Managing QoS for Multimedia Applications in The Differentiated Services Environment

Manish Mahajan
The Applied Software Systems Laboratory (TASSL)
Dept. of Electrical and Computer Engineering, Rutgers University,
94 Brett Road, Piscataway, NJ 08854
Phone No: 732-445-5256
Email: manishm@caip.rutgers.edu

Manish Parashar
The Applied Software Systems Laboratory (TASSL)
Dept. of Electrical and Computer Engineering, Rutgers University,
94 Brett Road, Piscataway, NJ 08854
Phone No: 732-445-5388
Fax No: 732-445-0593
Email: parashar@caip.rutgers.edu

Abstract

The overall quality of network connections has a significant impact on the performance of networked applications. As a result, Quality of Service (QoS) management for networked multimedia applications over IP is a significant and immediate challenge. While differentiated services (diffserv) provide a sense of resource allocation and QoS, they do not guarantee QoS. This paper presents the design, implementation and evaluation of a content-aware bandwidth broker (CABB) that manages QoS for multimedia applications in a diffserv environment. CABB allocates network resources to multimedia flows based on client requirements, the adaptability of the application, and its tolerance to network level parameters such as bandwidth, delay, and latency. It has been implemented and evaluated using the NS-2 simulator toolkit. Evaluations show that CABB improves network resource allocations and increases overall throughput. Furthermore multimedia application flows are better managed and controlled, improving perceived QoS and avoiding possible congestion at core routers.

Keywords: QoS, Content Aware Bandwidth Broker (CABB), Diffserv, Policies, SLA

1. Introduction

This paper presents the design, implementation and evaluation of a content-aware bandwidth broker (CABB) that manages Quality of Service (QoS) for multimedia applications in a differentiated services (diffserv) environment. CABB controls multimedia application flows between diffserv domains using service policies that are defined based on the requirement, nature and adaptability of the flow, and its sensitivity to network level parameters such as bandwidth, delay, and jitter.

The overall quality of network connections (e.g. link capacity, available end-to-end bandwidth, and congestion) has a significant impact on the performance of networked applications. These applications generate traffic at varying rates and have a varying ability to tolerate delays and jitter in the network. Networked multimedia applications, which form a large part of today's Internet traffic, present a significant challenge, as they require quality guarantees from the network [1]. Internet Protocol (IP), however, is best effort and does not provide any QoS guarantees [2] – that is, there are no mechanisms in IP for policing or controlling unresponsive and high bandwidth flows that can cause congestion in the network. As a result, all QoS management is left to the application [3]. Multimedia applications typically have very limited feedback control to prevent them from causing congestion in the network. Consequently, QoS management for networked multimedia applications over IP is a significant and immediate challenge.

Different mechanisms such as Integrated Services (e.g. Resource Reservation Protocol (RSVP) [4]), Differentiated Services [5], and Multi-Protocol Label Switching (MPLS) [6] have been proposed for providing some level of QoS over the best-effort IP network [2]. These mechanisms improve QoS at the edges of the network by identifying and controlling traffic flows or aggregates using classification, conditioning and scheduling techniques. The Differentiated Services provide an approach to IP Quality of Service management that is modular, incrementally deployable, and scalable while introducing minimal complexity [5]. A diffserv domain provides different levels of service (e.g. premium, assured) to meet the client's request for guarantees. Network resources are partitioned between these levels of service, and packets belonging to client flows are marked with specific diffserv code points (DSCP) that map these packets to a particular level of service. This mapping is based on the service level agreement (SLA) between the client and the network service provider. The bandwidth broker manages resources in a diffserv domain. It tracks available network resources and classifies flows using service polices defined based on client requirements and service classes offered by the diffserv domain [7]. In the diffserv architecture, flows are allocated resources without any knowledge of the nature of the application or the information being transmitted. As a result, the broker statically overallocates resources in order to meet the guarantees made to the client. This overallocation wastes resources and causes future requests to be rejected. Furthermore, the broker does not consider the nature of the flow and may allocate resources to rogue flows. These flows can exceed their allocations and "hog" resources, causing congestion and severely affecting the QoS of conforming flows. Thus, while diffserv does provide a sense of resource allocation and QoS, it does not guarantee QoS or eliminate the possibility of congestion.

In this paper we present a *content aware bandwidth broker* (CABB) that provides adaptive brokering for networked multimedia applications. CABB builds on the observation that multimedia applications are flexible or tolerant with respect to network parameters such as packet loss, delay and jitter [8][1]. It exploits this flexibility of multimedia flows to network level parameters to adapt the flows based on the state of network resources and maintain a quantifiable level of QoS despite unfavorable network conditions. For example, when the application's demand for resources exceeds availability, rather than refusing allocation, CABB may admit the application at a reduced QoS level until the required resources become available. This can be a significant advantage for time critical applications. Furthermore, in case of network congestion, CABB can adapt to the network state and reduce QoS rather than completely disrupting application flows. CABB also prevents non-conforming (or rogue) flows from affecting the performance of conforming flows by constantly monitoring and gradually degrading the level of service of the rogue flows. Thus it provides an incentive to support end-to-end congestion control for best effort traffic.

CABB has been implemented and evaluated using the NS-2 simulator toolkit [9]. Results presented in this paper show that by exploiting the flexible nature of multimedia flows, CABB improves network resource allocations leading to better network utilization. The results also show that multimedia flows are better managed and controlled, thereby improving perceived QoS and avoiding possible congestion. Furthermore, flow throughputs increase as CABB enables a larger number of flows to request for resources without over allocation.

The rest of this paper is organized as follows: Section 2 discusses background and related work. Section 3 introduces CABB and describes its architecture, and operations. Section 4 presents an experimental evaluation of CABB. Section 5 summarizes the results. Section 6 presents the conclusion.

2. Background and Related Work

Different approaches have been proposed to provide service guarantees to multimedia applications. These approaches can be broadly divided into network level protocols, reservation-based schemes, and adaptation-based schemes for QoS support. Network level protocols provide QoS support by interpreting the application's requirements in terms of network parameters, and enhancing network switches to service application flows or flow aggregates. Reservation-based schemes reserve network/system resources based

[1] Intolerant applications will be handled using the standard diffserv mechanism that provides guarantees.

on the application's requirements. These schemes are typically accompanied by admission control mechanisms that match application requests with existing resource availability. Adaptation-based schemes utilize the adaptive behavior of applications that do not require hard service guarantees, and perform application aware active resource management with runtime adaptations to provide better than best effort service. Finally, various optimization schemes have been proposed for monitoring applications, integrating application processing with data transport, and embedding QoS state information in application data to improve performance of applications requiring service guarantees. A brief description of these approaches is presented below.

2.1. Network Protocols with QoS Support

The Internet Engineering Task Force (IETF) has addressed the issues in building a QoS support infrastructure to be deployed over IP on the Internet. Several working groups have proposed promising alternatives. However, considering the scale and the increasing heterogeneity of the Internet, a single solution is still elusive. The most prominent proposals are Integrated Services, Differentiated Services (diffserv) and Multi Protocol Label Switching (MPLS).

The Integrated Services architecture provides QoS to individual applications or flows based on explicit resource requests [4]. It specifies a number of service classes designed to meet the needs of various networked applications. In principle it is a significant deviation from the highly successful and scalable best effort IP. Differentiated Services (diffserv) is a set of technologies that are used to provide QoS in a world of best effort service provisions [5]. In diffserv, all the complexities are pushed out to the edge routers, keeping core routers simple. As it exhibits greater flexibility and is able to allocate resources efficiently while still providing service guarantees, diffserv is well suited for providing network-level adaptive QoS support to distributed applications operating in heterogeneous networks. Multi Protocol Label Switching (MPLS) [6] is a traffic engineering protocol. It provides resource management for flow aggregates using network routing control based on fixed length 'labels' in packet headers. As MPLS is a protocol-independent mechanism resident in network level switches with no application control, technologies such as diffserv can readily leverage the management support provided by MPLS.

2.2. Reservation-Based Schemes

Intuitively, the most straightforward method for assuring service guarantees is to reserve resources according to an application's request. However, reservation based schemes involve admission control algorithms, reservation protocols, monitoring protocols, and signaling mechanisms. These schemes are generally complex and require extensive network support. This affects their scalability and robustness. Nevertheless a number of innovative approaches have been proposed that address key technical challenges in achieving QoS on the Internet. Some of these are discussed below:

In [10], Chandra et al. have an intelligent network architecture based on service brokers that attempts to provide application-oriented reservations. Service brokers perform resource discovery and optimization of resource usage using application domain knowledge. An application-oriented signaling protocol handles the flow for an application. A hierarchical brokerage and resource management structure is suggested to handle resource allocation and sharing.

The multi-resource reservation algorithm proposed in [11] utilizes the resource broker model for an integrated approach to reserving and scheduling resources with low contention. It adopts a component-based approach with Resource Brokers, QoS Proxies and service components as the main entities. A resource reservation plan reserves a minimum amount of bottleneck resources while deciding appropriate levels of input and output quality for each service component. Simulations show that the proposed algorithm works better, in terms of reservation success rate, than a random reservation path selection.

