Active Resource Management For The Differentiated Services Environment

Manish Mahajan, Ananthanarayanan Ramanathan, Manish Parashar

The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey
94 Brett Road, Piscataway, NJ 08854
{manishm, ananthr, parashar}@caip.rutgers.edu

***Abstract**: This paper presents a mechanism for active resource management (ARM) in a differentiated services environment. While the differentiated services architecture and the bandwidth broker agent provide a mechanism for QoS management through resource reservation, this mechanism is based on a static provisioning of resources. As bandwidth requirement are typically dynamic, such a static reservation approach can either lead to wasted bandwidth or leave applications resource-starved. The active resource management approach presented in this paper addresses this problem by dynamically reallocating resources based on current network state and applications requirements. An implementation and evaluation of ARM using the NS-2 simulation toolkit is also presented.*

***Keywords**: Active resource management, quality of service, differentiated services, bandwidth brokers.*

1 Introduction

In the current Internet architecture, a large percentage of the traffic is either multimedia related or a form of real time data that is critical to an application. Typical applications are Voice over IP (VoIP) and video conferencing. Such time-critical data require some level of Quality of Service (QoS) guarantee. QoS is the classification of packets for the purpose of treating certain classes or flows of packets in a particular way as compared to the other packets. The Internet Protocol (IP), however, is based on best effort and lacks the capability to provide such QoS guarantees [1]. Various solutions have been proposed to address this problem by guaranteeing applications their required resources. These include integrated services (e.g. RSVP), differentiated services and Multi Protocol Label Switching (MPLS).

The Differentiated Services (DiffServ) network architecture attempts to provide these QoS guarantees in the most scalable and least complex manner [2]. A DiffServ domain defines two levels of service provisioning: the standard best effort service, which is similar to IP, and the premium services where the clients' requests for service guarantees are met. In the DiffServ architecture, the Bandwidth Broker (BB) manages a domain's resources using service policies defined based on the client's requirements. The BB reserves the bandwidth requested by a client for a price. This reservation however, is made without any understanding of the nature of the information that will be transmitted. Although such a reservation provides a better sense of resource allocation than that provided by the DiffServ domain on it's own, the result is still a static provisioning of resources, and can lead to wasted bandwidth.

This paper presents the active resource management (ARM) approach that actively manages the resources in a DiffServ domain by dynamically reallocating resources based on the current requirements of applications and the state of the network. The ARM approach is motivated by the observation that the actual traffic generated by a client rarely approaches the peak rate bandwidth that has been reserved for the client. Consequently, when the client's traffic drops below the reserved rate, a portion of the unused bandwidth can be returned to a pool of available bandwidth. ARM is implemented and evaluated using the network simulator (NS-2) [3] toolkit. Our experiments demonstrate that by actively over provisioning and dynamically reallocating available resources, ARM can effectively increase the overall utilization of the available bandwidth and support increased numbers of clients while honoring QoS guarantees.

The rest of this paper is organized as follows. Section 2 presents background material and outlines related work. Section 3 describes the ARM approach. Section 4 outlines the implementation and evaluation using the NS-2 network simulator, and presents experimental results. Section 5 presents our conclusions.

2 Background and Related Work

2.1 Background

Existing models for providing Quality of Service are based on service differentiation. These models can be classified as reservation protocols, label switching and relative priority marking [4, 5]. The reservation protocol model, such as RSVP, relies upon traditional datagram forwarding in the default case, and uses an exchange of signaling messages to establish packet classification and forwarding state on each node along the data transfer path. This requires the maintenance of state at each hop along the path for the duration of the transfer, reducing its scalability. The label-switching model includes MPLS and ATM. In this model, path forwarding state and traffic management is established for traffic streams on each hop along the network path permitting finer granularity resource allocation to traffic streams. This improved granularity comes at the cost of additional management and configuration required to establish and maintain the label switched paths. In addition, the amount of forwarding state maintained at each node scales in proportion to the number of edge nodes of the network in the best case (assuming multipoint-to-point label switched paths), and it scales in proportion with the square of the number of edge nodes in the worst case, assuming edge-edge label switched paths with provisioned resources are employed. We have accepted DiffServ as the least complex and scalable solution for providing QoS.

2.1.1 Differentiated Services

Differentiated Services is a set of technologies that are used to provide quality of service (QoS) in a world of best effort service provisions [2]. It provides a framework and building block to enable deployment of scalable service discrimination in the Internet. To achieve scalability, the individual host-to-host microflows are aggregated into a single larger aggregate flow and the aggregate flow receives special treatment. The DiffServ architecture as shown in Figure 1 [4] is

based on a simple model where the traffic entering a network is classified and possibly conditioned at the boundaries of the network, and then assigned to different service classes, thus pushing all the complexities to the edge routers, leaving the core routers as simple as possible.

