the MFC with the egress identity carried by the message. If this is a new egress intercepted by the router it makes an entry in the MFC and the OIF of the link through which the EC_JOIN message was received. If there is already an entry in the MFC, the router does not discard the message like the traditional multicast, rather it forwards this towards the ingress. This is a major difference between the SSM/Anycast JOIN and the EdgeCast EC_JOIN. This message must reach at least the ingress – does not matter whether the intermediate nodes already have the entry. This is required because ingress needs to know all the egress points involved in the multicast packet distribution. When the ingress receives the EC_JOIN it makes the entry in its table of egress corresponding to each source. Thus ingress collects the complete egress information of multicast distribution.

If the ingress finds that the source corresponding to the new join is already active, it stops the message at that point and discards it. On the other hand if this is the very first EC_JOIN message for a specific source, after making the proper egress entries the ingress forwards the message to the source itself. Only after receiving such an explicit join request the source starts to transmit data.

Member Leave

Like a join request, when a leave request arrives at an egress router, it will include a source/group identifier. The egress router must then forward this control packet to the appropriate ingress node that is connected to the multicast source. The '*Leave*' problem can be summarized as follows:

For a leave from source S_x and group G_x received from node R_y that arrives at egress edge node (E_{Egress}), update the appropriate core and edge router databases so that the leaving receiver does not receive the packets destined for the (S_x, G_x) address.

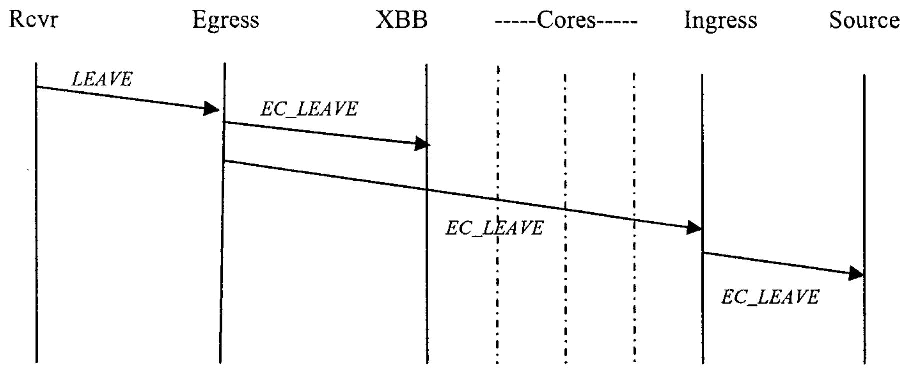


Fig 3.7 EC_LEAVE Protocol Timeline

The egress, after receiving the LEAVE request for an *(S,G)* first checks if this is the last receiver connected to it. If true then there is no need for sending any multicast packet to that egress and the egress deletes the specific (S,G) entry from its forwarding table. Moreover, the leave message must be sent towards the ingress of the source so that the ingress database can be updated for not including the specific edge to the consecutive passing packets. Like the JOIN message, the egress also sends a modified EdgeCast specific leave message - EC_LEAVE which carries egress identity. The EC_LEAVE message also carries the *(S,G)* entries that would leave the group. Although the core routers need only the edge value to make any update to their forwarding cache, the *(S,G)* entries will be required by the ingress to update its database. On the way to the ingress the EC_LEAVE message deletes the entries of the specific egress from the routers' forwarding tables. To make sure that the consecutive packets through ingress does not include the leaving egress, it is required that the ingress updates its database of egress entries corresponding to each *(S,G)*. When the ingress receives the EC_LEAVE, it checks the database for the corresponding *(S,G)* entry that has been carried by the EC_LEAVE and deletes the leaving egress' entry from the database. This makes sure that the consecutive packets for that specific *(S,G)* do not go to the leaving egress point. Moreover, if the ingress finds that this was the only egress point involved to the specific *(S,G)* then it deletes all the information regarding that *(S,G)* and makes sure that the source stops sending data to that group.

There is another situation where the leave request comes to the egress which serves as the egress point to more than one *(S,G)*s. In that case the egress updates its local forwarding table by deleting appropriate information regarding the leaving *(S,G)*. If this was not the last receiver of the specific *(S,G)* connected to the egress then there is no need to send the EC_LEAVE message upwards. Otherwise, the EC_LEAVE message needs to be forwarded towards the ingress to update the ingress table. But, in this case the core routers do not make any changes to their forwarding table. This is because in this case more than one *(S,G)* has been using the same egress to multicast their packets to the group members. So, only the ingress database needs to be updated so that packets from the specific source do not receive the leaving egress' entry in their attached EEH. Thus it is made sure that the egress that has requested to leave the (S,G) does not receive the packets destined for (S,G).

The leave process is a bit more complex then the traditional multicast leave operation. But this is essential for the EdgeCast operation. Like the join process the leave process also must reach the ingress for making the process complete (except only a single case when multiple receivers are receiving data from a specific *(S, G)* and are connected to the same egress).

Table 3.3 lists the messages and parameters for 'Join' and 'Leave' operations.

Table 3.3 EC_JOIN and EC_LEAVE Messages and Their Parameters

Messages	Parameters
EC_JOIN	Edge Address/ID, Source, Group, QoS
EC_LEAVE	Edge Address/ID, Source, Group

3.3.7 Packet Transport

The join process has set up the necessary condition for multicast packet replication and forwarding throughout the tree. After the EC_JOIN message has been received by the source, it starts to transmit packets. The first hop router for the packets is the ingress router – as per the DiffServ architecture. In addition to the packet classification, marking, policing, the ingress router in the EdgeCast architecture, must also perform the proper EEH construction and inserting the EEH at the appropriate location of IP packet.

The ingress forwards the packet to the next hop and this forwarding is done based on egress values - not based on *(S, G)*. The core router checks the EEH entries and does the MFC lookup to identify the set of OIFs through which the packet is to be replicated and forwarded. But this MFC lookup is different from the traditional multicast MFC lookup. Instead of a single round of lookup there will be multiple rounds on MFC lookup, since there are a number of egress entries in the EEH. This is one of the major disadvantages of the EdgeCast. By employing some kind of heuristics in the search process and with the aid of better and faster hardware this problem can be substantially reduced. For processing packets in the network core EdgeCast assumes that there will be a hardware mechanism in the routers that will deal with the EEH processing [27]. Alternatively, such support may be offered in software, but this will incur significant performance penalty. Once replicated, the packet will be placed in the standard DiffServ unicast class queues and treated according to the PHB specified by the DSCP. Once given to the unicast queue, the multicast packet is treated identically to all other unicast packets at the router. Another point is that the identified set of OIFs may result in unnecessary duplication, resulting in multiple copies of the same packet being forwarded through the same outgoing link. The following example helps to clarify the matter:

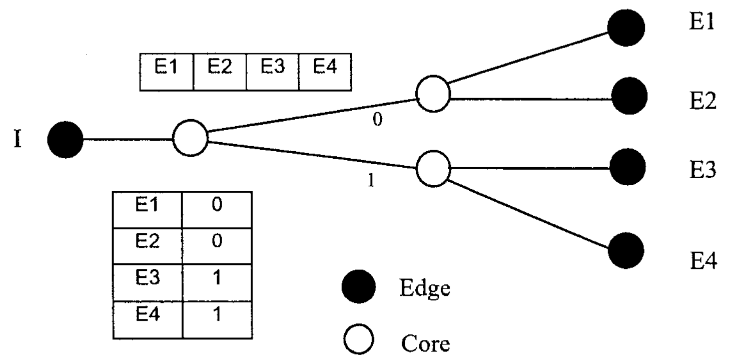


Fig 3.8 Redundant Packet Replication Situation

Fig 3.8 shows a network with four egress routers, one ingress router and three core routers. Let us assume that receivers connected to the four egresses joined a group whose source is connected to the ingress (I). Now the EEH inserted to the packets include the entries for the four egresses. The MFC in the first core router after the ingress has entries for all the egresses with the corresponding OIF entries is shown in the figure. Here only two OIFs are involved and only two copies of the original packet should be forwarded to the next hops. Since the EEH has entries for four egresses the EEH processing leads to four potential copies of the packet. But a look at the MFC reveals that actually same OIF has been entered twice and if packet is copied only based on egress entry there will be two redundant copies. So, only two copies are needed, not four and the replication decision needs to be taken also based on OIF entries.

To solve this problem the set of potential edges will be filtered based on OIFs so that there will be one edge entry per OIF. The packet is replicated and the copies are forwarded based on the filtered list so that the redundant packet replication can be avoided. This section considered only the homogeneous QoS situation where all the replicated packets get the same PHB because packets are replicated with the same DSCP value. The case for heterogeneous QoS support and the matter of DSCP replacement will be considered later. It is also assumed that the whole domain that implements EdgeCast must have all the routers capable of EdgeCast because of different routing mechanism than the traditional approach. A combination of capable and non-capable routers in the EdgeCast is not considered here.

When the packet reaches the egress, as per the DiffServ architecture, the egress will perform the SLA conformance test before sending the packet out to the destination. Additionally, the EdgeCast requires that the egress removes the EEH from the packet. The EEH is needed to transport packet only till the egress point. To reach the destinations after the egress, the packets are forwarded based on the traditional *(S,G)* entries. That is why the egress keeps the MFC like traditional multicast with *(S,G)* when the join process takes place. And for this reason EdgeCast also keeps all the original packet information including the source and group addresses, although the packet does not require them for core transport and multicast.

When the proportion of active (source, group) is much more than the network edges, it is evident that EdgeCast will not let the state explosion to take place, since whatever the number of active *(source, group)* in the domain, the states will always have a pre-calculated maximum value that is bounded by the number of edges. Thus EdgeCast provides the network administrators the advantage of having a much clear idea about the worst case situation in the domain.

3.3.8 Heterogeneous QoS Support

As shown in Fig 3.9, the sender-driven approach of DiffServ has problems with heterogeneous QoS requirements for receivers. Let us consider the case of egress nodes (receivers) R1 and R2. Whereas R1 is requesting EF (Expedited Forwarding) level service, R2 is requesting only (Assured Forwarding) AF11 level service. Since both receivers share the same path to the ingress node, the difference must be resolved. Without the option for dynamic DSCPs in the DiffServ domain, the simple solution would be to send separate packets for each DSCP in the group.

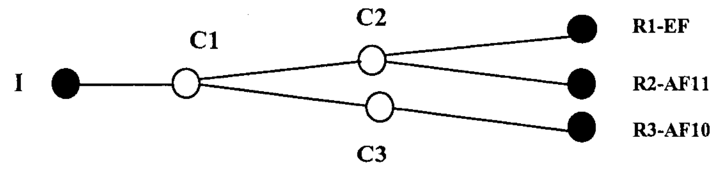


Fig 3.9 Heterogeneous QoS Requirements in DiffServ Domain

Depending upon the relative heterogeneity of members and the locations of such members within groups in the DS domain, this solution may waste excessive amount of network bandwidth. In contrast, a second solution is to require all receivers for the group (egress nodes) in the domain to use the highest DSCP for the group. This solution suffers from two problems in that the solution unnecessarily consumes resources and that all of the receivers in the group may not be willing to pay for the highest level of QoS. From the perspective of the service provider, the only cost of adding R2 should be the last link to R2, provided that R1 is willing to pay the full cost for EF service from I to R1. As long as all upstream links from a receiver meet the necessary QoS of the receiver, the cost of adding additional receivers for the group which share those same links is zero for the shared links (from I to C2). Thus, the optimal solution is to introduce a scheme for multicast whereby the DSCP is allowed to change in the network and adapt to the needs of the downstream receiver.