[12] presents the GARA resource management architecture, which addresses the challenges in achieving end-to-end QoS guarantees across heterogeneous collections of shared resources. GARA builds on existing techniques and concepts to support end-to-end discovery, reservation, allocation and management of heterogeneous ensembles of computers, networks, storage-systems, and other resources under independent local control. GARA exposes both reservations and objects as first-class, abstract objects, defines uniform representations and operations for diverse resource types, and uses an information service to reveal site-specific policies. These constructs enable the construction of reusable co-reservation and co-allocation agents, which can combine domain- and resource-specific knowledge to discover, reserve, and allocate resources that meet application QoS requirements.

2.3. Adaptation-Based Schemes

An alternative approach to reservation-based schemes is to adapt an application so that it performs satisfactorily given existing resource constraints. These schemes are feasible only for applications that are capable of adaptations, such as multimedia applications. Although adaptation-based schemes can be complex, the complexity is typically at the middleware or application level and at the end-hosts. As a result, the scalability of the approach is not adversely affected.

Applications that can deal with changes in the network environment are called network-aware. A network-aware application attempts to adjust its resource demands in response to network performance variations. In [13], J. Bollinger and T. Gross present a framework-based approach for the construction of network-aware programs. The framework investigates techniques to measure dynamic changes in network service quality and further, to map application-centric quality measures (e.g., predictability) to network-centric quality measures (e.g., QoS models that focus on bandwidth or latency). A layered architecture with an adaptation layer that implements feedback based on network resource availability is proposed.

In [14], Chang et al. present an application-independent adaptation framework with a tenability interface and a virtual execution environment. Together, the two mechanisms enable the development of a run-time adaptation system. This system continuously monitors resource conditions and application progress (in terms of user preferences of QoS metrics), and automatically determines both, *when* adaptation should be performed and *how* the application must be modified (i.e., which of its configurations must be chosen) based on application profiles.

2.4. Miscellaneous Optimization Schemes

Several QoS and optimization schemes have been proposed to enhance the Internet infrastructure for better QoS support. Such schemes involve monitoring tools for obtaining network and system resource state in real time, integrating media encoding and transport for better adaptability support, and using QoS state information in routing decisions. Some of these approaches are summarized below.

A QoS broker that orchestrates resources at the end-points and coordinates resource management across layer boundaries is proposed in [15]. As an intermediary, it hides implementation details from applications and per-layer resource managers. The broker uses services such as translation, admission, and negotiation to properly configure the system to application needs.

In [16], Zhang et al. present an end-to-end transport architecture for multimedia streaming over the Internet. They propose a new multimedia streaming TCP-friendly protocol (MSTFP) that combines forward estimation of network conditions with information feedback control to optimally track the network conditions. This scheme improves end-to-end QoS by allocating resources according to network status and media characteristics.

An approach to managing real-time traffic in multimedia networks is presented in [17]. The respective roles of the real-time control system, the management system, and the network operator, are clarified and their interactions aimed at managing real-time services are described. It introduces an architecture based on the concept of managing network services by tuning the resource control tasks in the network control system. The L-E model to deal with the complexity of the network control system, a generic system-level abstraction of a resource control task, is presented.

[18] reviews the main requirements and challenges for effective management of multimedia networks. It presents a case study of a thin-client based multimedia system called CSL (Computer-supported Learning System). In the framework proposed, most of the link management information will be kept in a Management Information Base (MIB) in the network and a very simple MIB will be maintained in the thin client. The MIB in the network is accessible to the network management application, and a lightweight protocol is proposed for the updating of the network MIB using an agent at the thin client.

Table 1, gives an overview of existing approaches to adaptive QoS at various levels i.e. network, application, middleware etc. Note that it is not exhaustive nor do we propose to present a taxonomy of QoS management approaches.

Table I: QoS Management for Multimedia Applications

Network Protocols with QoS support	These approaches provide QoS support by interpreting the application's requirement in terms of network parameters and enhancing the network switches to service application flows or flow aggregates according to the assigned service levels. Examples include Integrated Services [5], Differentiated Services [4], Multi Protocol Label Switching [6].
Reservation Based Schemes	These approaches reserve network/system resources based on the application's requirements, and are typically accompanied by admission control schemes that check if the application's request can be serviced with the existing resource availability [10][11][12].
Adaptation Based Schemes	These approaches utilize the adaptive behavior of applications that do not require hard service guarantees, and provide a better than best effort service by performing application aware active resource management and runtime adaptations [13][14].
Miscellaneous Optimization Schemes	Miscellaneous optimization schemes for monitoring, integrating application processing with data transport and embedding QoS state information in application data to achieve a better performance for applications requiring service guarantees [15] [16] [17] [18].

3. Content Aware Bandwidth Broker: Architecture, Policies and Operations

This section presents the architecture, operations, policies and implementation of the content aware bandwidth broker (CABB). The following section summarizes the diffserv approach and the traditional bandwidth broker architecture.

3.1. Diffserv and the Bandwidth Broker Architecture

Diffserv classifies the network traffic and allocates the network resources to flows based on a management policy. Service classes are created with different QoS guarantees and flows are assigned these classes. Service Level Agreements (SLA) are set up between adjacent diffserv domains. The SLA establishes policy criteria and defines the traffic profile to be maintained by independently managed domains. As shown in Figure 1, packet classification is done based on one or more bits in the packet at the ingress router. The packet is then marked, using diffserv code points (DSCP), as belonging to one of the many service classes and injected into the network. All packets with the same code point are grouped together and are known as a behavior (flow) aggregate. DSCP determines an aggregate's per hop behavior (PHB), i.e. the treatment given to the aggregate at routers on the way to its destination. Core routers examine the DSCP to decide how to forward the packet. Most of the work in this scheme is done at the edge routers. The edge routers are responsible for classifying and conditioning packets using a multifield classifier, flow markers, traffic meters, traffic shaper and policers [5].

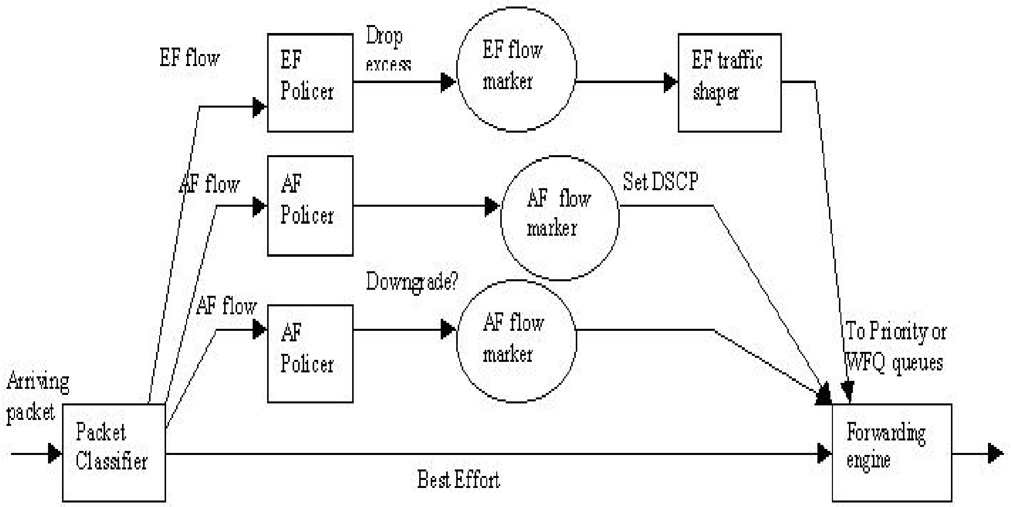


Figure 1: Diffserv edge router functionality: traffic classification and conditioning

There are two defined PHBs: expedited forwarding (EF), and assured forwarding (AF) [19]. EF PHB supports low loss, low delay, and low jitter. EF PHB is assigned to flows that cannot tolerate QoS degradation. AF PHB defines four relative classes of service and each service supports three levels of drop precedence (i.e. a total of twelve code points). The conditioner at an edge router shapes and smoothes bursts of EF traffic before injecting it into network. EF traffic not conforming to its SLA is termed “out-of-profile”, and is dropped. AF traffic conforming to the SLA is termed “within profile”, and is delivered with a higher probability than out-of-profile AF traffic. Out-of-profile traffic may be demoted or downgraded to a reduced bandwidth allocation instead of being dropped. Thus diffserv provides simple and coarse mechanism for QoS support.

As shown in Figure 2, a Bandwidth Broker (BB) is an agent that controls the resources within a diffserv domain, using service policies defined by client requirements. SLAs are used to define the relation between policies and PHBs. A service provisioning policy (SPP) indicates the configuration of traffic conditioners in the edge routers of the diffserv domain and the mapping of traffic streams to diffserv behavior aggregates. The BB requires both, the SLAs and the SPPs to achieve the range of user services. The broker uses the SLA to determine if it can provide the allocation and configures the particular edge router to classify and mark the packets accordingly [5,7].

The bandwidth broker has been designed to add intelligence to the diffserv and better utilize the existing resources. But it wastes resources by statically overallocating them to meet client guarantees. Furthermore, it may allocate resources for rogue flows severely affecting the QoS of guaranteed or conforming flows. For example, a flow that is continuously bursty after promising constant rate may be considered a rogue flow. Thus, bandwidth broker along with diffserv does not guarantee QoS or eliminate the possibility of congestion.