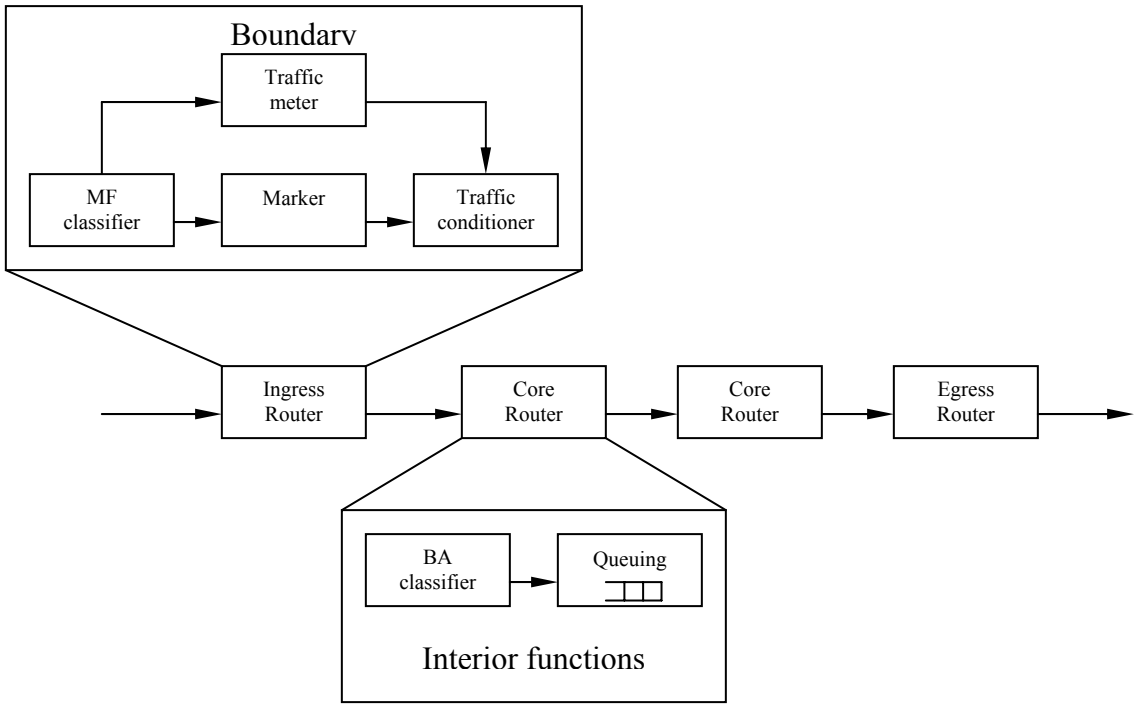

Figure 1: The DiffServ Architecture

The approach taken by DiffServ is to classify individual micro flows at the edge routers in the network, into one of the many classes and then apply a per-class service in the core of the network. This classification is performed at the network's ingress router, based on the service requested, and marked with a DiffServ Code Point (DSCP) in the ToS (Type of Service) field of Ipv4. The core routers that forward the packet examine the DSCP to determine how the packet should be treated. All packets marked with the same DSCP form a behavior aggregate (BA); and a Per Hop Behavior (PHB) is applied to each BA inside the network. The PHB defines the service the packet receives at each hop as it is forwarded through the network. The boundary router or the edge router is positioned at the edge of the DiffServ capable network. This router is responsible for packet classification, packet marking, metering and traffic conditioning. Interior nodes are

core switches or routers that provide the PHB based on the DSCP bits. The core routers employ queue management techniques and scheduling mechanisms such as random early detection (RED) and weighted fair queuing (WFQ) to provide the PHB. There are two defined PHB's: expedited forwarding (EF), and assured forwarding (AF). EF PHB [6] is defined to support low loss, low delay, and low jitter. The AF PHB [7] defines four relative classes of service with each service supporting three levels of drop precedence.

DiffServ is being regarded as a reasonable solution to provide Quality of Service on the Internet. Research and testing of the DiffServ architecture is being conducted by TF-TANT [8] for a European environment, CSIRO and AARNET (Australian Academic and Research Network) [9] for an Australian environment and by other universities such as the University of Kansas [10] and Massachusetts Institute of Technology [11]. Vendors such as Cisco, IBM, Nortel Networks, Lucent, Cabletron and Ericsson [9] provide DiffServ functionality in their routers, and Nortel Networks has evaluated DiffServ using NS-2 toolkit [12].