3.3.8.1 Heterogeneous QoS Solutions

There have been two approaches of Heterogeneous QoS solutions: by extending traditional IP multicast and by encapsulated header in DSMCast

Extending Traditional IP Multicast

In [22, 23], the authors presented a scheme extending traditional IP multicast to support heterogeneous QoS in a DS domain. In order to support heterogeneous QoS, each multicast routing entry would also include a DSCP to be used when replicating traffic for the given group onto the specified link. The work specifically provides support for two levels of QoS, the current group QoS and a LE (Limited Effort) PHB [21] for those that do not wish to use the codepoint of the group. However, this work suffered from several notable problems. First, the issue of upstream PHB conflicts for full heterogeneous support was left as a management issue. The underlying architecture of traditional IP multicast (i.e. per-group information embedded in core routers) makes the issue of upstream PHB resolution non-trivial. Unlike DSMCast where the entire tree is known for PHB resolution, PHB resolution must take place at each router along the tree.

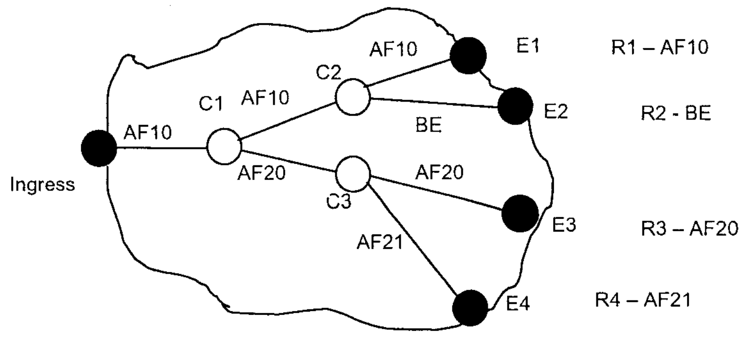


Fig 3.10 Example of Heterogeneous QoS in a DiffServ Domain with Dynamic PHB

For example during a multicast join, one must consider not only the issue of when PHBs should be propagated but also necessary reservation/communication with the BB as the PHB propagates up the tree. In addition, a past history must be maintained of downstream links in order to appropriately degrade the service if the higher classed node leaves the QoS group. In addition, let us consider the network of Fig 3.10 where R4 joins the group first. In this case, the tree has a codepoint of AF21 from edge to edge. Now, if R3 join in succession, a priority problem occurs. If the core nodes along the tree do not propagate the `better PHB' QoS up the tree, the downstream nodes will not be receiving the QoS that they requested. For the case of R3 which is requesting AF20 service, it would actually receive QoS of AF21-AF21-AF20 as the packets go across the multicast tree if the priority is not propagated upstream. Although the order may not always be the worst case, this simple example shows the necessity of propagating the best PHBs upstream.

The propagation of higher priority PHBs introduces the second issue with traditional IP multicasting with PHBs. For each case where the PHB must be propagated, the distributed nature of knowledge of the multicast tree requires that the core router must communicate with the BB. In addition to violating the DiffServ principle of simple core routers, such a scheme also creates the potential for fairly inefficient resource allocation due to excessive message passing.

DSMCast and Heterogeneous QoS

DSMCast [27] uses dynamic PHBs where the core routers are kept stateless and the propagation of `better PHBs' is automatically taken care of at the ingress router. This prioritization is done with complete knowledge of the multicast tree by the edge routers. As a result of the centralized location for a given group, only a single message is required from the ingress router to the BB. For cases where the traffic allocation may fail (insufficient resources), the edge-based approach prevents unnecessary resource allocations. Most importantly, the scheme removes the complexity of such operations from the core, thus maintaining the simplicity outlined by DiffServ architecture. Whereas DiffServ employs a coarse-grained QoS (i.e. the PHB does not change as the packet traverses the domain), DSMCast could also allow unicast packets to use dynamic PHBs as they cross the domain.

Heterogeneous QoS Support in EdgeCast

The core routers' MFCs do not keep the DSCP values for the downstream branches. The MFC entries are not fined granular since more than one group can use the same egress entries. Hence, the DSCP requirements must be carried by the EEH. The EC_JOIN message carries the requested DSCP to the ingress and then the ingress inserts that DSCP value next to the corresponding egress entry. When a packet reaches a core router the EEH is inspected by the core hardware and replication and forwarding decisions are also made. When the final set of egresses is selected for replication their corresponding QoS values are also inspected. When the replication decision is made, the original DSCP of the packet is compared to the requested DSCP. Any mismatch prompts the DSCP replacement according to the requested DSCP.

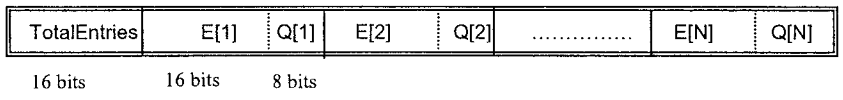


Fig 3.11 EEH with Requested QoS Values

This strategy of providing Heterogeneous QoS support requires a modification to the original EEH architecture. Now a new field *Q[i]* (to store the requested QoS by the join) is added to the EEH and each of these *Q[i]*s follows the location of corresponding *E[i]*. Fig 3.11 shows the new EEH architecture. The question may arise about the size of the *Q[i]* field to be used in EEH. The DiffServ proposal utilizes the 6 bits of the IPv4 TOS field and keeps the rest two bits currently unused (CU). These 6 bits allow 2^6 different DSCP values to be assigned. Conforming to the DiffServ proposal EdgeCast also utilizes 6 bits to store DSCP requests per edge entry. But this field takes 8 bits and two bits of them are kept 'currently unused' like the original DiffServ proposal. Here, these two 'extra' bits also serve another important purpose - each entry is forced to byte boundaries to ensure ease of hardware support.

Although this new EEH structure provides the Heterogeneous QoS support, it increases the EEH size and hence bandwidth overhead because the DSCP information now has to be put inside the EEH.

3.4 Summary

In this chapter the EdgeCast architecture - a new architecture for supporting DiffServ multicast, has been proposed. Each of the different components of EdgeCast related to DiffServ multicast has been described. New protocols for member dynamics have also been introduced. Despite the fact that this architecture incurs per-packet overhead due to the extra header (EEH), it makes the multicast states independent of active (source, group) in the domain. Thus the administrator need not worry about the number of active (source, group)s, rather the edge entries of the domain are capable of handling any number of active (source, groups). EdgeCast also is capable of providing Heterogeneous QoS to the group members.

Chapter 4

THEORETICAL AND SIMULATION STUDIES

4.1 Introduction

In this chapter the performance of EdgeCast is analyzed from both the theoretical as well as simulation perspectives. In the theoretical analysis, EdgeCast is examined for its overhead versus the encapsulation-based DSMCast approach, and multicast states versus the state based approach like SSM. The simulation studies examined the performance of EdgeCast versus the other two approaches (encapsulation-based and state-based) regarding a variety of parameters including active group numbers, header overhead, core routers' states, packet loss and receiver heterogeneity.

The rest of this chapter is organized as follows. Section 4.2 discusses EdgeCast parameters from the theoretical perspective. The header overhead, the multicast states and the message complexity have been examined. Section 4.3 presents the results of different simulation studies and provides the necessary explanation to the results. Finally, Section 4.4 outlines the summary of the chapter.

4.2 Theoretical Studies

4.2.1 Packet Header Overhead

The EdgeCast EEH incurs overhead to the packets that are to be multicast. This overhead depends on the number of egresses involved in the multicast distribution tree. For every egress involved that has group member associated with it, there will be an entry in the EEH by the ingress. The cost of EEH can be summarized as follows:

$$\text{Cost}_{\text{EEH}} = \textit{TotalEntries} + E*\textit{EntrySize}$$

where *TotalEntries* is the field giving the number of entries in the EEH, *E* is the number of egress entries, and *EntrySize* is the size of each entry. *TotalEntries* field takes 2 bytes and *EntrySize* field takes 2 bytes each. Hence

$$\begin{aligned}\text{Cost}_{\text{EEH}} &= 2 \text{ Bytes} + \text{E}*2 \text{ Bytes} \\ &= 2(\text{E}+1) \text{ Bytes}\end{aligned}$$

This is the cost considering the homogeneous QoS support where there is no need to put additional requested QoS information in the EEH. If heterogeneous QoS support is desired then there will be extra information to be added to the EEH – one byte succeeding each of the edge entries to hold the requested QoS information. Hence, in the case of heterogeneous QoS support,

$$\begin{aligned}\text{Cost}_{\text{EEH (Het)}} &= 2 \text{ Bytes} + \text{E}*3 \text{ Bytes} \\ &= (2+3*\text{E}) \text{ Bytes}\end{aligned}$$

On the other hand, in the encapsulation based technique, DSMCast, the header overhead of TEH is dependent upon the number of non-leaf routers in the domain that are on the multicast tree. For each involved core router an entry will be present in the TEH. The cost of the TEH can be summarized as follows:

$$\text{Cost} = \textit{NumEntries} + \textit{Options} + C*\textit{EntrySize}$$

where *NumEntries* is the field giving the number of entries in the TEH (1 byte), *Options* is the options field (1 byte), *C* is the number of entries of the cores, and *EntrySize* is the size of each entry. The cost of each entry is as follows:

$$\text{EntrySize} = \left\lceil \frac{\text{ID} + \text{Rep} + \text{Rep} * \text{SizeQos}}{8} \right\rceil$$

where ID is the size of the unique ID (8 or 16 bits), Rep is the maximum number of interfaces on any node in the domain (5 or 6 bits), and SizeQoS is the size of the QoS transformation field (2 or 3 bits). In addition, each entry is forced to byte boundaries to ensure ease of hardware support. The 5/2 (Rep = 5, SizeQoS = 2) and 6/3 settings deliver sizes of 23 bits (rounded up to 3 bytes) and 32 bits (4 bytes) respectively [27]. The general expression for the DSMCast header for a 4 byte *EntrySize* is as follows:

$$\begin{aligned}\text{Cost}_{\text{DSM}} &= 1 \text{ Byte} + 1 \text{ Byte} + C*4 \text{ Byte}\\ &= 2(2*C + 1) \text{ Byte}\end{aligned}$$

From the expressions for EdgeCast and DSMCast header overhead the necessary condition when EdgeCast overhead is less than that of DSMCast can be seen from the following relationship:

$$\begin{aligned}&2(E + 1) < 2(2*C + 1)\\ &\Rightarrow E < 2*C\end{aligned}$$

But when heterogeneous QoS support is considered for EdgeCast, then the condition is as follows:

$$\begin{aligned}&2 + 3*E < 2(2*C + 1)\\ &\Rightarrow E < 4/3*C\end{aligned}$$

4.2.2 Multicast Routing States

Multicast routing states in EdgeCast depends on the number of edge routers involved in the tree(s). Here, only the states in the core are considered since edge routers in both EdgeCast and other (source, group) based systems store the same (source, group) entries. Later on, in the simulation results also, the core states have been shown – no edge states have been considered in simulation.