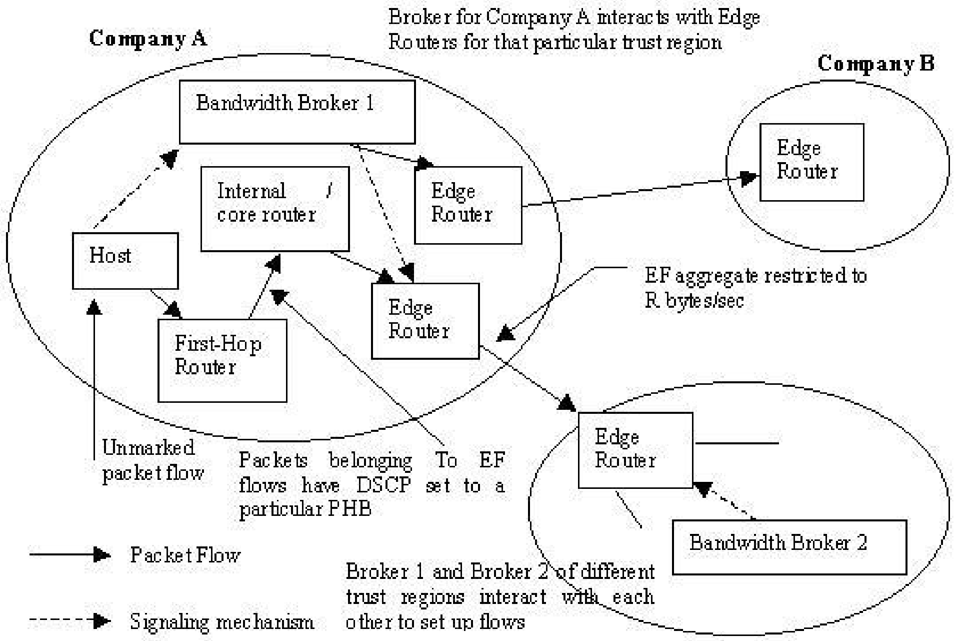

Broker for Company A interacts with Edge Routers for that particular trust region
Company A
Company B
Bandwidth Broker 1
Internal / core router
Edge Router
Edge Router
Host
First-Hop Router
Edge Router
EF aggregate restricted to R bytes/sec
Edge Router
Bandwidth Broker 2
Unmarked packet flow
Packets belonging To EF flows have DSCP set to a particular PHB
Packet Flow
Signaling mechanism
Broker 1 and Broker 2 of different trust regions interact with each other to set up flows

Figure 2: Bandwidth broker in a diffserv domain

3.2. CABB Architecture

The CABB presented in this paper uses the bandwidth broker architecture defined by the Qbone team [7] [20] [21]. The key components of this architecture include User Interface, Inter-domain Interactions, Intra-domain Interactions, Routing Table, Database and Policy Services.

CABB extends on this architecture by adding content aware policies. These policy engines define specific policies for the various service classes offered by the diffserv domain. These policies consider the nature of the request and the flexibility of application with respect to network level parameters (e.g. delay, jitter, and loss) while resolving resource requirements to adaptively manage application QoS. CABB uses these policy engines to configure functionality of edge routers in the diffserv domain [7][20].

3.3. CABB Operations and Policies

A CABB may receive a resource allocation request from one of two sources: a request from an element in the domain that the broker controls, or a request from a peer CABB. It either confirms or denies the request and responds accordingly. Additionally, it may generate request messages for downstream resources. CABB consists of four components, viz. a resource allocator, a database to store the parameters required to make the reservation decisions, a broker manager for inter-broker communication, and a policy engine to create a particular policy for the available service classes. We have three different service classes in our implementation: EF, AF and Best-Effort.

As described above, networked multimedia applications are flexible, i.e. they can tolerate resource scarcity (within certain limitations) by reducing performance, and can utilize additional resources to improve performance. Furthermore, they can sacrifice the quality/performance of some parameters in order to preserve (or improve) the quality/performance of more critical parameters [22]. CABB maps this flexibility into various network level parameters to maintain an acceptable QoS range for an application. This is done such that the end user perceives a quantifiable QoS even if all requested resources are not allocated.

When an application requests CABB for network resources, it invokes an API provided by CABB and passes application specific QoS requirements and constraints. These parameters can include quality descriptions for the specific media characteristics (e.g. height, width, and color specification in a video stream), media sample rate, priority/criticality, and transmission characteristics requirements for end-to-end delivery (e.g. end-to-end delay bounds). On receiving such an application flow request along with QoS parameters, CABB translates the parameters into network resource requirements such as rate usage profiles (i.e. committed and peak information rate), burst, delay, flexibility to delay, jitter and loss, for the time during which the created profile is to be active. The translation builds on heuristics presented in [23][24][22][3] that are based on application flexibility, and provide mechanisms to translate the multimedia application level parameters such as frame rate, frame size, etc to network resource requests and constraints, such as rate usage profiles, burst, and delay. Application data transport requirements are identified based on frame size and frame rate. For example, video resolution (pixels per frame) and encoding (bits per pixel) define the frame size. Frame rate for video can be typically altered from 30 frames per second for very good quality to as low as 15 frames per second for an acceptable quality. The bit rate encoding for the audio signal determines its fidelity and frame size. Audio signal are highly sensitive to delay and jitter and hence have a smaller range of permissible frame rates. Collectively, frame size and frame rate modifications translate into a range of bandwidth requirements for each media type for acceptable performance [3]. Once the translation to network requirements is complete, CABB tracks current allocation of marked traffic [25], and interprets the new request in light of defined policies and available resources. It looks up its policy table for the existing policy that governs the requesting host and contains the relevant service level agreements (SLA), service/DSCP mappings, management information, current reservations/allocations, and edge router configurations.

These parameters along with the application's flexibility [22, 26] are passed to the policy engine defined by the SLA. The policy engine further translates the parameters into specific network actions such as bandwidth management (allocated transmission rate), queuing (per hop behavior), buffer space management for queuing, network monitoring, and accounting. In our implementation, rate usage profiles are pre-specified for simulated applications. The policies determine the users' bandwidth requirements and allocate resources accordingly. Note that an application's flexibility is used to determine its minimum resource requirements [22]. As will be shown in the experimental evaluation presented in Section 4, resource allocation is done in a succinct and organized fashion and treats all flows fairly. Furthermore, the allocation is based on the client's needs as defined by its QoS parameters, rather than its peak request that is typically larger. This conserves resources and allows a larger number of requests to be served even though all the requested resources are not available.

The EF policy is outlined as follows. The diffserv router uses two parameters to police EF flows: committed information rate (CIR) and peak information rate (PIR). The application generates data at an average rate of CIR and can go to peak rates of PIR. Although it reserves PIR as its peak bandwidth it may not generate data at PIR all the time during its duration. If the flow exceeds PIR at any time, it goes out-of-profile and will be downgraded to a lower DSCP with reduced bandwidth and eventually dropped. The CABB invokes its EF service policy manager to decide the result of a given request. It does a query on the edge router(s) from source to destination in its domain and waits for a reply. After receiving the reply, CABB decides whether the edge router(s) has sufficient resources to handle the flow or not. If available resources are not sufficient, i.e. the available bandwidth is less than CIR or between CIR and PIR, CABB uses the flows' flexibility, represented as a flexibility number, to determine its minimum resource requirements. It then retries to set up the flow with reduced resource requirement. The CABB API return call then informs the application of the status of current reservation/marking. It provides a set of alternate reduced parameters if resource request fails.

The diffserv router uses two parameters to police AF flows: committed information rate (CIR) and committed bucket size (CBS), which is a token bucket filter. The flow reserves CIR as its peak bandwidth. If it exceeds CIR at any time, it will be downgraded to a lower DSCP with reduced bandwidth and eventually dropped. The CABB invokes its AF service policy manager to decide the result of a given request. It does a query on the edge router(s) from source to destination in its domain and waits for a reply. After receiving the reply, CABB decides whether the edge router(s) has sufficient resources to handle the flow or not. If the resources are not sufficient i.e. the available bandwidth is less than CIR, CABB uses the flows' flexibility number to determine its minimum resource requirements. It then retries to set up the flow with reduced resource requirement. The CABB API return call then informs the application of the status of current reservation/marking. It provides a set of alternate reduced parameters if resource request fails, e.g. required resources are not available, and the application decides to retry with reduced parameters. On allocating the flow, CABB updates the parameters at edge router(s) along the path from source to destination.

The CABB policy engines essentially leverage the flexibility of multimedia applications and use it to effectively map them to available service levels to achieve better admission control [22]. A flow is rejected only if available resources remain insufficient even after reducing the level of service. Furthermore, with the help of the broker manager, CABB can generate additional request messages to reserve resources downstream [7,15]. Each downstream CABB takes into account the inter-domain traffic SLA before allocating resources. There is a limit on the number of flow reservations allowed per service class, which ensures good bandwidth utilization [5]. For a successful admission, the CABB makes appropriate reservations (usable bandwidth, buffers) using the resource allocator and assigns a DSCP to the application flow. Diffserv-enabled routers use the DSCP marking to provide the requisite PHB. The DSCP mappings may be unique to each router but the PHB is the same for all diffserv-enabled routers. CABB then schedules the flow to the appropriate queue manager (Priority or Weighted Fair Queue) and updates the policy table. Diffserv's internal policing mechanisms then force the flows to adhere to the agreed policy. CABB prevents non-conforming (or rogue) flows from affecting the performance of conforming flows by constantly monitoring network condition and gradually degrading the service of rogue flows. Thus it provides the incentive in support of end-to-end congestion control for best effort traffic. A highly congested network may stop best effort but will allow high priority traffic (EF service class) to go through. Hence even though it is congested the users of high priority service class don't see the congestion. The CABB may also maintain a log of a flow's history to help it to identify and curb rogue flows.