2.1.2 Bandwidth Broker

A bandwidth broker [13] manages network resources for IP QoS services that are supported within a network and used by customers of the network services. A BB may be considered a type of policy manager in that it performs a subset of policy management functionality. A policy manager is one who manages the access of users to network services. A service contract between a customer and a service provider that specifies the forwarding service a customer should receive is called a Service Level Agreement (SLA). The Service Level Specification (SLS) is a translation of a SLA into appropriate information necessary for provisioning and allocating QoS resources within the network devices, in particular, at the edges of the domain on links connecting the domain to adjacent domains. The bandwidth broker requires both the SLA and the SLS to achieve a range of services accorded to the user. Based on the SLA the broker decides whether it can provide the requested allocation, and it configures the edge router accordingly to

mark and classify the packets as decided in the SLS [4]. The BB is also responsible for managing inter-domain communication with BBs in neighboring networks to coordinate SLSs across the domain boundaries.

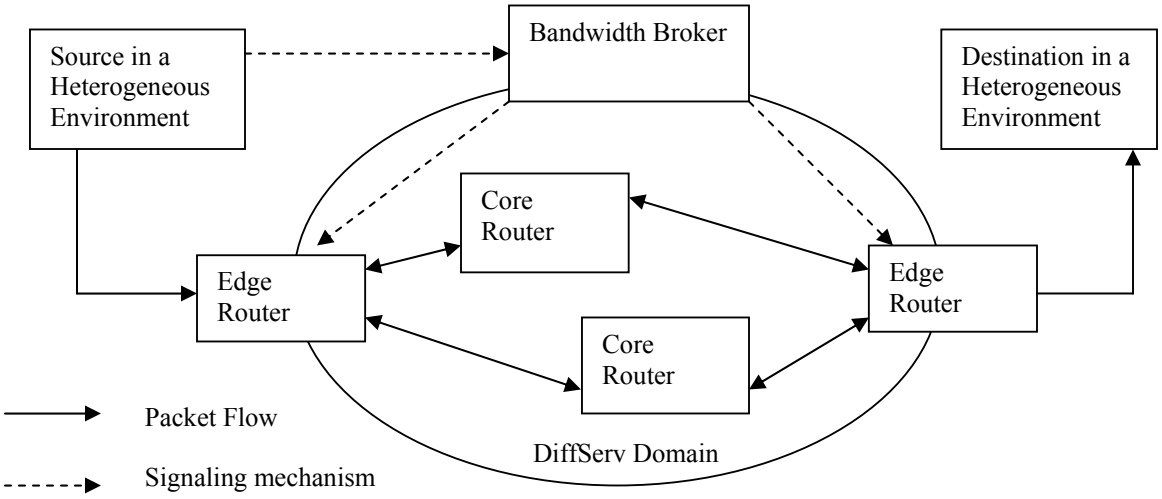

Figure 2: A Generic Model of a Bandwidth Broker in a DS Domain

As seen in Figure 2, the BB gathers and monitors the state of QoS resources within its domain and on the edges of the domain to and from adjacent domains [4]. That information, together with the policy (from the policy rules database) is used for admission control decisions on QoS service requests to the network. BB makes use of the network state information to verify that resources are currently available in the network to support a request. Across boundaries, the SLS may be on an aggregate basis, where aggregation is on all flows within the domain of a particular QoS service type (i.e. DiffServ code point). Within a domain, individual flows may be allocated resources based on the SLS by issuing Resource Allocation Requests (RARs). It is the responsibility of the BB to coordinate allocation and provisioning of the aggregate resources of the SLSs, into and out of its domain, with the resources requested by the RARs.

The bandwidth broker consists of a few basic components as shown in Figure 3. Their functions are as defined [14]:

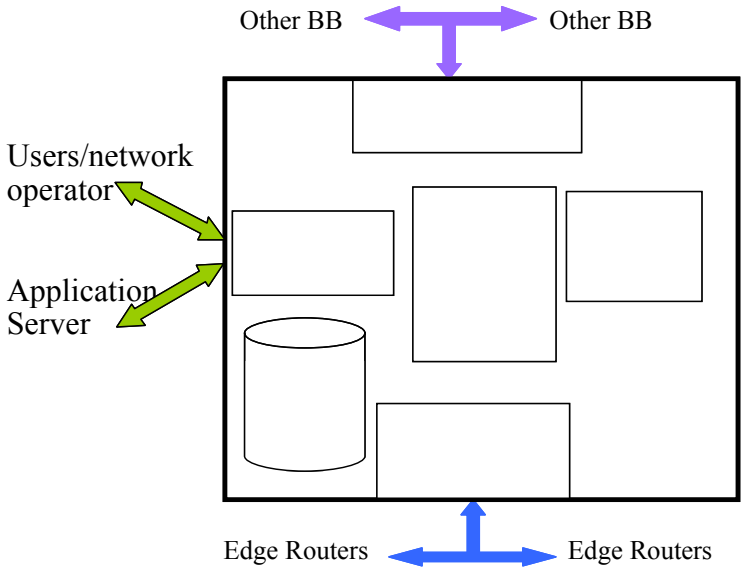

Figure 3: Functional Decomposition of Bandwidth Broker [14]

- User Interface: The user/application interface provides a means for the user to make resource requests directly, or to the network operator who will enter the requests. The interface also receives messages from setup protocols.
- Inter-domain Interactions: A method for enabling peer BBs' to make requests for resources and admission control decisions to enable flow of traffic.
- Intra-domain Interactions: Providing a method for the BB to configure the edge routers within the domain so as to provide quality of service.
- Routing Table: A routing table is maintained to access inter-domain routing information so that BB can determine the edge routers and the downstream routers before allocating their resources. Also additional routing paths can be maintained for different flows within the domain.
- Database: A database is used to store information about all the BB's parameters. The different information stored within the repository are SLAs, current reservations, configuration of routers, DSCP mapping, and policy information.

The Bandwidth Broker has been designed to add intelligence to the DiffServ, to help improve resource allocation. The Internet2 Qbone Bandwidth Broker Advisory Council (BBAC) has defined a set of standards for BB, and work is being carried on at the Lawrence Berkeley National Labs (LBNL) [15], University of Kansas [10], Siemens, and various universities. There are two commercial products with BB capabilities made available by Orchestream [16] and Extremeware [17].

2.2 Related Work

Quality of Service can be achieved through efficient resource allocation by network architecture or by software programs that provide good resource management. Of the many network architectures suggested, such as IntServ (RSVP), MPLS and others, DiffServ provides the most scalable and least complex solution. Resource management is a relatively new field that is a popular research area as it is easier and necessary to manage existing resources for heterogeneous networks, than it is to design and implement a new network architecture that provides QoS.

2.2.1 Resource Management

Key works in resource management include PBNM, and GARA. Intel's Policy Based Network Management (PBNM) [18] technology provides the ability to define and distribute policies to manage heterogeneous networks. These policies control critical network resources such as bandwidth, security, and Web access. PBNM is scalable and offers multi-vendor support. The Globus group defines the Globus Architecture for Reservation and Allocation (GARA) [19, 20] that supports flow specific QoS specifications, immediate and advance reservations, and online monitoring and control of both individual resources and heterogeneous resource ensembles. The prototype GARA implementation builds on the differentiated services mechanisms to enable the coordinated management of two distinct flow types - foreground media flows and background

bulk transfers, as well as the co-reservation of networks, CPU's and storage systems. Other works in resource management include resource allocation schemes for connections' tolerating statistical QoS guarantees in public wide area ATM networks using effective bandwidth [21].

2.2.2 Resource Management for the DiffServ

Many schemes have been proposed to help better manage the resources in the DiffServ architecture. The scheme proposed by Reichmeyer and Zhang [13], defines a two-tier resource management model and introduces the concept of Bandwidth Broker as the resource manager for each domain and a BB-to-BB protocol equivalent to BGP in routing, for inter-domain resource management. The BB is one of the more popular and accepted resource management concepts for the DiffServ. The second scheme proposed by Jun Ogawa and Yuji Nomura [22] is an extension to the first one, wherein it is shown that the BB is not sufficient to manage the resources due to the existence of multi-vendor routers running various routing protocols within the same administrative domain. Since DiffServ specifies only externally observable behaviors in the Forwarding Path, equipment vendors can use different mechanisms to implement these behaviors and if BBs were to directly control each router they would have to be aware of the details of each router's implementation. This level of detail complicates the design of a BB, especially in large heterogeneous domains where a number of different router designs coexist. Thus the second approach proposes the design of an edge router be equipped with *Virtual Configuration Manager* (VCM), where a higher-level description named *Virtual Configuration Description* (VCD) generated in BB is translated into the specific parameters for the Forwarding Path. As VCM succeeds to veil the details of the Forwarding Path implementation, BB can concentrate on the management of SLA without being aware of different implementations of routers. This is made possible by designing the VCM in Java and making it OS independent.