If $Eg_{(i)}$ represents the number of edge routers at the end of the distribution tree(s) that passed through the core router *i* and there are total *C* core routers then the total number of states in the core routers is

$$EC_{states} = \sum_{i=1}^{C} Eg_{(i)}$$

In the worst case scenario for a core router when all the edge points are included in the distribution trees that pass through the core router the maximum entries will be equal to the total number of egress points. If maximum number of egress points is EgMAX then in a single core worst case states will be EgMAX.

For SSM if there are *G* individual groups having *S* individual sources the expression for total states is

$$SSM_{states} = \sum_{i=1}^{C} G_{(i)}$$

4.2.3 Message Complexity

From edge-to-edge for a new member to join a group, EdgeCast requires three messages: a QUERY_REQ message, a QUERY_ACK message and an EC_JOIN message towards the source. In case of QUERY_NACK there will not be any join. So, for a successful join there will be three messages and for an unsuccessful join there will be two messages involved. Hence, considering successful join there will be *3m* messages involved for establishing a group of size *m*. A join related message will be processed by all the intermediate hops along the way up to the other edge. If the average number of nodes from egress to ingress is H_{avg}, for a group of *m* the message processing cost is $3*m*H_{avg}$.

For DSMCast, a new member join requires sending DSMCast-Join-Request message to all eligible edge routers. There will be DSMCast-Bid messages sent back to the egress from each of the eligible edge routers. After selecting the proper edge for the join message to be sent, the egress sends a DSMCast-Join-Accept message to the appropriate edge. If the average number of eligible edge routers is *E,* then for a new member to join a group there will be $(2*E + 1)$ messages required. For establishing a group of m members there will be $(2*E + 1)*m$ messages involved. If h_{avg} is the average number of nodes to be traversed for a complete join cycle (by all the join related messages) then the processing cost for a group of m is $(2*E + 1)*m* h_{avg.}$

4.3 Simulation Studies

The performance of EdgeCast versus other solutions was analyzed through extensive simulation studies. The simulation studies were conducted using the ns-2 simulator [60] and its extension for differentiated services by Nortel Networks. The flowcharts of different simulation processes have been given in the appendix section. Since SSM is a state based multicast approach, the performance of EdgeCast (core routers' states) was compared to that of a generic version of PIM-SSM. Also the header overhead was compared with that of DSMCast because DSMCast is an encapsulation based multicast approach that works based on an additional header (TEH). The parameters for the simulations were as follows:

- The network topology consisted of 50 core nodes which were interconnected with degree of at least 3 and at most 6. There were 40 edge routers (ingresses and egresses) connected to the 20 core routers such a way that 2 edges were connected to one core. The edge routers were not directly connected to one another. Fig 4.1 shows the schematic of the network used for all the simulations in this thesis.

- There were 100 sources and 300 receivers available for simulation. It was assumed that there would be one source per group and so there would be maximum 100 groups available for simulation.
- Each multicast source produced Constant Bit Rate (CBR) traffic with constant packet size. Packets of varying rate were not considered.
- For different simulations the network bandwidth was changed to see the effect of resource scarcity on packet drop. Some other simulations were done keeping the network over-provisioned with BW of 10Mbps and link delay of 5ms per link.

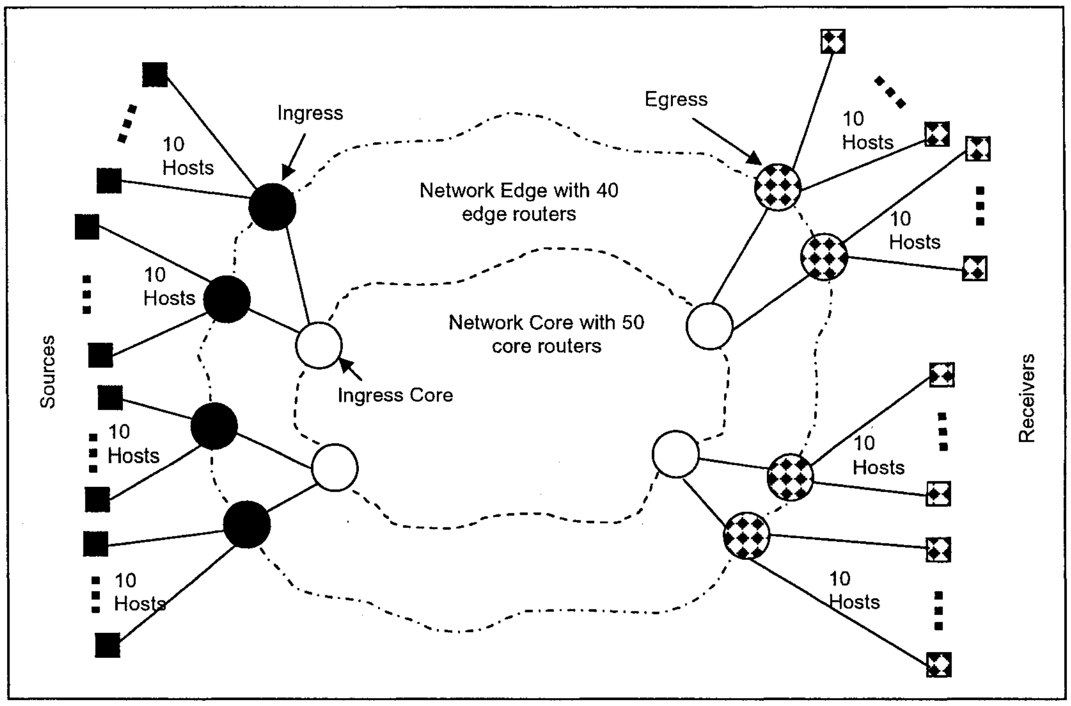


Fig 4.1 Schematic of the network used for simulation

It is to be noted that for DiffServ simulation the parameter of interest was packet loss in the network due to resource scarcity, no delay simulation was considered. The simulations were conducted for a single domain scenario of varying QoS distributions (homogeneous and heterogeneous). Another point of importance is that, for the packet statistics, the simulated data was considered only after the network reached a '*stable*' condition – means, when all the involved receivers joined the groups. The data of rather '*transient*' network conditions - when receivers were still joining the groups was not considered for consistency of the results.

4.3.1 Core Routers' States

EdgeCast core router states depend on the number of egresses involved in the distribution tree. Whereas, SSM core states depend on number of active (source, group). To have a comparative look at the core router states of EdgeCast and SSM, different situations have been simulated.

Case 1: *100 receivers joined different number of groups in different simulations. The number of groups was increased from 1 to 10 in successive simulations. But all the group sources were connected to a single ingress.*

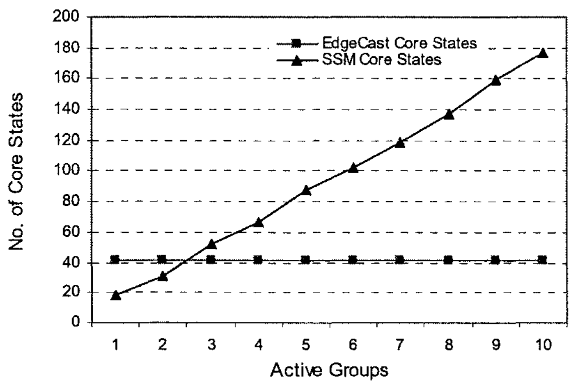


Fig 4.2 Total core states for EdgeCast vs. SSM for Case 1

Fig 4.2 depicts the scenario when EdgeCast shows its great advantage of resolving the multicast state scalability problem. Since EdgeCast stores the states of only the egress states of the domain, no matter how many active (source, group)s are involved in the multicast sessions in the domain, the states are constant as long as the same numbers of edges are involved in the distribution tree. On the other hand, SSM states are dependent on the number of active (source, group)s in the domain. The more groups are actively participating in the multicast communication, the more states are required to be stored in the on-tree routers. The figure shows the graph for SSM to be sharply increasing with the number of active groups, although all the packets for those different groups traversed the same distribution tree and hence the same on-tree routers. But graph for the EdgeCast is constant to a certain value, since the same edges had been traversed by the multicast packets.

***Case 2**: Number of groups remained constant (20 groups involving 2 ingresses). In each simulation the number of participating receivers was increased.*

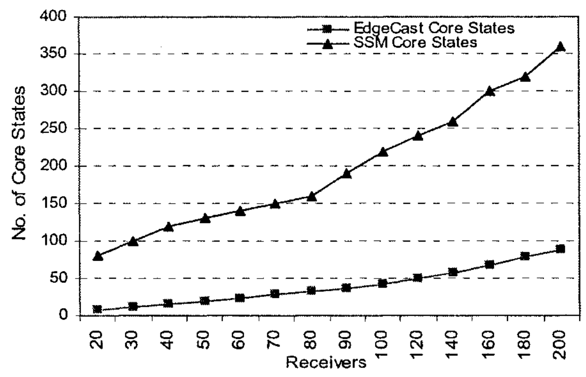


Fig 4.3 Total core states for EdgeCast vs. SSM for Case 2

Figure 4.3 is another situation where the difference between EdgeCast states and SSM states can be visualized clearly. As the number of receivers joining the multicast groups increases so does the involved egress points. Also the distribution tree becomes more and more sparse involving more and more core routers. But here the numbers of egresses being involved is much less than the number of new cores involved. And hence the SSM states increased much sharply than the EdgeCast states.

***Case 3**: Number of receivers (200) and number of active groups (20) were kept constant. But number of ingress involved with the group sources increased in each simulation.*

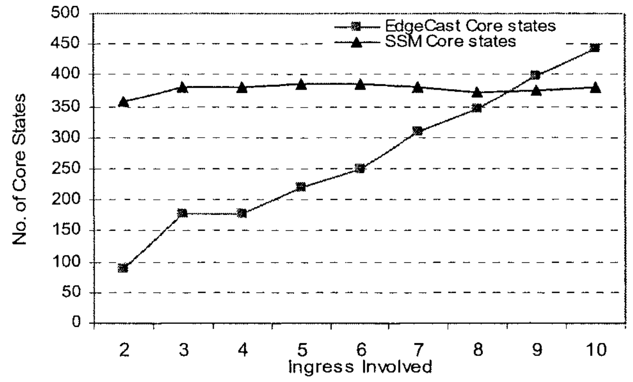


Fig 4.4 Total core states for EdgeCast vs. SSM for Case 3

Case 3 and its corresponding Fig 4.4 illustrate the situation of core states when the number of involved ingresses changes. For that situation, we see that the SSM states were almost the same for different numbers of ingresses involved. The core distribution changed the core states very little since the core states were dependent on the number of active groups – which was unchanged in that case. But for EdgeCast, because of the states' dependence on egresses, the graph rose rather sharply and at some point intersected the graph for SSM states. This was because, as more ingress got involved, more distinct trees got involved. So, new edge entries were required in the 'new' involved cores, resulting in an increase in the number of core entries.

4.3.2 Encapsulated Header Overhead

Here the simulations were done to have a comparative picture of EdgeCast and DSMCast header overhead – EEH of EdgeCast and TEH of DSMCast. A couple of cases were simulated to see the effect of egress involvement and ingress-core involvement. Here '*ingress-core*' means the core routers which are directly connected to the ingresses. Another simulation was performed to quantify the number of core routers involved for

DSMCast vs. number of edge routers involved for EdgeCast, since the header overhead is directly dependent on the number of cores for DSMCast and on the number of edges for EdgeCast.