Networked multimedia application flows are usually bursty and it is difficult to define the characteristics of such a flow [2]. CABB uses the minimum information given by such a flow (such as committed and peak information rate), the ranked importance of an application, the importance of the user, the current available resources, the result of inter-broker communication and the tolerant adaptability of the application to network level parameters to allocate the flow. It achieves better admission control by implementing content aware policy engines.

3.4. An Illustrative Example

The operation of CABB is illustrated using the sample diffserv configuration shown in Figure 3. This network includes two diffserv domains, two sources in domain 1 and a destination in domain 2. The two diffserv domains are required to show the inter-domain interaction between the broker agents to provide end-to-end resource allocation for a source-destination pair. We assume static routes in this example.

Source 1 in diffserv domain 1 (DS1) requests a flow setup to destination 2 in diffserv domain 2 (DS2). It provides its requirements such as average and peak transmission rates, delay and jitter, along with the flow type (e.g. multimedia audio) to CABB1 in DS1. CABB1 uses its policy engines to determine the required bandwidth and PHB based on the agreed upon SLA, the current available resources, and the flexibility of the application. The flexibility for various multimedia applications is encoded into the policy engine as a flexibility number based on their tolerance to the above mentioned network level parameters [22]. CABB1 forwards the flow's parameters, either as is or reduced, to the downstream broker CABB2 in DS2. On receiving a positive acknowledgement from CABB2, it assigns a DSCP to the traffic flow between this source-destination pair in its policy table. If the available resources are insufficient or CABB2 returns a negative acknowledgement, CABB1 re-attempts the flow setup by assigning it a lower but still acceptable quality and correspondingly lower DSCP. It also informs the application of this adaptation.

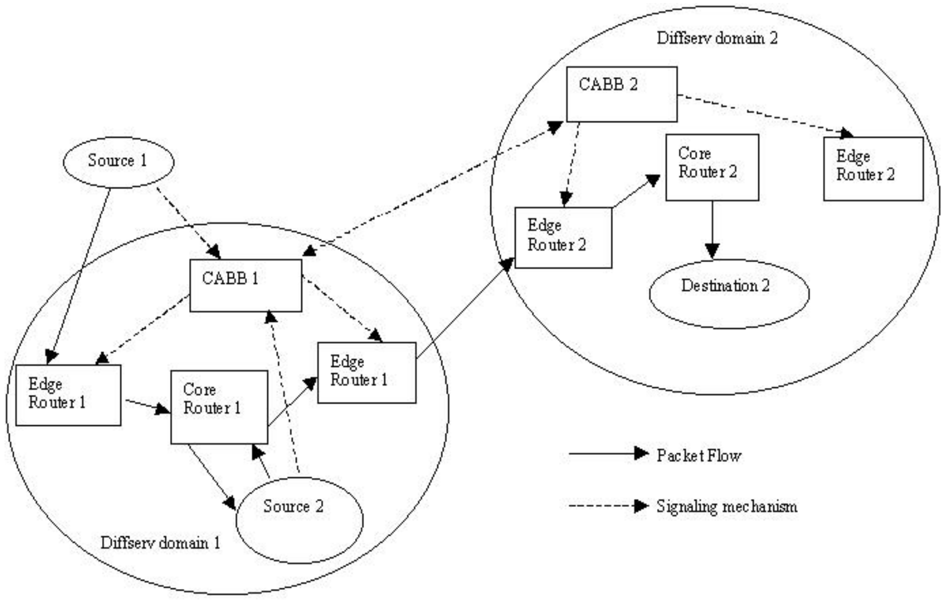


Figure 3: Sample Differentiated Services Configuration

Source 2 in DS1 requests a multimedia video flow also to destination 2 and provides its parameters such as peak transmission rates, delay, jitter, packet loss and required resources. As before, CABB1 invokes its policy engines to determine the resources to be allocated. Note that the flexibility number will be different in this case as multimedia video is more flexible to packet loss than audio. It once again sets up the flow after receiving a positive acknowledgement from CABB2. In this example, Source 1 is given EF PHB and Source 2 is given AF PHB. The timing diagram in Figure 4 shows the steps involved in setting up a connection when a flow requests transmission. It describes the sequence of interactions after the Host (Source 1) initiates transmission request to Dest (Destination 2).

The CABB, prevents congestion by effectively ordering the flows in such a manner that they conform to the traffic profile as agreed in the SLA. An appropriate number of flows are allowed at a time so that the network is not unduly loaded. We can now control multimedia traffic at the edge router; i.e. keep the core simple and move all complexity to the edge router. The CABB ensures that all applications get a fair share of bandwidth according to their SLA and that the bandwidth for various service classes is efficiently utilized. When the network is congested, the CABB may allow a multimedia application to go through, but with reduced service. As CABB is content aware and knows the flexibility of the applications, it takes it into account while allocating flow rate. Note that it is finally up to the application to accept the reduced service or not. The application may opt to retry later to get a richer set of services. In case of premium service with EF flows, the broker guarantees that this flow will not face any queuing delay along the way and will be delivered within the constraints of its QoS parameters, giving it the impression of a virtual leased line. Such flows will rarely experience any packet drops (early or late) at the routers.

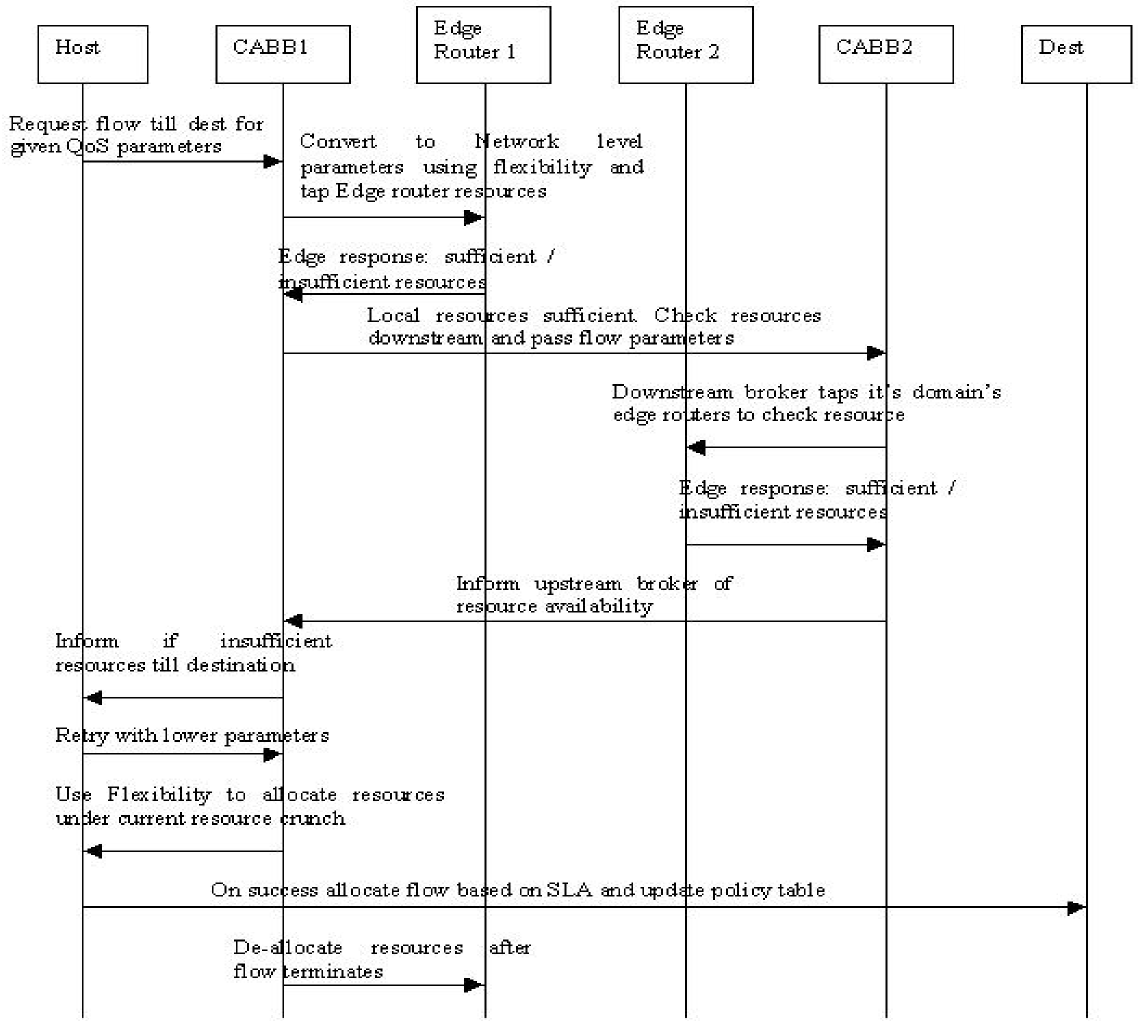

Host
CABB1
Edge Router 1
Edge Router 2
CABB2
Dest
Request flow till dest for given QoS parameters
Convert to Network level parameters using flexibility and tap Edge router resources
Edge response: sufficient / insufficient resources
Local resources sufficient. Check resources downstream and pass flow parameters
Downstream broker taps it's domain's edge routers to check resource
Edge response: sufficient / insufficient resources
Inform upstream broker of resource availability
Inform if insufficient resources till destination
Retry with lower parameters
Use Flexibility to allocate resources under current resource crunch
On success allocate flow based on SLA and update policy table
De-allocate resources after flow terminates

Figure 4: CABB timing sequence diagram.