2.2.3 Active Resource Management for the DiffServ

Resource management on the Internet is essential and the introduction of DiffServ and Bandwidth broker to further satisfy these needs is appropriate. However these methods provide a static one-time management at the start of a flow, resulting in resource wastage. For current network traffic, which is dynamic in nature, we need to actively manage existing resources so as to increase utilization, to reduce network congestion and provide QoS. One of the first approaches towards this goal was the Active Queue Memory Management concept [23] that actively manages a DiffServ's queues in order to provide better and more proactive response to network congestion to meet QoS goals. The two mechanisms that support active queue management in large IP networks are: Random Early Detection (RED) (currently being deployed in large IP networks along with various extensions to RED such as Weighted RED (WRED)), and Explicit Congestion Notification (ECN) (initially an experimental addition to the IP architecture). Another proposed solution is the Resource Management in DiffServ (RMD) [24] framework for radio access networks (RAN). The current strategies for resource management do not meet the requirements for resource management within a RAN. It is a lightweight solution for the edge-to-edge dynamic resource management problem of DiffServ domains. The RMD framework follows a distributed admission control and resource management approach, which is different from the bandwidth broker designs for DiffServ. The RMD framework for the DiffServ is an open framework interoperable with other resource management mechanisms with wide scope of applicability in different DiffServ networks. It is a simple framework with good scaling properties and has low cost of implementation. Since it is designed for extension of DiffServ concepts, the RMD relies on DiffServ principles for QoS provisioning and preserves its scalability properties. The RMD framework enhances these concepts with new ones necessary to provide dynamic resource provisioning and admission control in DiffServ domains. In the RMD framework the problem of a complex reservation within a domain is separated from a simple reservation within a node.

Accordingly there are two types of protocols defined within the RMD: Per Hop Reservation (PHR) and Per Domain Reservation (PDR). The PHR is a newly defined protocol, while the PDR could be one of the existing protocols such as RSVP [5], RSVP Aggregation, Simple Network Management Protocol (SNMP) [25], and Common Open Policy Service (COPS) [26].

This paper presents Active Resource Management (ARM) for the DiffServ environment. DiffServ provides an approach to IP QoS management that is modular, incrementally deployable, and scalable while introducing minimal complexity [4]. ARM uses the DiffServ bandwidth broker as the resource manager and proposes an improvement in the reservation mechanism by giving it run-time capabilities. Resource utilization is improved by keeping a track of the network characteristics per client flow and reusing the unused, already allocated, bandwidth without loss of service. This prevents wastage of resources, and increases the number of clients who can receive service.

2.2.4 A Comparison of Active Resource Management Mechanisms

Table 1, gives an overview of existing approaches to active resource management for various networks.

Table 1: A Comparison of Different Active Resource Management Schemes

	Active Queue Memory Management	RMD	ARM
Target Networks	Heterogeneous Networks	Radio Access Networks	Heterogeneous Networks
Resource Manager	None	None	Bandwidth Broker
Changes made to DiffServ	All the Router Queues	DiffServ's Per Hop Behavior Functionality	Bandwidth Broker
Contributions	Random Early Detection (RED) & Explicit Congestion Notification (ECN)	Per Hop Reservation (PHR) & Per Domain Reservation (PDR)	Active Resource Management (ARM) Algorithm

Results	Manages congestion and end-to-end delay and supports delivery of DiffServ classes.	Improves resource utilization and provides better resource management capabilities to RANs	Improves resource utilization, reduces resource wastage and increases the number of clients.

3 Active Resource Management (ARM)

3.1 Architectural Framework

As described above, in the DiffServ model, predefined policies or SLAs are used to allocate the resources to a particular client. These policies are based on certain parameters such as the clients' peak traffic rate, the time for which the service is required, and the acceptable delay and jitter for an application. For many applications, for e.g. where the transmitted information is in the form of streaming media, the traffic rate is bursty in nature and is rarely at peak transfer rate, which is the amount of bandwidth allotted to the client. In such a situation, a portion of the allocated bandwidth remains unused. However, as this bandwidth is provisioned to the particular client, no other client can make use of it. Furthermore, bandwidth is reserved for a particular client and this reservation is applicable to all applications belonging to the client. As a result, a client will make reservations based on its maximum requirements, and every application belonging to the client, whether it is a streaming media application or just a simple mail application, will get this allocated service. This might result in delay or jitter being introduced into a client's dataflow.