Case 4*: A number of receivers joined a group. In each simulation the number of egresses involved was increased and hence number of receivers joining the group also increased. But the ingress involved was kept constant to one.*

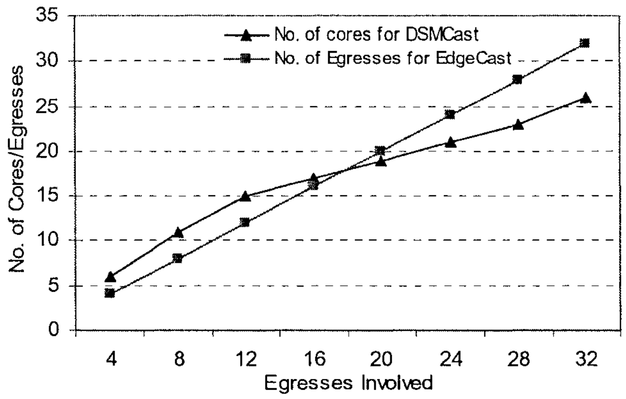


Fig 4.5 Total number of cores/egresses for DSMCast/EdgeCast for Case 4

Since DSMCast performs based on the TEH created depending on the involved cores and EdgeCast performs based on the EEH created depending on the involved egresses, it is an interesting thing to see, for certain cases, how the involved cores and edges varies for DSMCast and EdgeCast respectively. Case 4 and Fig 4.5 illustrate such a situation. As more receivers involving more egress points join the multicast groups, both the involved cores and edges increase up to a certain point and EdgeCast shows better performance than DSMCast. But after a certain threshold point, the core routers are not increased as the same rate of egresses. This is because; at that time more and more cores start being shared by the groups. So after that special situation, DSMCast core numbers start to fall below the EdgeCast edge numbers.

***Case 5**: The header overhead related to the transport of a single packet generated from a source - for DSMCast and for EdgeCast. The number of cores and egresses involved were taken from the previous Fig 4.5 of Case 4. This is for homogeneous QoS situation.*

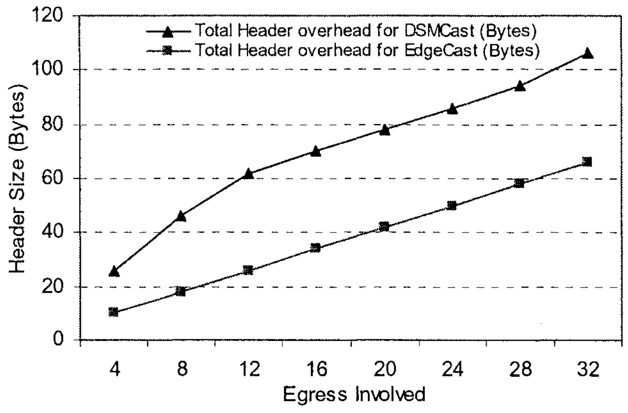


Fig 4.6 Header Overhead for EdgeCast vs. DSMCast for Case 5

***Case 6:** The effect of number of ingress-cores (cores directly connected to the ingresses) involved in multicast to 100 receivers which are connected to 10 egresses. One sender per ingress is considered. Since two ingresses are connected to a single ingress-core, each ingress-core, in this case, will handle two packets (one packet from each sender).*

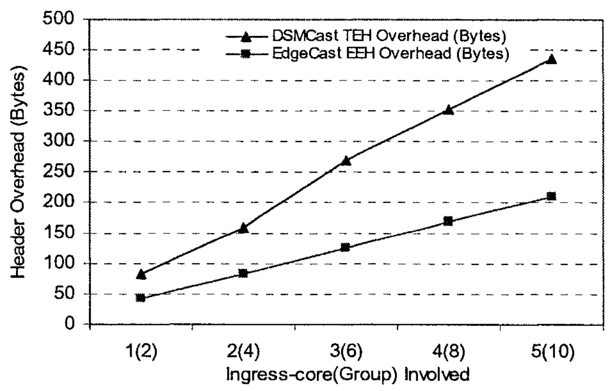


Fig 4.7 Header overhead for EdgeCast vs. DSMCast for Case 6

Fig 4.6 and Fig 4.7 illustrate the header overhead of DSMCast TEH vs. EdgeCast EEH for situations described in Case 5 and Case 6. DSMCast TEH uses 4-Bytes/entry and 2-Bytes for *NumEntries* and Options fields and EdgeCast EEH uses 2-Bytes/entry and 2 Bytes for *TotalEntries* field. For Fig 4.6, as the number of egresses increases in the distribution tree, the tree becomes more sparse. This also increases the number of cores involved in the tree. From the figure it is evident that the graph for EdgeCast is linearly increasing, because of the linearity in the number of joining egresses. But the graph for TEH is nonlinear since the number of cores does not increase linearly as the group members become more sparse.

For Fig 4.7, the graphs are a bit tricky. Although the number of joining egresses was the same, the EdgeCast EEH graph is linear – how is that? It is to be noted that for two different packets from two different sources the egress number will be cumulative. So, when two groups are active, cumulatively the egresses will be 20 (10 for each group's packet). That is why the graphs are increasing – linearly for EdgeCast and nonlinearly for DSMCast. For both the figures, till the simulation limit the EEH overhead was always less than the TEH overhead, since the network could not satisfy the necessary condition for overhead equality. But from the previous discussion, it is clear that if the network satisfied the necessary condition, the graphs would intersect at some point.

4.3.3 DiffServ by EdgeCast

This section provides simulation results that prove that EdgeCast provides differentiated services in the domain. There are different parameters of QoS that can be measured as to verify the differentiated services treatment by the DiffServ network. Here only packet loss has been considered as the DiffServ parameter. A couple of situations have been simulated to provide a clear picture of how the packet loss percentage varies depending on the level of network resource scarcity. Here the considered network resource was bandwidth. A number of senders were sending data to a number of receivers who joined different multicast groups. The senders were classified into two major categories based on their priorities. Packets from the first class of traffic source were given QoS level 10 as default and any traffic exceeding the desired rate were marked with QoS level 11. On the other hand, packets from the second class of sources were marked 20 by default and the desired rate exceeding packets were marked as 21. So, in the domain, there was traffic with four different levels of QoS priorities. The priorities were set as follows:

$$10 > 20 > 11 > 21$$

In the case of resource scarcity (lack of necessary bandwidth) packets with QoS level 10 should be dropped the least and packets with QoS level 21 should be dropped the most. The following simulation results prove this fact by showing different drop percentage for packets with different QoS levels.

***Case** 7: Bandwidth of the entire network was increased gradually in different simulations. In each simulation, all the links were given the same BW and delay.*

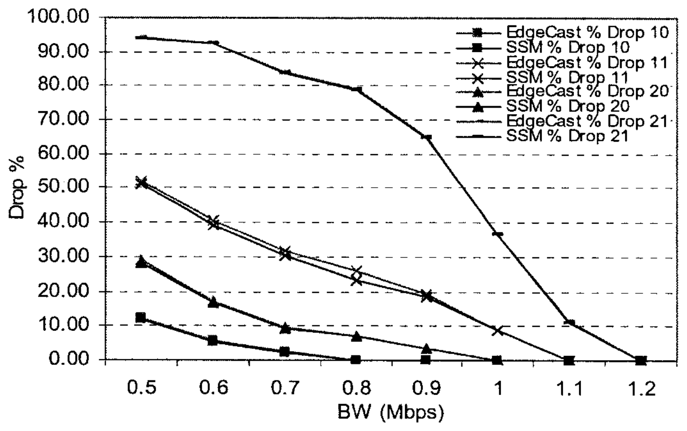


Fig 4.8 Drop percentage of packets with different DSCPs for varying network BW for EdgeCast vs. SSM

Case 8: *Entire network BW was kept constant, but the number of active sources was increased gradually in different simulations.*

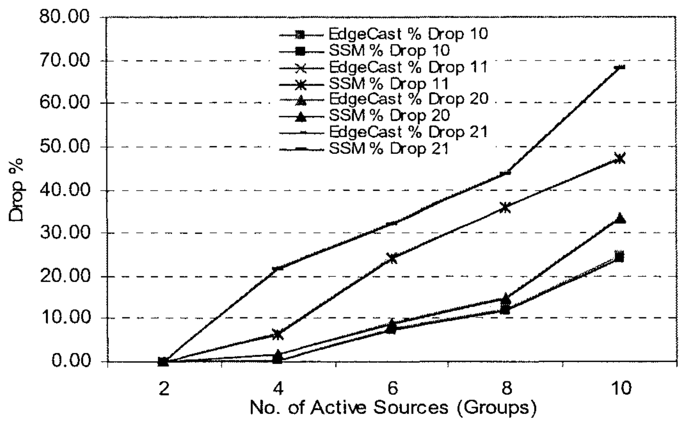


Fig 4.9 Drop percentage of packets with different DSCPs for varying number of active sources for EdgeCast vs. SSM

Case 7 and Case 8 and their corresponding Fig 4.8 and Fig 4.9 illustrate the situations where network traffic was getting differentiated services. Both of the figures plot comparative data of EdgeCast and SSM providing DiffServ multicast. Fig 4.8 shows that at the beginning, when the BW was scarce, there was more packet drop. But the percentage drop of higher priority packets was always less than that of the lower priority packets. As the network BW increased, the drop percentage started to decrease, since there were less congestions being experienced by the network. Moreover, higher priority packets' drop percentage reached zero level much earlier than the lower priority packets. As per the DiffServ theory this is exactly how the network was supposed to behave.

Fig 4.9 depicts another DiffServ situation where the network BW was kept constant, but the number of active groups varied. At first there was zero drop percentage, since only two groups were active. As the active groups started to increase there was more traffic in the network. Some bottleneck points started to experience congestion and the packets were being dropped. For more active groups, more packets were being dropped. But, the higher priority packets' drop percentage was always less than that of lower priority packets.

It is evident from both scenarios and the figures that EdgeCast is very much capable of providing DiffServ multicast. Simulations show it performs almost the same as SSM. In a few graphs EdgeCast shows marginally degraded performance than SSM, but in most of the cases the graphs are of same shape. The marginal degraded performance is a direct effect of the little bit complex lookup and forward technique involved in EdgeCast. With the aid of better lookup algorithm and high end hardware, this marginal difference could be eliminated.

4.3.4 Heterogeneous QoS Support in EdgeCast

While the previous sections presented the results for header overhead, core multicast states and differentiated services treatment of the packets, this section presents simulation results that prove the heterogeneous QoS supporting capability of EdgeCast. Specifically, the issue of QoS is complicated by the need for support of heterogeneous QoS in the multicast tree. Another aspect of consideration is the '*good neighbor effect*' [27]. Although two different receivers who have joined the same (source, group) can ask for two different levels of QoS, till the branching point, the receiver requesting lower QoS will receive the higher QoS. This is because, the other receiver, who is paying for the higher QoS must receive treatment accordingly and being the same flow till the branching point, the second receiver's packets will receive better treatment because of the first receiver – the *good neighbor*.