From the results in Section 4, we see that queue management and diffserv policing work in such a manner that in the case of a non-conforming flow, only that particular queue to which the flow belongs will be penalized. The other flow aggregates are not affected. If for any reason the broker is not able to allow a flow, it will log a report indicating the flow's requirements, source, destination and time the flow was requested.

3.5. CABB Implementation

We have implemented the bandwidth broker on the Network Simulator-2 (NS-2) toolkit. The NS-2 toolkit has substantial functionality for simulating different network topologies and traffic models. NS-2 also has an open architecture that allows users to add new functionalities. We implemented a scheduler that allowed us to implement and service various types of queues for EF, AF and Best-Effort service classes. This scheduler enables setting up and serving various FIFO queues such as non-preemptive priority queues, weighted round-robin queues and best effort queues for the various edge routers. Using the diffserv patch provided by Nortel Networks [9] and extending it with our scheduler and the CABB, we can generate diffserv domains and create suitable test networks as shown in the experimental evaluation in Section 4.

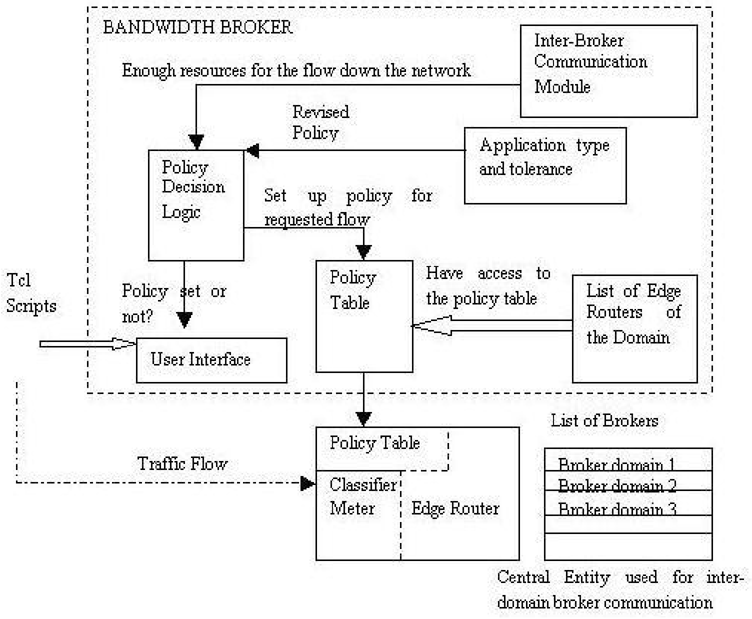

BANDWIDTH BROKER
Inter-Broker Communication Module
Enough resources for the flow down the network
Revised Policy
Application type and tolerance
Policy Decision Logic
Set up policy for requested flow
Policy Table
Have access to the policy table
List of Edge Routers of the Domain
Tcl Scripts
Policy set or not?
User Interface
List of Brokers
Policy Table
Traffic Flow
Classifier Meter
Edge Router
Broker domain 1
Broker domain 2
Broker domain 3
Central Entity used for inter-domain broker communication

Figure 5: CABB internal block diagram and interactions

The diffserv implementation has three modules, two of them are related to the edge router and core routers, and the third module is the policy and resource manager. The policy class handles the creation, manipulation, and enforcement of edge router policies. A policy defines the treatment that packets will receive at an edge router. Policies are set using Tcl scripts [9]. The policy class uses a policy table to store the parameter values. The packet that arrives at the edge router is classified to decide to which traffic aggregate it belongs, and a specified meter is used to check the average traffic rate of that client to make sure it corresponds to the current sending rate. If it does not conform, it gets downgraded to a lower DSCP.

As shown in Figure 5, the CABB is used to configure the policy module of the diffserv. Our CABB implementation consists of four modules: user interface module through which the user/network operator can allocate resources, a database module that stores all the parameters required to make the reservation decisions, a service policy manager module, and a central entity called the broker manager that handles inter-broker communication. The policy manager module creates a particular policy module for EF, AF or Best-Effort flows and updates the policy at all the other edge routers in its domain.

The CABB makes the provisioning based on the SLA agreed upon by client/user (through the user interface module) using Tcl scripts and in compliance with other parameters in the database module. These include the current reservations and the router configurations. Configuration changes are made to the policy module. Within the policy module every source-destination flow is associated with a policy type, meter type, current rate of traffic (the rate agreed upon with the client) and other policer specific parameters. The CABB's policy table has only one entry for a particular source-destination pair. Any change in that entry is then communicated to all the edge-routers in that domain. We associate a set of DSCPs with each flow. Each DSCP corresponds to a different traffic rate and PHB and is internally implemented as a specific queue that will serve the packet.

To aid the broker in it's DSCP mapping, we have implemented a priority queue for EF PHB (also called EF code points) and weighted round-robin queues for AF and Best-Effort PHB (also called AF and Best-Effort code points respectively). The key idea is that EF flows should receive guaranteed service, and should not face any queuing delay in their path to the destination. A priority queue staves off other traffic while it is being served. The diffserv-capable edge router also takes care of shaping and policing the EF flows which ensures that the flows downstream confirm to their profile. The weighted round robin queues are serviced according to their weights. We put higher weights for AF flows and lower weights for Best-Effort flows.

For interbroker communication, the CABB looks up its database and routing table to identify the downstream edge router and peer broker. It communicates the flows' parameters to the peer broker. The downstream broker then uses these parameters along with the link's latest characteristics and resources to decide whether to allow the flow to continue or not. If the allocation is not successful, it informs the broker upstream. The flow is then allowed to retry once it is rejected. If the allocation is successful, this process continues on another downstream edge router and broker till the destination is reached. This is a one-way communication from host to destination and does not imply that the destination has resources reserved at the same time. CABB gives us the advantage of setting up a flow using RSVP type signaling without the overheads of keeping and refreshing up-to date network state information.

4. Experimental Evaluation

In this section we present an experimental evaluation of the performance and effectiveness of CABB. The evaluation consisted of a series of simulations using different network topologies. The simulations are designed to evaluate the performance and effectiveness of CABB under various network conditions. They evaluate CABB adaptations in cases of congested links, out-of-profile clients and insufficient resources. The evaluations concentrate on multimedia flows and uses different applications types such as audio, video, and video conferencing. The network topologies consist of multiple source nodes, heterogeneous links and a single sink node.

Table II – Notations

Notation	Explanation
PIR	Peak Information Rate: Peak rate of transmission.
CIR	Committed Information Rate: Average rate of transmission.
CBS	Committed Burst Size: This parameter is used by AF policer in conjunction with CIR.
ABW	Available Bandwidth: Bandwidth currently unused and available.
UBW	Usable Bandwidth: Actual allocated bandwidth.

The notations used in the discussion below are summarized in Table 2. Early drops follow the RED algorithm [8], while late drops occur when packet arrival exceeds buffer size. We define -DSCP 10 for EF PHB and DSCP 21 for AF PHB. For our experimental purposes, in the case of non-conforming flows, DSCP 10 may be downgraded (i.e. allocated reduced bandwidth) to DSCP 11. Similarly DSCP 21 and 23 may be downgraded to DSCP 22 and 24 respectively. Simple policies for EF and AF flows are show in Table 3. We allocate 1/3rd bandwidth to EF flows, 5/12th bandwidth to AF flows and the rest (1/4th) to best effort in all the topologies given below [8]. These simple rules/policies, which are centralized at the level of CABB, effectively provide it control over flow allocations while being fair to all flow types.

Table III: EF and AF Policies

EF Policy	AF Policy
If (ABW > PIR), __UBW = function_1(PIR) If (CIR < ABW < PIR), __UBW = function_2(flexibility, CIR, PIR), If (ABW < CIR || UBW) retry with reduced parameters	If (ABW > CIR), __UBW = function_3 (CIR, CBS) If (ABW < CIR), __UBW = function_4 (flexibility, CIR, CBS). If (ABW < UBW), retry with reduced parameters

The CABB processes requests on a first come first serve. Requests are evaluated on the basis of their flexibility, SLA, the availability of resources and interactions with peer brokers, and are either allocated the requested resources, allocated reduced resources or rejected. After the request is successfully processed, it is assigned a DSCP, which is then used by the diffserv-enabled router to classify and monitor the associated flow. In what follows, we present experimental results for 3 different network topologies. The results are for 80 simulation timesteps. Each set of results consists of a pair of plots for each source, one showing the number of transmitted packets and the other showing the number of dropped packets. Graphs 2 and 3 are for the EF source (DSCP 10) and Graphs 4 and 5 (and possible 6 and 7) are for the AF source(s) (DHCP 21, 22). The table in the figures lists the diffserv parameters and transmission rates for the different sources. The notation used in the figures is as follows: cp –DSCP; TotPkts – Total packets, TxPkts – Transmitted packets; edrops – Early Drops and ldrops – Late Drops at the core router (Core).