In order to allocate these resources in a more intelligent fashion, a broker agent is used. The agent maintains a database of parameters pertaining to the various flows. Parameters such as service level agreements, current reservations/allocations, edge router configurations, service mappings/DSCP mappings, policy information, and management information. In accordance with these parameters the broker agent makes a reservation for the client and assigns a DSCP for that service. This is certainly a better form of allocation, as sets of parameters are taken into consideration for each reservation. Each client gets to define his requirements and these get

translated in to SLA's, which affects the resource allocation. But the end result is the same static reservation, where bandwidth once allotted to a client is used solely according to that client's traffic flow.

Thus there is a need to reuse the bandwidth wasted on each reservation that is made and if possible re-allot it to another client. The basic concept behind ARM is that by effectively knowing when a client is sending packets and how much of this allotted bandwidth is being used at any given time, the excess bandwidth can be reallocated without loss of promised service. Each client's request is equated to an SLS, which specifies the amount of bandwidth, the duration of the connection and a few other parameters. These parameters map to a particular DSCP that is used to mark incoming packets from that client so as to inform the core routers to forward the packets with appropriate priority. The core router functionality is thus kept simple. In order to measure the traffic rate of every client, the bandwidth broker agent uses a meter that is provided by the DiffServ [27, 28]. For example, the TSW (Time Sliding Window) Tagger [29] is a meter that measures the average traffic rate, using a specified window size for the TSW2CM (Time Sliding Window 2 Color Marker) and TSW3CM (Time Sliding Window 3 Color Marker) [30] policers. With the knowledge of incoming traffic, different DSCP's are defined for various traffic rates. So when the broker agent notices a traffic rate that is less than the rate agreed upon, it steps down to a lower DSCP that suits the current rate. The remaining unused bandwidth is now sent to a pool of available bandwidth and is used when required by new clients or then given back to clients when they require it. This helps in increasing the number of clients who get the same kind of reservation guarantees, translating to more revenue for the same amount of bandwidth. It is a basic case of over allocation of resources. For the worst case scenario where all the clients send in traffic at their peak rate, the additional bandwidth is provided by dipping into the pool of bandwidth belonging to the best effort services. To achieve this, a set of preconditions for allocations is necessary. The preconditions are: 1. There should be a limit on the number of reservations allowed per class, and, 2. A fixed amount of bandwidth must be reserved for best

effort services. By limiting the number of reservations for the Expedited Forwarding (EF) PHB, we can limit the number of premium service reservations, which have strict bandwidth requirements and correspondingly reduce the amount of unused bandwidth. Also by reserving bandwidth for best effort and providing a threshold of tolerance within which we can add or remove bandwidth as required, we efficiently reshuffle the unused bandwidth without adverse effect on the service agreements.

3.2 An Illustrative Example

The functioning of the ARM algorithm is explained with the help of a test network. As shown in Figure 4, the test network includes two DiffServ domains, and a client belonging to a heterogeneous network architecture. 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. The individual client from the heterogeneous network provides a method of verifying the allocation rules for out of domain traffic sources.

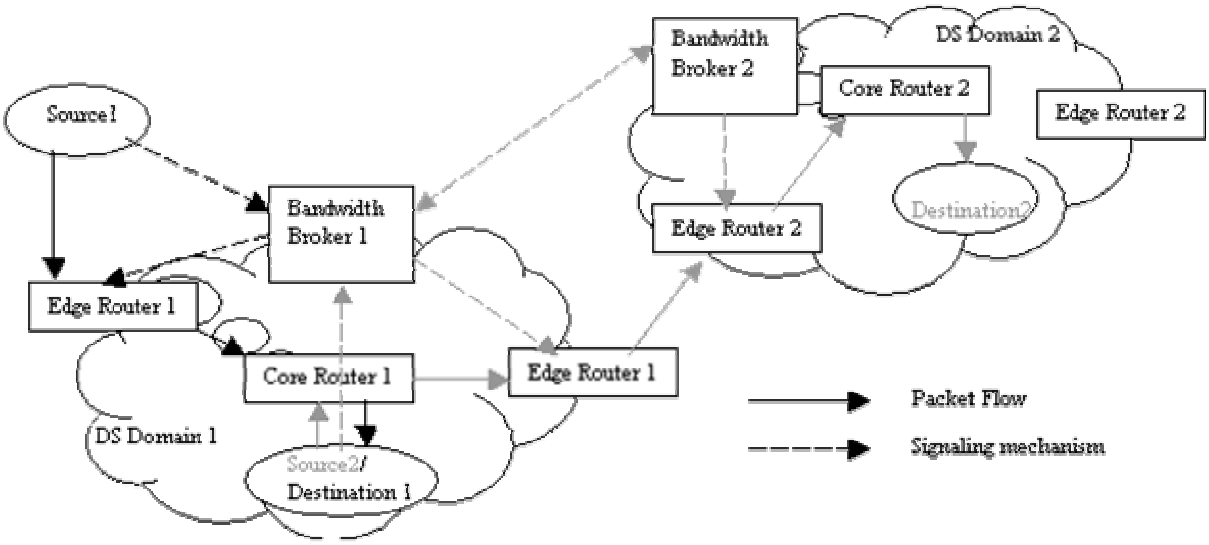

Figure 4: Test Network Showing Different Scenarios

When Source1 that is outside the DiffServ domain requests service, it contacts the BB1 on DS1 enroute to Destination1. If Destination1 is within this domain, as shown in Figure 4 using