Let us consider a section of network as the following Fig 4.10, where two receivers R1 and R3 have joined the group G1. The different receivers requesting QoS are: R1 asking QoS 10 (G1) and R3 asking QoS 11 (G1). The source S1 is the source for groups G1.

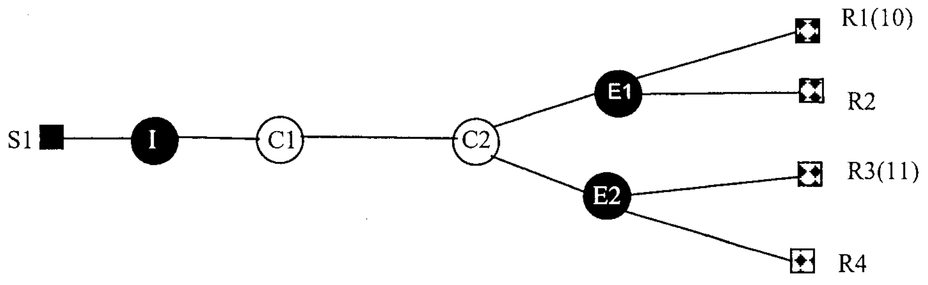


Fig 4.10 A Network Section Requesting Heterogeneous QoS

Two different situations have been simulated to show the effect of heterogeneous QoS support. In the first case, the simulation was done in an over-provisioned network with enough bandwidth and no congestion. The second case simulated the situation when bandwidth was scarce and congestion occurred in the network.

Case 9: *A number of receivers joined different groups. The bandwidth was enough so that no congestion occurred.*

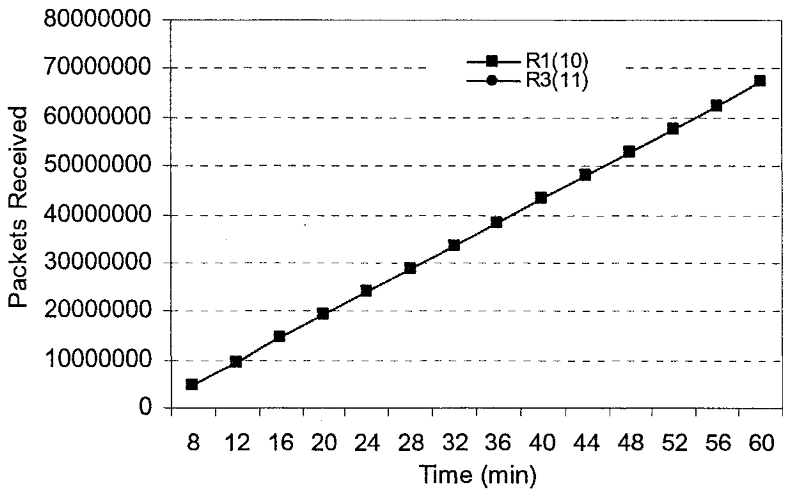


Fig 4.11 Packets Received by the Same Group Receivers (over-provisioned network)

***Case 10:** A number of receivers joined different groups. The bandwidth was reduced to a certain level so that there was bandwidth scarcity and congestion occurred in the network.*

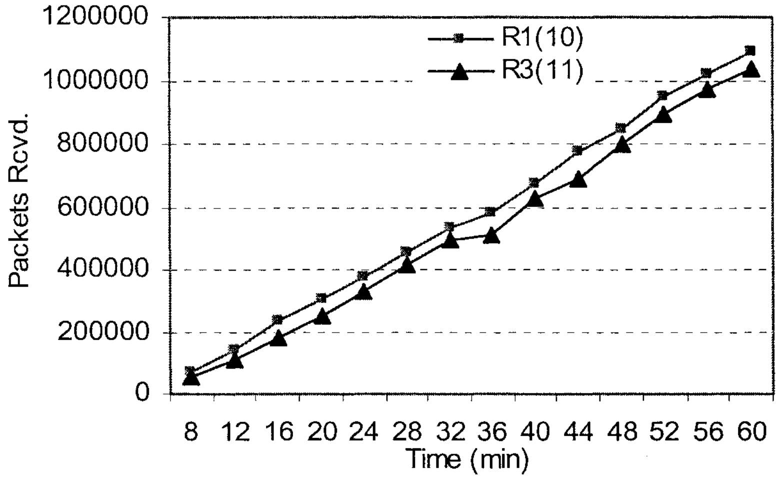


Fig 4.12 Packets Received by the Same Group Receivers (congested network)

Fig 4.11 and Fig 4.12 show the packet reception by the two receivers of the same group. A number of groups were actively sending data to a number of receivers in the network. But, here only two receivers and a single source have been chosen as a network section to show the effect of heterogeneous QoS requirements by different receivers. Fig 4.11 shows that both the receivers received the same number of packets regardless of their QoS requirements. That is why two different graphs have been superimposed on each other. Here, the network had ample bandwidth and no congestion occurred. There was no packet drop and the receivers received data properly. But the Fig 4.12 shows a different scenario when there was congestion in the network because of bandwidth scarcity. Packets with different DSCPs received different treatment in the time of congestion resulting variation in packet reception by the receivers.

Another point to note is that, the graphs are not smooth in their shapes. This is because, the results considered a section of the network. There were other sources sending data to some other receivers. And the congestion was the result of all data from a number of sources than simply the two sources. Moreover the simulated network was not '*perfectly*' tuned due to limitations. But the overall picture from the graph show that the network was providing DiffServ multicast services.

4.4 Summary

This chapter has presented the theoretical and simulation studies of EdgeCast versus other encapsulation-based approach DSMCast and state-based approach SSM. The header overhead, multicast states' cost and message complexity have been analyzed from theoretical point. Later on, some simulation results have been presented to quantify different parameters of EdgeCast versus other approaches like DSMCast and SSM. EdgeCast showed better performance in state cost when a large number of source and groups were active involving a rather small number of edge points in the end of distribution tree(s). Also, the encapsulated header overhead showed better performance than that of DSMCast up to a certain number of edges with respect to cores. The results also confirmed that EdgeCast is capable of providing differentiated multicast services. Intra-group heterogeneous QoS can also be provided by EdgeCast, although by sacrificing the header overhead a bit with that of DSMCast.

Chapter 5

CONCLUSIONS AND FUTURE WORKS

In this chapter, a summary of the contributions to the thesis is presented and additional considerations of the work are discussed. In addition, the chapter offers several concluding remarks regarding the use of the contributions and concludes with several future research directions.

5.1 Contributions

The work in this thesis contributed to a unique idea of DiffServ multicast. The notable contributions are as follows:

- Introduction to a hybrid approach – EdgeCast, of state-based and encapsulation-based approaches for DiffServ multicast.
- Resolving multicast state scalability problem by using edge states in the routers instead of (source, group) states.
- Introduction to the architectural and functional extensions to the Bandwidth Broker and ingress/egress routers of a DiffServ domain.
- Developments of member join and leave protocols for supporting multicast member dynamics that are different in operation than traditional IP multicast member dynamics protocols.
- The support of QoS heterogeneity - different receivers joining the same multicast group with different levels of QoS requirements.

5.2 Conclusions

So far, all the multicast techniques were based on either the (source, group) or encapsulation based approaches. This thesis proposes EdgeCast that provides multicasting based on a hybrid approach of both the states and header encapsulation. And moreover, the states are not of (source, group)s, rather of the egresses that are being used in the multicast distribution tree. The most notable contribution is the introduction of a method that made the multicast states independent of the number of active (source, group)s, which tends to explode for a large number active (source, group)s in a domain. The multicast states of EdgeCast are much more stable than the number of (source, group) states. The network administrator can assume the worst case scenario - from the scalability perspective, with much ease because of the knowledge of domain edge.

Unlike DSMCast, EdgeCast does not need the ingress routers to have complete knowledge about the domain topology for operation. The XBB performs the source registration, authentication and QoS verification. This solves the problems of malicious source activities, source status confirmation and resource management. EdgeCast operates by comparing the EEH entries with that of MFC entries in the routers - replacing the procedure of how the multicast packets themselves are transported. EdgeCast shows better performance from scalability and header overhead perspectives for certain network situations, but these come at the cost of requiring hardware assistance at each of the core routers. As an intermediate step or on networks with low levels of multicast traffic, the processing of the EdgeCast EEH could be done via software although such an approach is merely an interim step.

The EdgeCast architecture is also capable of supporting heterogeneous QoS that is a growing need for next generation QoS multicast services. The introduction to heterogeneous QoS support in EdgeCast reduces a bit, the benefits from header overhead with respect to other encapsulation-based approach like DSMCast. Even then, in general, EdgeCast incurs less header overhead than DSMCast. The work on heterogeneous QoS offered interesting insights into how heterogeneous QoS performs in a single multicast tree. Most notably, the tradeoff of network efficiency versus service differentiation is a difficult problem that is an open topic for research.

The work in this thesis did not consider the fault detection and tolerance situations. It was assumed that there will be an underlying scheme to take care of the fault situations. Also it was assumed that the Bandwidth Broker is capable of managing the QoS matters. Based on the query by the egress the BB (or XBB, the Extended BB) performs the necessary check to evaluate whether the requested QoS can be guaranteed.

5.3 Future Works

Although EdgeCast provides a solution to DiffServ multicast problems, the work has also opened up several other interesting areas of future research. The areas of potential future research are as follows:

- *'Load Shedding' off the EEH:* The encapsulated packet header EEH of EdgeCast is the necessary part of the system. This EEH has to be carried by each and every packet through the ingress towards the multicast members. When the distribution tree is very sparse and lot many egresses are included in every packet, a lot of bandwidth is wasted. Moreover, after the branching nodes a good number of egress entries become totally unnecessary, since one branch of the tree does not require the egresses of other branches for replication and forwarding decisions. An efficient procedure that can '*shed off*' those unnecessary entries can save a lot of network bandwidth. But development of such a protocol would be very challenging, since DiffServ core needs to be kept as simple as possible.

- *Efficient EEH lookup:* The EEH lookup and matching with the MFC entries is an area of performance concern for EdgeCast. A better and smarter algorithm can be used that can perform the EEH lookup in the least number of iterations and avoid any kind of redundant packet replication. This should be done in such a way that the process complexity does not overwhelm the core functionality.
- *Delay Analysis:* Delay is another very important parameter of QoS sensitive network. The work in the thesis only considered 'packet loss' as the QoS parameter, not delay. The future work would be also to simulate the delay (both link delay and processing delay) to analyze the performance of EdgeCast from the delay perspective. In that case, both the 'packet loss' and 'delay' analysis would be able to provide a complete picture of the EdgeCast supported network.
- *Fault Management:* Although the proper provisioning and monitoring of QoS is critical for next generation applications, an equally important factor for QoS is the underlying fault-tolerance of the network. In order to provide the necessary QoS levels to QoS-based applications, the network must offer satisfactory connectivity across the network at all times. A protocol like EI-HELLO (Edge-based Intelligent HELLO) [61] can be used to augment the existing link state protocols through edge-based monitoring of the internal core routers. The 'heartbeat' data packets can be used to detect the network connectivity situation and thus ensuring that the network is

healthy. If a fault is detected then necessary measures can be taken to restore the desired operation with the least amount of performance penalty. This can be done by rerouting the multicast traffic as per the new join establishment. This is important, since EdgeCast is a receiver-initiated DiffServ multicast provisioning technique.