4.1. Network Topology I

The first network topology evaluated is shown in Figure 6 and consists of – three routers (one core and two edge), and a CABB that manages and configures the edge routers. The goal of this experiment is to demonstrate the operation of CABB and its ability to effectively manage resource allocation and to limit the effect of unresponsive EF and/or AF flows. The experiments consist of three sources (S0, S1, and S2) and one sink (Dest). The three flows are: (1) *S0-Dest*: audio-- EF PHB, (2) *S1-Dest*: video – AF PHB, (3) *S2-Dest*: video conference – AF PHB. The *Core- E2* link is the bottleneck link with a capacity of 5 Mb. The edge routers *E1*, *E2* have two queues, one priority and one WFQ.

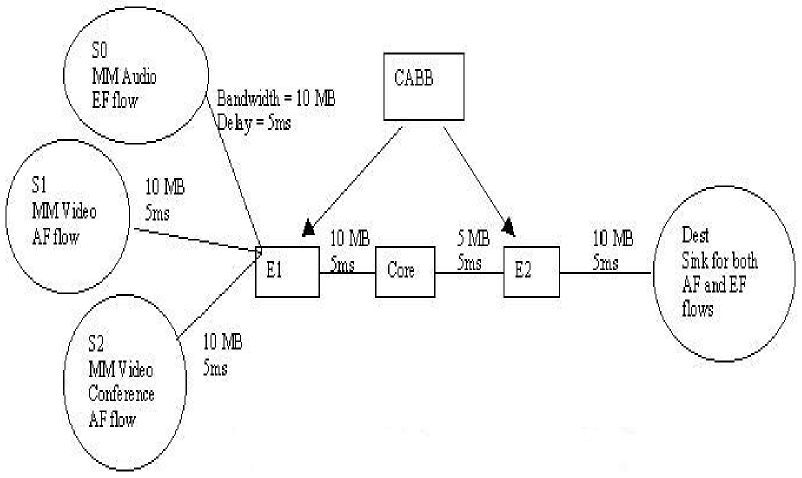
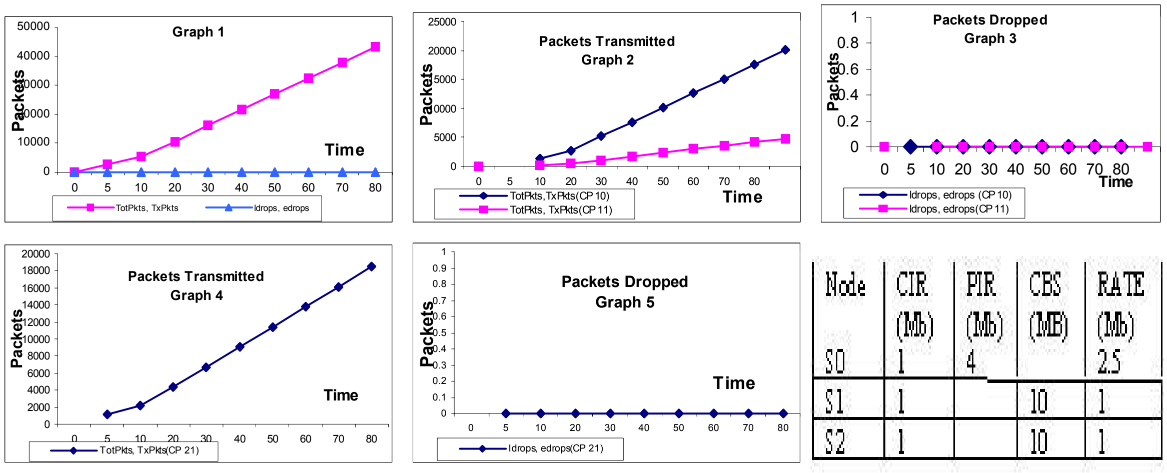


Figure 6: Topology 1 (1 EF, 2 AF flows)

4.1.1. Experiment 1

In this experiment, S0, S1 and S2 request 4mb (PIR), 1mb (CIR), and 1mb (CIR) and transmit at 2.5mb, 1mb, and 1mb respectively. The results from this experiment are plotted in Figure 7. As $Rate_{S0}$ > UBW_{S0} (3.33mb), S0's packets are downgraded to CP 11 and it is allocated a reduced bandwidth (2 Mb).

This is shown in graphs 2 and 3 in Figure 7. S1 and S2 are allocated the resource they request, seen in graphs 3 and 4. For example, if S1 streams a movie containing uncompressed low-quality images of 320 x 240 pixels, each encoded by a single byte, leading to video data units of 76,800 bytes each. Assume that images are to be displayed at 30 Hz, or one image every 33 msec. It produces data packets at the rate of 615 Kbps. This rate confirms to its profile and packets belonging to this flow are never downgraded. Furthermore, we see from Figure 7 that S0 (audio in this case) goes out-of-profile and gets downgraded without affecting the other AF flows. This shows that CABB prevents the out-of-profile EF flow from defecting (downgrading or dropping) the in-profile AF flows from S1 and S2 at the edge router. This improves the transfer delay that would have resulted without CABB due to congestion at upstream routers caused by unchecked flows.



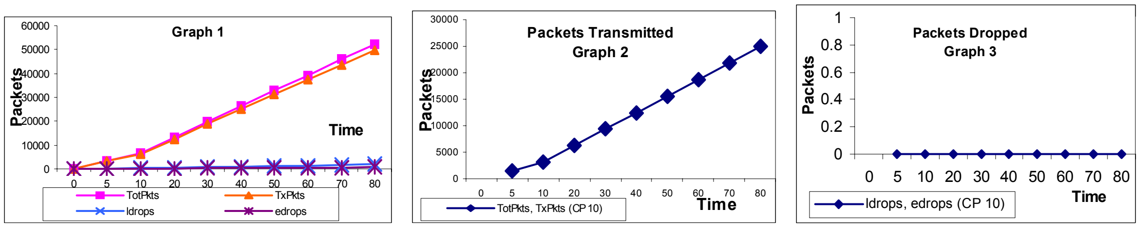
Node	CIR (Mb)	PIR (Mb)	CBS (MB)	RATE (Mb)
S0	1	4		2.5
S1	1		10	1
S2	1		10	1

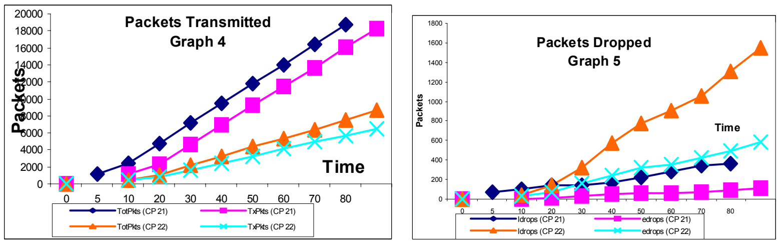
Figure 7. Results of experiment 1 (topology 1)

4.1.2. Experiment 2

In this experiment, S0, S1, S2 request 3mb, 1mb, and 1mb and transmit at 2.5mb, 2mb, and 1mb respectively. The results from this experiment are plotted in Figure 8. As can be seen from these plots, all three sources are allocated full resources. S0 is in-profile and is never downgraded as seen in graphs 2 and 3. S1 goes out-of-profile and its AF aggregate packets are downgraded to CP22 and are eventually dropped (see graphs 4 and 5). As graph 5 shows, edrops are fewer than the corresponding ldrops. Finally, we observe that the out-of-profile AF flow does not affect the in-profile EF flow. Here, once again, CABB improves throughput and performance of conforming multimedia applications while gracefully degrading the performance of rogue flows. For example, if S0 was a critical audio application and S1 was a non-critical video (rogue) application, CABB's polices avoid the video application to hog network resources by gracefully degrading its performance and later dropping the video flow's packets. Audio packets from S0, which remain in-profile, are not affected by the rogue nature of the video flow from S1.

Thus S0's guaranteed service level agreement is met. This results in the audio application's improved performance inspite of having to compete with rogue flows.





Node	CIR (Mb)	PIR (Mb)	CBS (MB)	RATE (Mb)
S0	1	3		2.5
S1	1		10	2
S2	1		10	1

Figure 8. Results of experiment 2 (topology 1)

4.2. Network Topology II

The second network topology is shown in Figure 9. It builds on network topology I but removes source S2. The goal of this set of experiments is to evaluate the ability of CABB to fairly allocate resources to flows and to control unresponsive EF flows. The experiments consist of two flows: (1) *S0-Dest:* audio-EF PHB, (2) *S1-Dest:* video – EF PHB. The *Core-E2* link is the bottleneck link with a capacity of 7 Mb. The edge routers *E1*, *E2* each have one priority queue in this case.