the black markings, BB1 looks into its database, decides upon the best available bandwidth, jitter and delay parameters, defines the SLA, and assigns a DSCP for the traffic flow between this source-destination pair. It then configures the edge router to mark the packets from this client with the correct DSCP so that the core routers just forward the packet according to the priority accorded to the flow.

The BB1 also assigns a set of lower DSCPs, which define slightly lesser bandwidth requirements. During a particular packet exchange from Source1, if the BB1 notices that it is running short of bandwidth to allocate, it uses the meter to check the traffic rate from Source1, and if the rate is any lesser than the bandwidth allotted then it steps down the service to a lower DSCP which provides only the required amount of bandwidth. The remaining unused bandwidth is returned to a pool of unused bandwidth, which can be allotted to another client or returned to Source1 when needed. If the source (Source2) is in DS1 domain and the destination (Destination2) is DS2 domain, as shown by the gray markings, Source2 contacts the BB1 of its domain. BB1 then looks at the database and the routing table to figure out the downstream edge router and the peering BB2, and sends a RAR based on the SLA it has with Source2. On confirming the reservation it allots a set of DSCPs corresponding to different levels of reservation, and configures the edge router of its domain to mark the packets from Source2 accordingly. The peering BB2 also configures its routers to perform the same kind of marking for that source destination pair. Both BBs define their own set of DSCPs for this flow. When there is need for conserving bandwidth in either domain, the corresponding BB decides to step down on the DSCP marking for the flow through that particular domain, thus saving bandwidth for reuse.

4 Implementation and Evaluation of the ARM Algorithm using NS-2

4.1 Implementation

We have implemented the ARM algorithm on the NS-2 toolkit [3]. The NS-2 (Network Simulator-2) toolkit has substantial functionality for simulating different network topologies and traffic models. NS also has an open architecture that allows users to add new functionalities which proves very useful for us. Along with the DiffServ patch provided by Nortel Networks, we can generate DS domains and create suitable test networks [12].

The DiffServ implementation has three modules to it. Two of them are with regards 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 the packets will receive at an edge router. Policies are set using Tcl [31] scripts. The policy class uses a policy table to store the parameter values. The table is in the form of an array of structures that has various fields such as SLA, current reservation, router configuration, policies, and DSCP mappings. The packet that arrives at the edge router is checked to decide as to which traffic aggregate it belongs to, 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, else it gets downgraded to a lower DSCP.

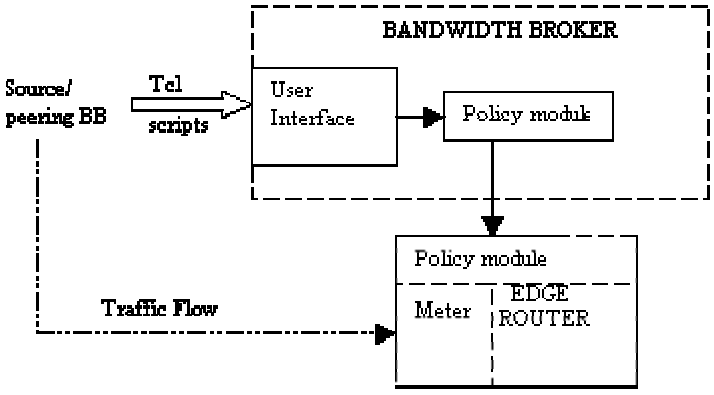

Figure 5: Modular Breakup of the BB and its Interactions in a Traffic Flow

The bandwidth broker is used to configure the policy module of the DiffServ. We define two modules for the broker agent as shown in Figure 5. The modules are; user interface module through which the user/network operator can allocate resources, and a DiffServ manager, which does all the resource allocations. These allocations are reflected in the Policy module of the DiffServ, which is used to configure the edge routers.