References

1. S. Deering, “Host Extension for IP Multicast”, IETF RFC 1112, August 1989
2. R. Braden et al., “Integrated Services in the Internet Architecture: An Overview”, IETF RFC 1633, June 1994
3. R. Braden et al., "Resource ReSerVation Protocol (RSVP) --Version 1 Functional Specification”, IETF RFC 2205, September 1997
4. S. Blake et al., “An Architecture for Differentiated Services” IETF RFC 2475, December 1998
5. Y. Bernet et al., "A Framework for Differentiated Services," Internet Draft, draft-ietf-diffserv-framework-02.txt, Feb. 1999
6. E. Rosen et al., "Multiprotocol Label Switching Architecture”, IETF RFC 3031, January 2001
7. S. Deering et al., “Distance Vector Multicast Routing Protocol”, IETF RFC 1075, November 1988
8. A. Adams et al., “Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)”, IETF Internet Draft, draft-ietf-pim-dm-new-v2-05.txt, June 2004, Work in Progress
9. S. Deering et al., “Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification”, IETF RFC 2362, June 1998
10. J. Postel, J. Reynolds, “File Transfer Protocol (FTP)”, IETF RFC 959, October 1985
11. W. Fenner, “Internet Group Management Protocol, Version 2”, IETF RFC 2236, November 1997

12. David J. Wright, “Voice over Packet Networks,” John Wiley and Sons, March 2001

13. Sandeep Sharma, “Achieving voice quality in packet networks”, Express Computers, http://www.expresscomputeronline.com/20021111/tech1.shtml, June 2004

14. D. Awduche et al., "Requirements for Traffic Engineering over MPLS", IETF RFC 2702, September 1999

15. E. Crawley et al., "A Framework for QoS-based Routing in the Internet", IETF RFC 2386, Aug. 1998

16. S. Shenker et al., "Specification of Guaranteed Quality of Service", IETF RFC 2212, September 1997

17. J. Wroclawski, "Specification of the Controlled-Load Network Element Service", IETF RFC 2211, September 1997

18. K. Ramakrishnan, S. Floyd, “A proposal to Add Explicit Congestion Notification (ECN) to IP," IETF RFC 2481, January 1999

19. B. Davie et al., “An Expedited Forwarding PHB (Per-Hop Behavior)," IETF RFC 3246, Mar. 2002

20. J. Heinanen et al., “Assured Forwarding PHB Group," IETF RFC 2597, June 1999

21. R. Bless, K. Wehrle, “A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services " IETF RFC 3662, December 2003

22. R. Bless, K. Wehrle, “IP Multicast In Differentiated Services Networks”, IETF RFC 3754, April 2004

23. R. Bless, K. Wehrle, “Group Communication In Differentiated Services Networks”, Internet QoS For Global Computing (IQ 2001), May 2001, pp. 618-625

24. Dave Kosiur, "IP Multicasting: The Complete Guide to Interactive Corporate Networks", John Wiley & Sons Inc., 1998

25. Mike Macedonia, Don Brutzman, "MBONE, the Multicast BackbONE", http://www-mice.cs.ucl.ac.uk/multimedia/projects/mice/mbone_review.html, June 2004.

26. White Paper "Building Next Generation Network Processors," Agere Systems, April 2001

27. Aaron Striegel, "Scalable Approaches for DiffServ Multicasting", PhD Thesis, Iowa State University, 2002

28. S. Bhattacharya, "An Overview of Source Specific Multicast", IETF RFC 3569, July 2003

29. C-Y. Lee, N. Seddigh, "Controlling the Number of Egress Points in Dynamic Multicast Groups," IETF Internet Draft, draft-leecy-multicast-egress-limit-00.txt, January 2000

30. P. Van Mieghem et al., "On the Efficiency of Multicast," IEEE/ACM Transactions on Networking, vol. 9, no. 6, Dec. 2001, pp. 719-132

31. R.C. Chalmers, K. Almeroth, "Modeling the Branching Characteristics and Efficiency Gains in Global Multicast Trees," IEEE INFOCOM, April 2001, pp. 449-458.

32. M. Ammar et al., "QoS Routing for Anycast Communications: Motivation and An Architecture for DiffServ Networks," IEEE Communications, June 2002, pp. 48-56

33. W. Moh et al., "Differentiated Service-Based Inter-Domain Multicasting Over the Internet," IEEE Optical Networks, March/April 2001, vol. 2, no. 2, pp. 41-56

34. K. Ravindran, Ting-Jian Gong, “Cost Analysis of Multicast Transport Architectures in Multiservice Networks," IEEE/ACM Trans. on Networking, Feb. 1998, vol. 6, no.1, pp. 94-109

35. B. Yang, P. Mohapatra, “Multicasting In Differentiated Services Domains”, IEEE GLOBECOM , Nov 2002, vol.3, pp. 2074-2078

36. M. Gupta, M. Ammar, “Providing Multicast Communication In Differentiated Services Network Using Limited Branching Techniques”, 3rd International Conference on Internet Computing (IC 2002), June 2002, pp. 156-163

37. G. Bianchi et al., “QUASIMODO: Quality of Service-Aware Multicasting over DiffServ and Overlay Networks”, IEEE Network, Jan./Feb. 2003, vol. 17, pp. 38-45

38. G. Bianchi et al., “PerFlow QoS Support over a Stateless DiffServ Domain”, Computer Networks, Special Issue, Towards A New Internet Architecture, September 2002, vol. 40, pp. 73-87

39. A. Fei, M. Gerla, "Receiver-Initiated Multicasting with Multiple QoS Constraints,", IEEE INFOCOM, May 2000, vol. 1, pp. 62-70

40. Z. Li, P. Mohapatra, "QoS-Aware Multicasting in DiffServ Domains," IEEE GLOBECOM , Nov 2002, vol.3, pp. 2118-2122

41. M. Faloutsos et al., “Qos-Aware Multicast Routing for The Internet: The Design and Evaluation of Qosmic”, IEEE/ACM Transactions on Networking, February 2002, pp. 54-66

42. Z. Li, P. Mohapatra. “QoS-aware Multicast Protocol Using Bounded Flooding (QMBF) Technique”, ICC, May 2002, vol. 2, pp. 1259 - 1263

43. G. Bianchi et al., "Admission Control Over Assured Forwarding PHBs : A Way to Provide Service Accuracy Over DiffServ Framework" , IEEE GLOBECOM, Nov 2001, pp. 25-29

44. J. Tian, G. Neufeld, "Forwarding State Reduction for Sparse Mode Multicast Communication", IEEE INFOCOM, Mar./Apr. 1998, vol. 2, pp. 711-719

45. D. Thaler, M. Handley, "On the Aggregatability of Multicast Forwarding State", IEEE INFOCOM, March 2000, vol. 3, pp. 1654-1663

46. M. Gerla et al., "A Protocol to Improve the State Scalability of Source Specific Multicast", IEEE GLOBECOM, November 2002, vol. 2, pp. 1899-1904

47. M. Faloutsos et al., "Scalable QoS Multicast Provisioning in DiffServ-supported MPLS Networks", IEEE GLOBECOM, November 2002, vol. 2, pp. 1450-1454

48. S.G. Rao et al., "A Case for End System Multicast", IEEE Journal on Selected Areas in Communications, vol. 20, October 2002, pp. 1456-1471

49. J. Brooks, M. Jurczyk, "Creating Edge-to-Edge Multicast Overlay Trees for Real Time Video Distribution", Internet Computing 2003 / International MultiConference in Computer Science , June 2003, Vol.1, pp. 371-377

50. K.Nichols et al., "A Two-bit Differentiated Services Architecture for the Internet", IETF RFC 2638, July 1999

51. R. Neilson et al., "A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment", Internet2 Qbone BB Advisory Council (work in progress), http://qos.internet2.edu/qbone/QBBAC.shtml, July 2004

52. Internet Assigned Numbers Authority, http://www.iana.org, July 2004

53. W. Fenner et al., “Internet Group Management Protocol, Version 3”, IETF RFC 3376, October 2002

54. H. Holbrook, D. Cheriton, “IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications”, ACM SIGCOMM, September 1999, pp. 65-78

55. Marcel Waldvogel et al., “Scalable High Speed IP Routing Lookups," ACM SIGCOMM, September 1997, pp. 25-36

56. IP Multicast Training Materials, “Multicast Source Discovery Protocol," Cisco Systems, July 2004.

57. IP Multicast Training Materials, “Multi-protocol BGP," Cisco Systems, July 2004.

58. Y. Rekhter et al., “Multiprotocol Extensions for BGP-4," IETF RFC 2283, February 1998

59. A. Ballardie, “Core Based Tree Multicast Routing Architecture”, IETF RFC 2201, September 1997

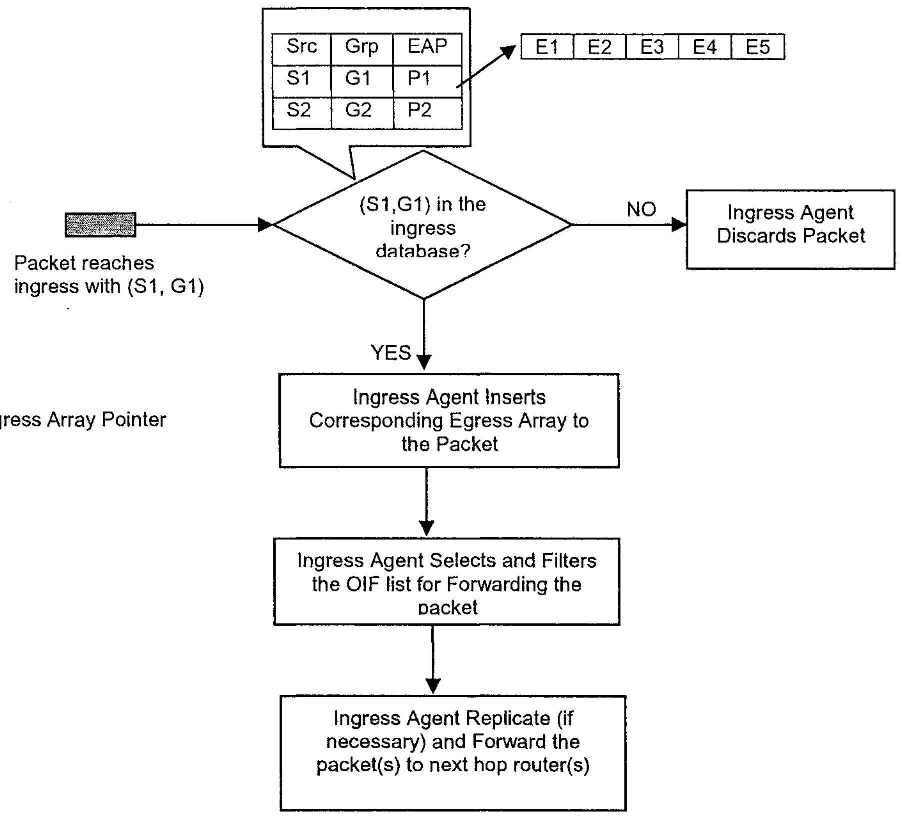
60. “The Network Simulator - ns-2”, http://www.isi.edu/nsnam/ns/, July 2004