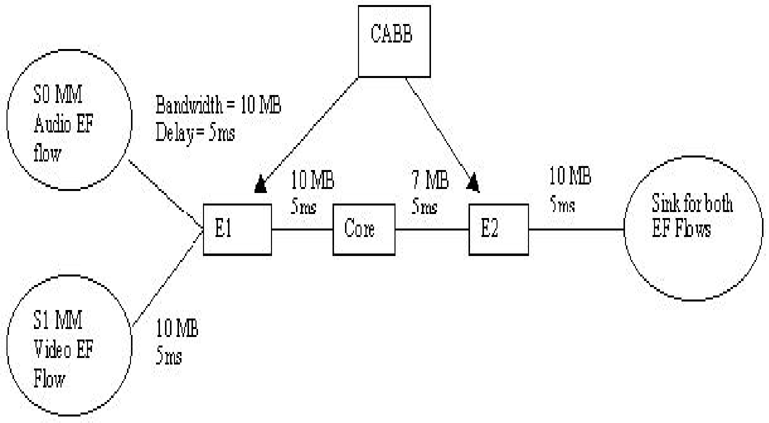


Figure 9: Topology 2, Multi EF flows

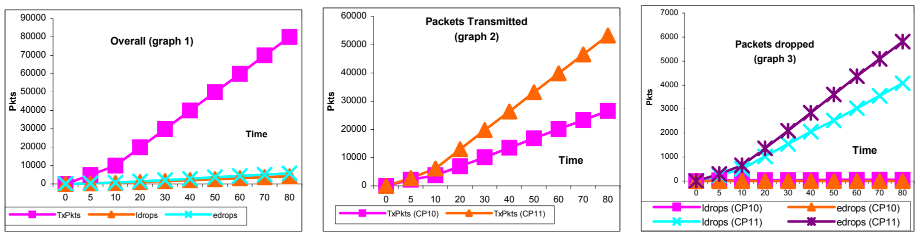
4.2.1. Experiment I

In this experiment, S0, S1 request 8mb and 2.6mb (PIR) and transmit at 8mb and 9mb respectively. The results of this experiment are plotted in Figure 10. Using its policy engine, the CABB makes decisions as follows:

Rate $_{S0}$ = 8 Mbps, UBW $_{S0}$ = function (1,3,8). - rejected.

Rate $_{S1}$ = 9 Mbps; UBW $_{S1}$ = function (5, 1, 2.6) = 2.6 Mbps. - accepted.

S0 is rejected as the request resource cannot be supported by the bottleneck link and the audio application is not flexible. This is illustrated in Figure 10. S1 is allocated the requested resources (PIR). However, Rate$_{S1}$ >> UBW$_{S1}$ (3.33 mb). As a result, one flow is rejected despite retrial and a later request was allocated full resources, thereby treating flows fairly while allocating among them. S1 exceeds its profile and is policed and forced to adhere to its SLA. Although there were sufficient resources (bandwidth) to meet S1's requirements, this source exceeded its allocation. As a result, CABB did not allocate extra resources to S1 but downgraded it to remain within profile. Without CABB, S1 would not have been downgraded. The resources saved by downgrading S1 can be used to support a new flow, which might otherwise have been denied resource. Note that our Active Resource Management (ARM) scheme presented in [25] extends this idea to dynamic management. This scheme helps in increasing the network's throughput while actively managing the flow's resource demands.



Node	CIR (Mb)	PIR (Mb)	RATE (Mb)
S0	3	8	8
S1	1	2.6	9

Figure 10. Results of experiment 1 (topology 2)

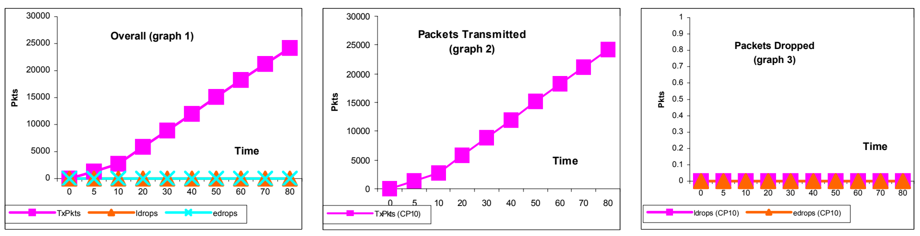
4.2.2. Experiment 2

In this experiment, S0, S1 request 1.5mb and 2mb (PIR) and transmit at 1.5mb and 1mb respectively. The results of this experiment are plotted in Figure 11. Using its policy engine, the CABB makes decisions as follows:

Rate $_{S0}$ = 1.5 Mbps, UBW $_{S0}$ = function (1.5) =1.5Mbps - accepted

Rate $_{S1}$ = 1 Mbps; UBW $_{S1}$ = function (5, 1, 2) = 1 Mbps - accepted.

AS can be seen from graphs in Figure 11, S0 is allocated full resources. S1 however is allocated reduced resources as video being more flexible than audio. We observed that CABB increased flow allocations with some flows getting reduced resources arrived at from their flexibility number.



Node	CIR (Mb)	PIR (Mb)	RATE (Mb)
S0	1	1.5	1.5
S1	1	2	1

Figure 11. Results of experiment 2 (topology 2)

4.3. Network Topology III

The third network topology is shown in Figure 12. It consists of: (1) *S0-Dest*: audio-EF PHB, (2) *S1-Dest*: video – EF PHB, (3) *S2-Dest*: FTP on TCP - AF PHB. The *Core-E2* link is the bottleneck link with a capacity of 7 Mb. The edge routers *E1*, *E2* each have one priority and two WFQ queues in this case. The goal of this experiment is to evaluate the ability of CABB to manage UDP and TCP flows.

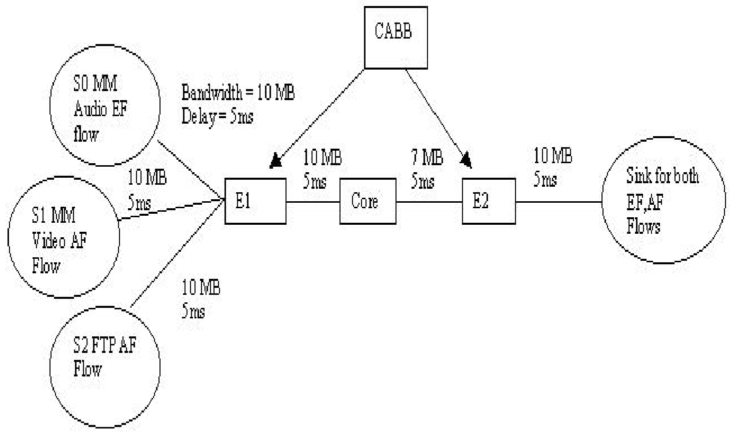


Figure 12. Topology 3: allocation and inter-effects of tcp and udp flows.

4.3.1. Experiment 1

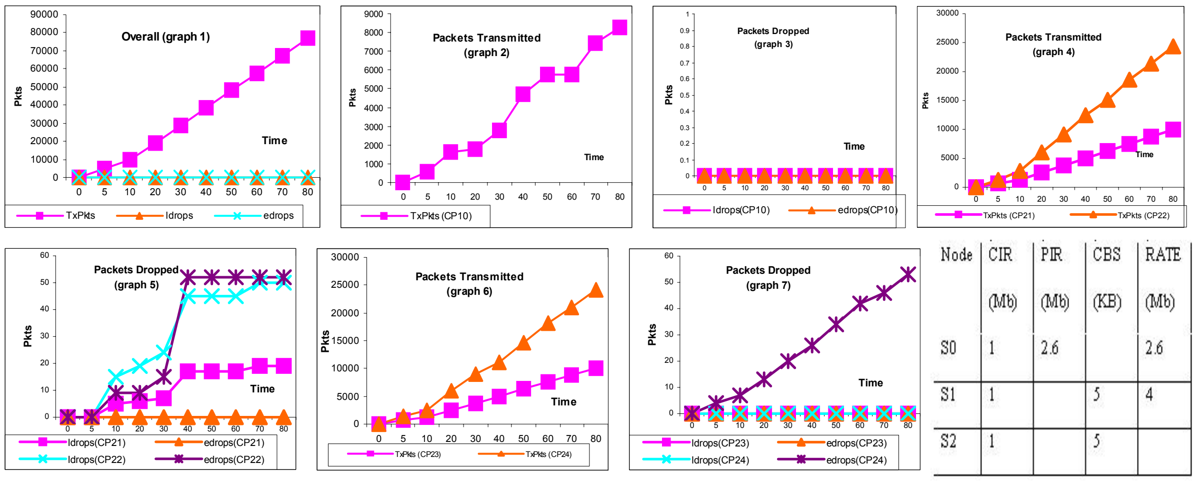
In this experiment we observe that all flows are allocated requested resources. Furthermore, the one flow aggregate does not affect another. Specifically, we see the effect of TCP going out-of-profile on UDP. S0, S1 and S2 request 2.6mb, 5mb and 5mb respectively. S0 and S1 both transmit at 1mb as noted in the table in Figure 13. Using its policy engine, the CABB makes decisions as follows:

Rate $_{S0}$ = 2.6 Mbps; UBW $_{S0}$ = function (2.6) = 2.6 Mbps. - accepted.

Rate $_{S1}$ = 4 Mbps; UBW $_{S1}$ = function (1) = 1 Mbps. - accepted.

Rate $_{S2}$ = 1 Mbps; UBW $_{S2}$ = function (1) = 1 Mbps. – accepted.