61. A. Striegel, G. Manimaran, "Edge-based Fault Detection in a DiffServ Network," International Conference on Dependable Systems and Networks (DSN), June 2002, pp. 79-88

Appendix

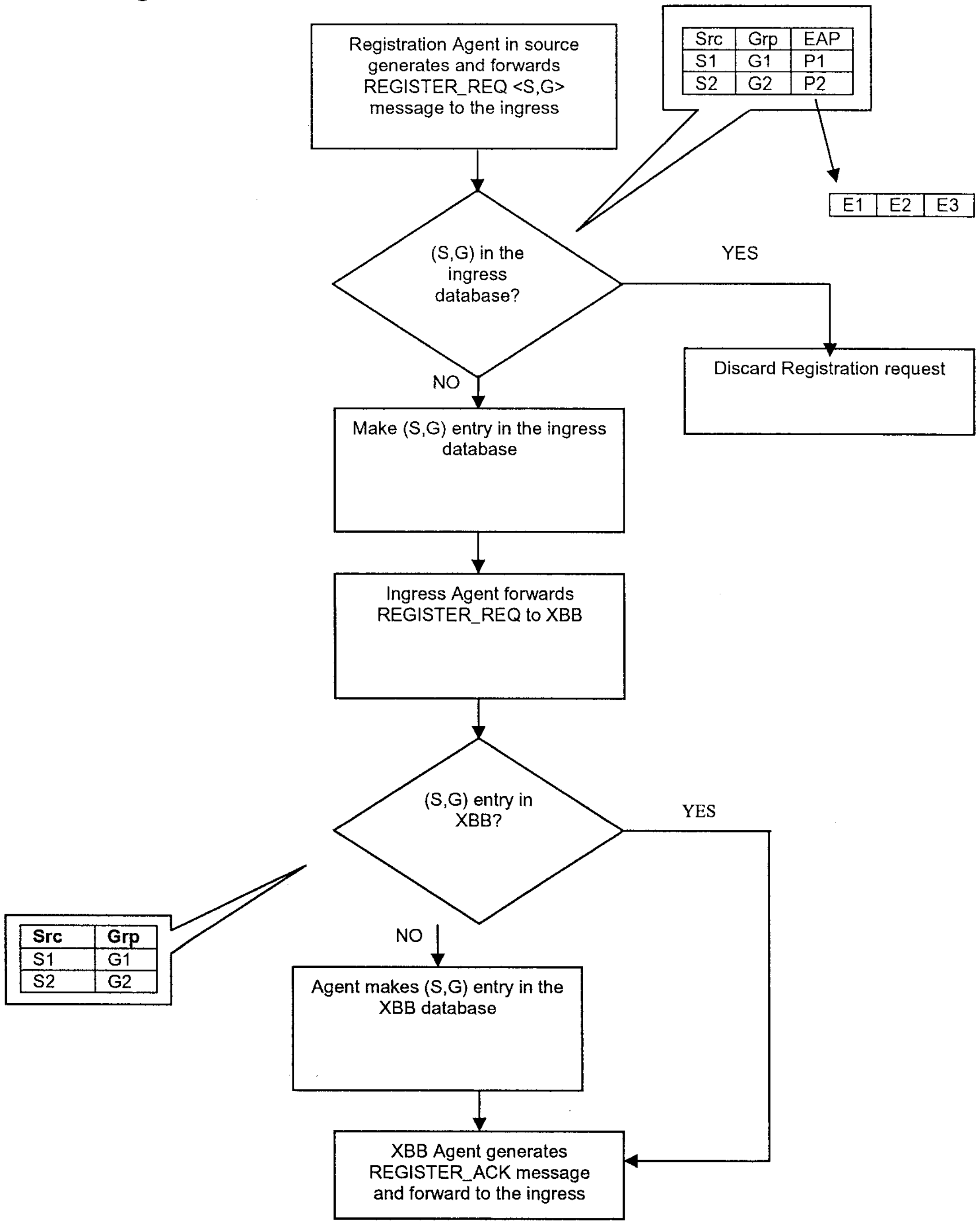
A.1 Simulation Flowcharts

A.1.1 Generation and Encapsulation of EEH



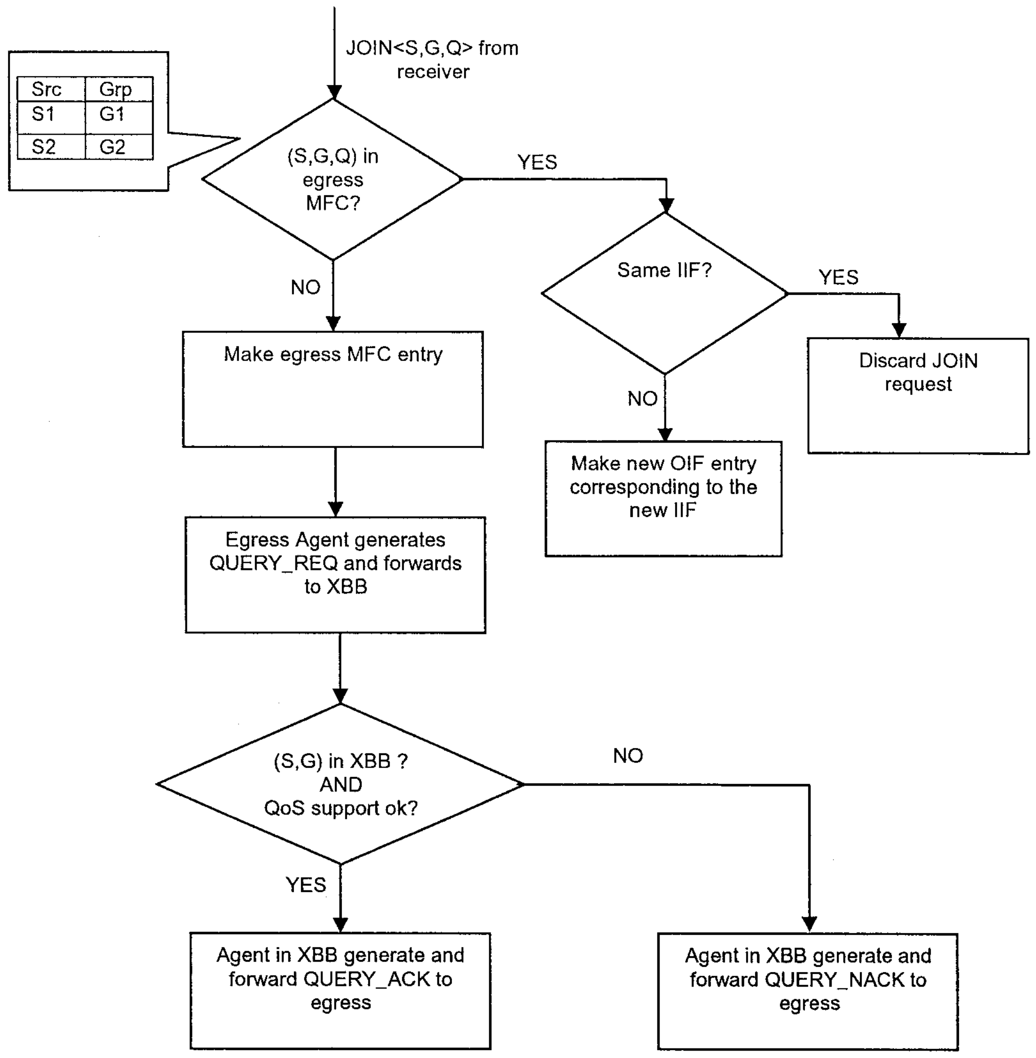
Src
Grp
EAP
S1
G1
P1
S2
G2
P2
E1
E2
E3
E4
E5
(S1,G1) in the ingress database?
NO
Ingress Agent Discards Packet
Packet reaches ingress with (S1, G1)
YES
EAP: Egress Array Pointer
Ingress Agent Inserts Corresponding Egress Array to the Packet
Ingress Agent Selects and Filters the OIF list for Forwarding the packet
Ingress Agent Replicate (if necessary) and Forward the packet(s) to next hop router(s)

A.1.2 Registration

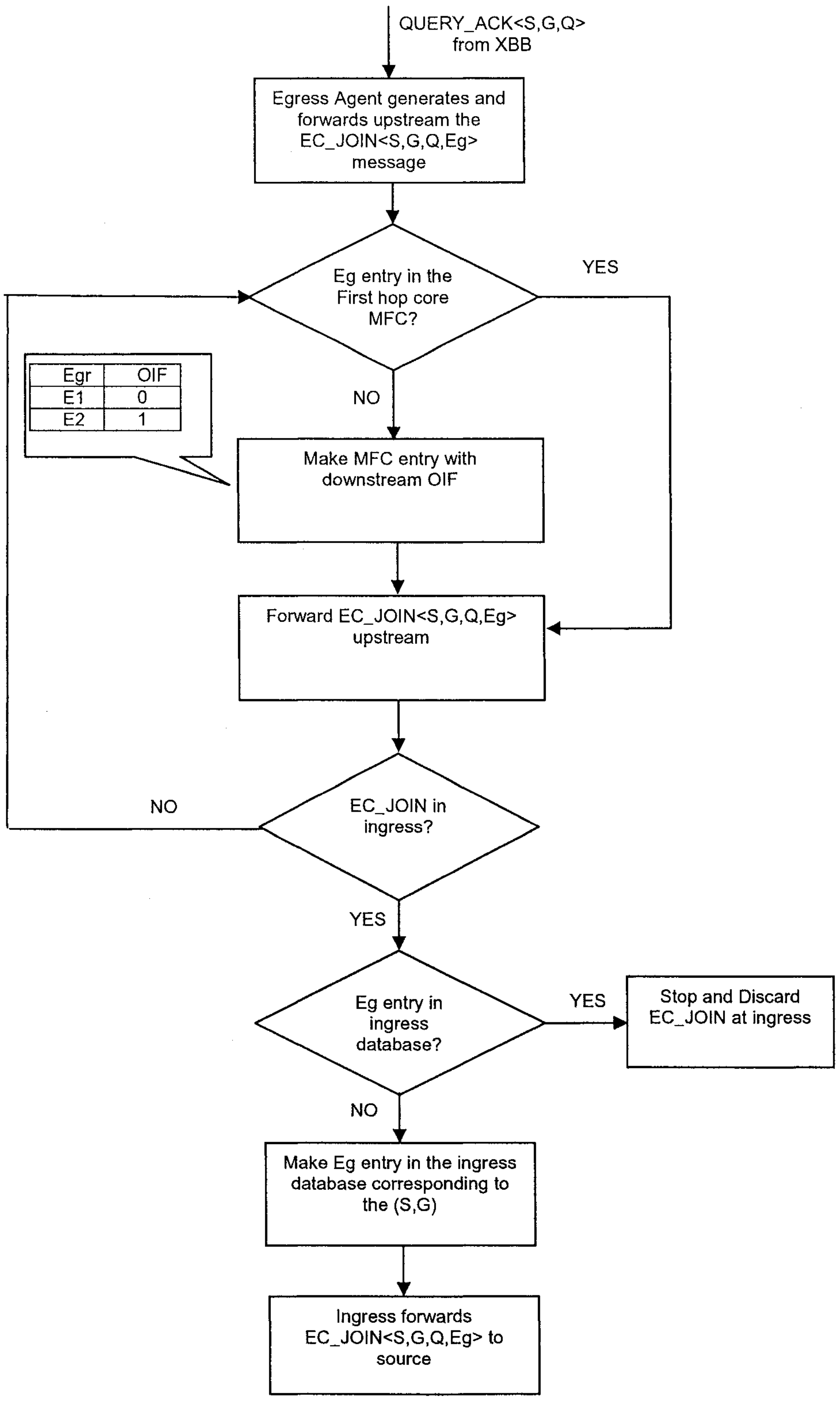


Registration Agent in source generates and forwards REGISTER_REQ <S,G> message to the ingress
Src Grp EAP
S1 G1 P1
S2 G2 P2
E1 E2 E3
(S,G) in the ingress database?
YES
Discard Registration request
NO
Make (S,G) entry in the ingress database
Ingress Agent forwards REGISTER_REQ to XBB
(S,G) entry in XBB?
YES
Src Grp
S1 G1
S2 G2
NO
Agent makes (S,G) entry in the XBB database
XBB Agent generates REGISTER_ACK message and forward to the ingress

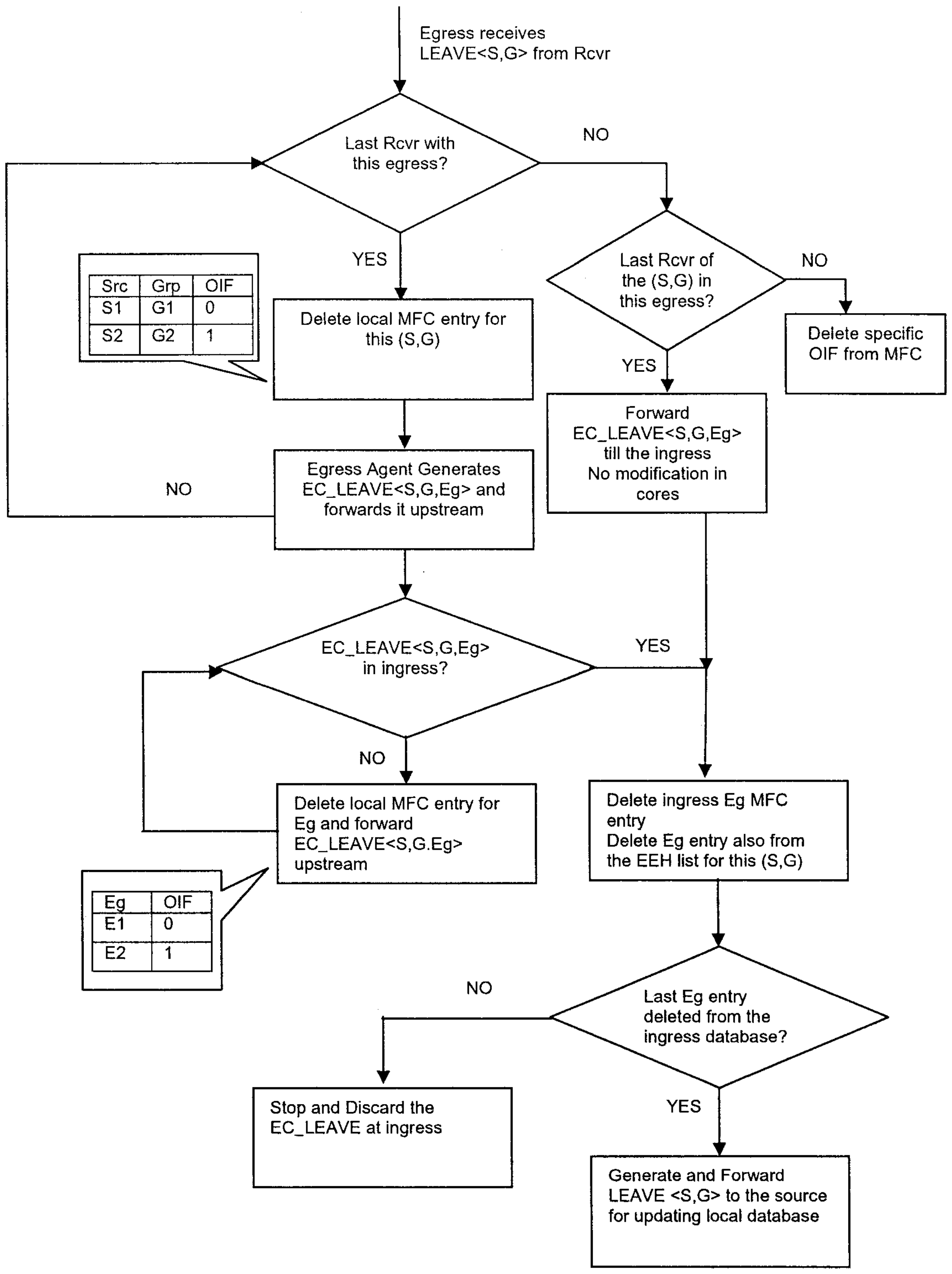
A.1.3 Query


JOIN<S,G,Q> from receiver
Src
Grp
S1
G1
S2
G2
(S,G,Q) in egress MFC?
YES
NO
Same IIF?
YES
NO
Make egress MFC entry
Discard JOIN request
Make new OIF entry corresponding to the new IIF
Egress Agent generates QUERY_REQ and forwards to XBB
(S,G) in XBB ? AND QoS support ok?
NO
YES
Agent in XBB generate and forward QUERY_ACK to egress
Agent in XBB generate and forward QUERY_NACK to egress

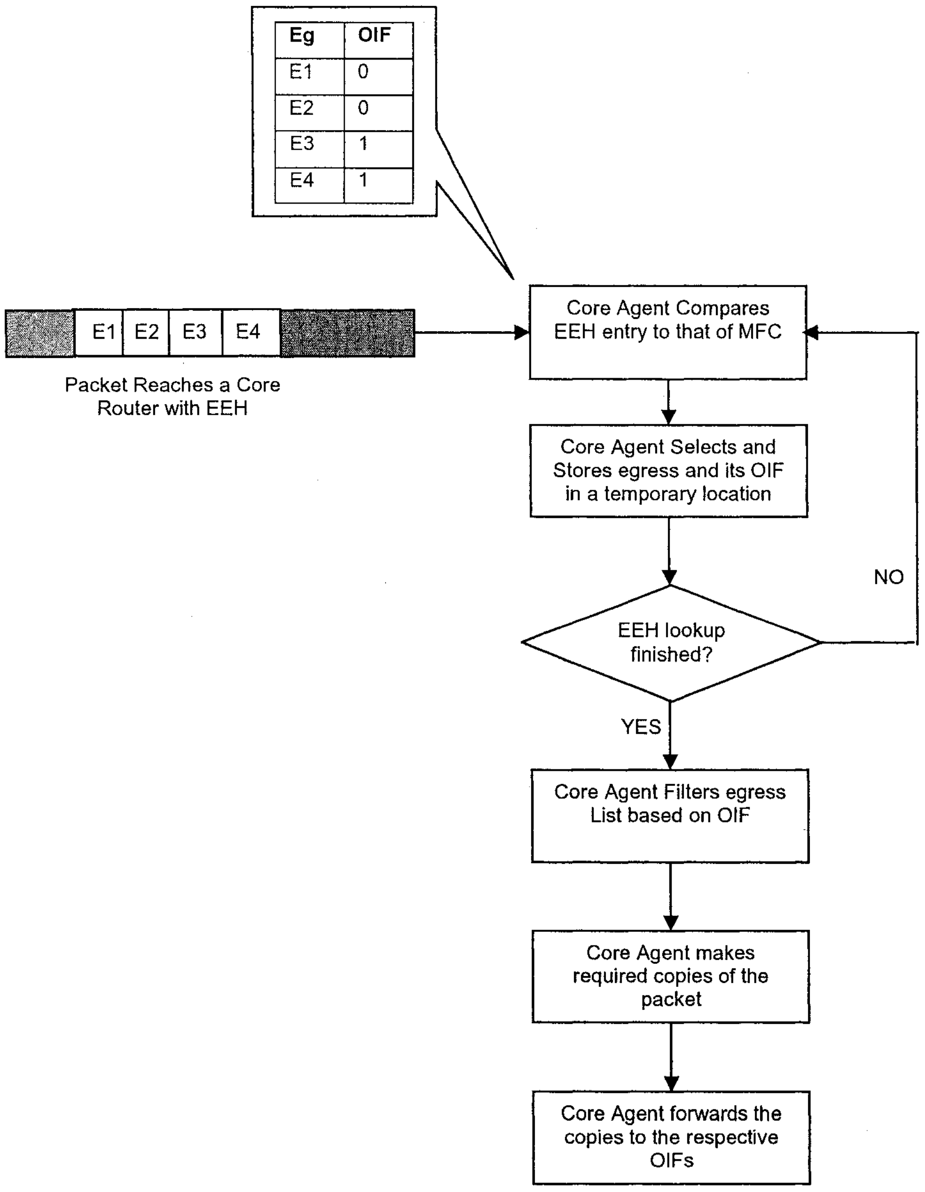
A.1.4 EC-JOIN


QUERY_ACK<S,G,Q> from XBB
Egress Agent generates and forwards upstream the EC_JOIN<S,G,Q,Eg> message
Eg entry in the First hop core MFC?
YES
NO
Egr | OIF
E1 | 0
E2 | 1
Make MFC entry with downstream OIF
Forward EC_JOIN<S,G,Q,Eg> upstream
EC_JOIN in ingress?
NO
YES
Eg entry in ingress database?
YES
Stop and Discard EC_JOIN at ingress
NO
Make Eg entry in the ingress database corresponding to the (S,G)
Ingress forwards EC_JOIN<S,G,Q,Eg> to source

A.1.5 EC-LEAVE


Egress receives LEAVE<S,G> from Rcvr
Last Rcvr with this egress?
NO
YES
Last Rcvr of the (S,G) in this egress?
NO
Delete specific OIF from MFC
YES
Forward EC_LEAVE<S,G,Eg> till the ingress No modification in cores
Src | Grp | OIF
S1 | G1 | 0
S2 | G2 | 1
Delete local MFC entry for this (S,G)
NO
Egress Agent Generates EC_LEAVE<S,G,Eg> and forwards it upstream
EC_LEAVE<S,G,Eg> in ingress?
YES
NO
Delete local MFC entry for Eg and forward EC_LEAVE<S,G.Eg> upstream
Eg | OIF
E1 | 0
E2 | 1
Delete ingress Eg MFC entry
Delete Eg entry also from the EEH list for this (S,G)
NO
Last Eg entry deleted from the ingress database?
Stop and Discard the EC_LEAVE at ingress
YES
Generate and Forward LEAVE <S,G> to the source for updating local database

6. Packet Replication and Forwarding in the Core



Eg
OIF
E1
0
E2
0
E3
1
E4
1
E1
E2
E3
E4
Packet Reaches a Core Router with EEH
Core Agent Compares EEH entry to that of MFC
Core Agent Selects and Stores egress and its OIF in a temporary location
NO
EEH lookup finished?
YES
Core Agent Filters egress List based on OIF
Core Agent makes required copies of the packet
Core Agent forwards the copies to the respective OIFs