The results from this experiment are plotted in Figure 13. S0, S1 and S2 are allocated full resources. From graph 4,5 in Figure 13, we see that S1 is first downgraded and then dropped. Graph 6 and 7 show that the FTP application flow is downgraded. However as graphs 2 and 3 show, the EF flow is not affected by this flow. Out-of-profile S1, S2 are downgraded while in-profile S0 is unaffected. This experiment shows that CABB only admitted flows that do not load the network, giving reduced resources to some if necessary. The TCP flow is not starved but is penalized, as it does not conform to its SLA. Furthermore, it does not affect in-profile UDP (audio) flow.



Node	CIR (Mb)	PIR (Mb)	CBS (KB)	RATE (Mb)
S0	1	2.6		2.6
S1	1		5	4
S2	1		5	

Figure 13.Results of experiment 1 (topology 3)

5. Summary of Results

In the CABB-based diffserv system, multimedia flows are analyzed, allocated resources and regulated before being allowed to use the network. CABB effectively orders flows so that they conform to the traffic profile as agreed in the original SLA or to the reduced profile defined based on the application's flexibility number. Furthermore, queue management and diffserv policing ensure that when a flow goes out-of-profile its packets are downgraded and eventually dropped, thus regulating the flow. EF flows serviced by priority queues are allowed to go through with minimal drop. Also, a downgraded DSCP faces a harsher penalty as compared to an initially allocated DSCP. The out-of-profile AF flows do not affect the throughput of EF flows and vice versa. Multimedia applications, due to their flow requirements, are usually assigned to EF flows. However, flows that can tolerate losses may be assigned to AF flows as a result of adaptation by the broker. Hence, these flows go through a congested network, with the CABB doing coarse tuning of bandwidth requirements (avoiding overallocation) and the application level adaptive QoS doing the fine-tuning of the applications response/sensitivity to network changes [3]. As a result, the end user perceives a quantifiable QoS even with reduced resource allocation. CABB ensures higher flow throughput by identifying and controlling rogue flows. Furthermore, CABB eliminates the need for applications to have to deal directly with the diffserv router for resources [27].

A traditional broker does not account for an application's adaptability, leading to overallocation of resources [25]. Consequently, when compared to a normal broker, the CABB efficiently utilizes the network by allowing only those flows that do not congest it.

6. Conclusions

In this paper we presented a content aware bandwidth broker (CABB) that provides adaptive brokering for networked multimedia applications. CABB builds on the observation that multimedia applications are flexible or tolerant with respect to network parameters such as packet loss, delay and jitter [8]. It exploits this flexibility of multimedia flows to network level parameters to adapt the flows based on the state of network resources and maintain a quantifiable level of QoS despite unfavorable network conditions. Furthermore, in case of network congestion, CABB can adapt to the network state and reduce QoS rather than completely disrupting application flows. CABB also prevents non-conforming flows from affecting the performance of conforming flows by constantly monitoring and gradually degrading the level of service of non-conforming flows. Thus it provides incentive to support end-to-end congestion control for best effort traffic.

Experimental results presented show that multimedia flows are better managed and controlled, thereby improving perceived QoS and avoiding possible congestion. Furthermore, flow throughputs increase as CABB enables a larger number of flows to request for resources without over allocation. Results also show that CABB's policy decisions, based on content awareness, prevent congestion in the downstream network. These policy decisions are simple, unbiased, and effective. Furthermore, they admit the appropriate number of concurrent flows so that the network is not unduly loaded and hence controls traffic at the edge router, keeping the core simple. This design is easily scalable as no state is maintained in the routers. Multimedia applications or those that use UDP for data transmission are now coarsely controlled by the broker's policy decision. This work can be extended further. The next step is to integrate our work with Active Resource Management (ARM) scheme [25] and to enhance the CABB with a better understanding of the usage of multimedia flows and/or users to further optimize these flow allocations.

Acknowledgments

The research presented in this paper was supported in part by the National Science Foundation via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR)

References

1 T. Bonald, A. Proutière, J. W. Roberts, Statistical Performance Guarantees for Streaming Flows using Expedited Forwarding, Proc. of INFOCOM 2001, pp. 26-28, Tallinn, Estonia, April, 2001.

2 X. Xiao, L. M. Ni, Internet QoS: The Big Picture, IEEE Network Magazine, March/April, Vol 13, No 2, pp. 8-18, 1999.

3 N. Shaha, A. Desai, M. Parashar, Multimedia Content Adaptation for QoS Management over Heterogeneous Networks, proc of International Conference on Internet Computing 2001,Nevada, USA, pp 642-648, Computer Science Research Education and Applications (CSREA) Press, June 2001.

4 L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP), RFC-2205, The Internet Engineering Task Force (IETF), Sep 1997.

5 K. Nichols, V. Jacobson, L. Zhang, A Two-bit Differentiated Services Architecture for the Internet, RFC 2638, July 1999, see http://www.cs.wisc.edu/~cs640-1/papers/jacobson.qos.ps

6 E. Rosen, A. Viswanathan, R. Callon, Multiprotocol Label Switching Architecture (MPLS), RFC-3031, The Internet Engineering Task Force (IETF), Jan 2001.

7 QBone, Simple Inter-domain Bandwidth Broker Signaling, Internet 2, see http://qbone.internet2.edu/bb/

8 L. Peterson, B. S. Davie, Computer Networks - A Systems Approach, pp. 446-514, 2nd edition, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2000.

9 The Network Simulator - NS-2, see http://ww.isi.edu/nsnam/ns/

10 P. Chandra, A. L. Fisher, C. Kosak and P. Steenkiste, Network Support for Application-oriented QoS, Proc. of Sixth International Workshop on Quality of Service (IWQoS' 98), pp. 187-195, 1998.

11 D. Xu, K. Nahrstedt, A. Viswanathan, D. Wichadakul, QoS and Contention-Aware Multi-Resource Reservation, In Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC-9), pp. 318-327, Pittsburgh, PA, August 2000.

12 I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, A distributed resource management architecture that supports advance reservations and co-allocation, International Workshop on QoS (IwQoS'99), 1999.

13 J. Bollinger, T. Gross, A Framework-Based Approach to the Development of Network-Aware Applications, IEEE Transactions Software Engineering (Special Issue on Mobility and Network-Aware Computing), Vol. 24, No. 5, pp. 376-390, May 1998

14 F. Chang, V. Karamcheti, Automatic Configuration and Run-time Adaptation of Distributed Applications, ninth IEEE Intl. Symposium on High Performance Distributed Computing (HPDC), pp. 11-20, August 2000.

15 K. Nahrstedt, J. M. Smith, The QoS Broker, IEEE Multimedia, Vol 2, No 1, pp. 53-67, 1995.

16 Q. Zhang, W. Zhu, Y. Zhang, Resource Allocation for Multimedia Streaming over the Internet, Multimedia, Vol 3, No. 3, pp. 339, September 2001.

17 M. Choon Chan, Rolf Stadler and G. Pacifici, Managing Multimedia Network Services, Journal of Network and Systems Management (JNSM), Vol. 5, No. 3, 1997.

18 J. A. Gutierrez, D. P. Sheridan, and R. Radhakrishna Pillai, A Framework and Lightweight Protocol for Multimedia Network Management, Journal of Network and Systems Management (JNSM), Vol. 8, No. 1, 2000.

19 IETF Differentiated Services Working Group, see http://www.ietf.org/ids.by.wg/diffserv.html

20 S. Hares, A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment (version 7). 1999.

21 CA*net II Differentiated Services Bandwidth Broker System Specification, British Columbia Institute of Technology (BCIT), Technology Centre Group for Advanced Information Technology, 1998.

22 B. Li, D. Xu, K. Nahrstedt, J. W. S. Liu, End-to-End QoS support for Adaptive Applications Over the Internet, SPIE International Symposium on Voice, Video and Data Communications, pp 147 – 161, November 1998.

23 C. Aurrecoechea, A. Campbell, and L. Hauw, A Survey of QoS Architectures, ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6, No. 3, pp. 138-151, May 1998.

24 A. Campbell, and G. Coulson, A QoS Adaptive Transport System: Design, Implementation and Experience, Fourth ACM International Conference on Multimedia (ACM Multimedia 96), pp. 117-127, Boston, November 18-22, 1996.

25 M. Mahajan, A. Ramanathan, M. Parashar, Active Resource Management for The Differentiated Services Environment, International Journal of Network Management, John Wiley and Sons, February 2003 (to appear).

26 D. D. Clark, S. Shenker, and L. Zhang, Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms, Proc. of ACM Sigcomm 92, pp. 14-26, ACM Press, New York, Aug. 1992.

27 J. Shin, J. Kim, and C.C J. Kuo, Content-Based Packet Video Forwarding Mechanism in Differentiated Services Networks, Proceedings of the Packet Video Workshop 2000, Sardinia, Italy, May 2000.

Biographies

Manish Mahajan is Ph.D. student in the Department of Electrical and Computer Engineering at Rutgers University. He received his BS from VJTI, Bombay University. His research interests include computer networks, and parallel & distributed computing.

Manish Parashar is an Associate Professor in the Department of Electrical and Computer Engineering at Rutgers University. His research interests include autonomic computing, parallel & distributed computing, scientific computing, and software engineering.