The agent makes the provisioning based on the SLA's as agreed upon with the client/user (through the user interface module) using Tcl scripts and in correspondence with other parameters in the database module such as the current reservations and the router configurations. The configuration changes are made to the policy module, and these changes are reflected in the policy module of the DS edge router. This achieves static provisioning. The Active Resource Management (ARM) algorithm keeps a track of every client's average traffic rate using a meter such as the TSW Tagger that is a part of the edge router. We use two meters, one that measures the traffic rate using a window size and one that measures the current flow rate. The algorithm uses both the values to decide how much bandwidth can be reclaimed and re-allotted. Within the policy module we associate every source-destination flow with a policy type, meter type, current rate of traffic (the rate agreed upon with the client) and other policer specific parameters. We associate a set of DSCPs with this flow. Each DSCP corresponds to a different traffic rate and the algorithm switches the current DSCP marking of the packet flow according to the traffic rate indicated by the meter.

4.2 Experimental Evaluation

We have evaluated the ARM algorithm with three sets of experiments. Each experiment consists of three evaluations –first performed on a DS domain that does the resource provisioning using its own capabilities (DS), then on a DS environment that uses a Bandwidth Broker to help provision the resources intelligently (DS+BB), and finally on the DS environment that uses bandwidth brokers implementing the ARM algorithm (DS+BB+ARM). The first experiment

allocates the entire available bandwidth, the second experiment pushes the allocation over the limit, and finally the third experiment tests the system for a longer simulation time.

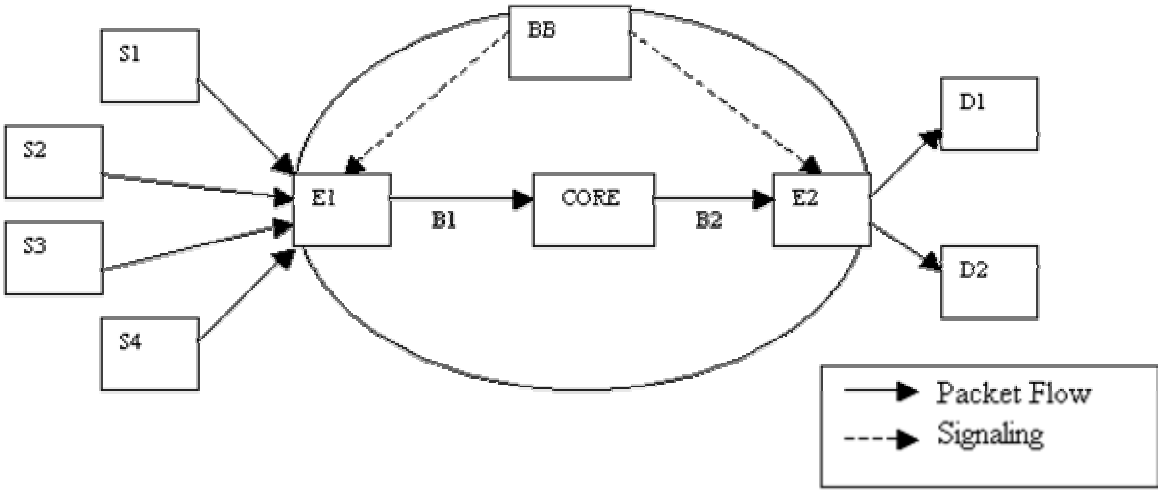

Figure 6: Test Network

Our test network is shown in Figure 6 and consists of 6 nodes and 3 DS enabled routers. S1, S2, S3, S4, D1 and D2 are the nodes, E1 and E2 are the edge routers, and Core is a core router. The DS Domain consists of the three routers and one bandwidth broker agent that configures the edge routers. Table 2 lists the various parameters used in the test network and their values. Table 3 lists the various source parameters.

Table 2: Legend for the Test Network

Name	Type	Parameters
S1	Source 1	Bursty Source with Pareto distribution. Pkt size 210, burst time 500ms, idle time 300ms, rate 500 Kbps.
S2	Source 2	Bursty Source with Exponential distribution. Pkt size 210, burst time 500ms, idle time 300ms, rate 1Mbps.
S3	Source 3	CBR type Source on UDP. Rate 1.5 Mbps.
S4	Source 4	CBR type Source on UDP. Rate 2 Mbps.
D1	Destination 1	Generic node.
D2	Destination 2	Generic node.
BB	Bandwidth Broker agent	Configures the Edge routers.
B1	Link between E1 and Core	5Mbps, 5ms delay, 4 queues with 3 drop precedence, Priority scheduling mode.
B2	Link between Core and E2	5Mbps, 5ms delay, 4 queues with 3 drop precedence, Priority scheduling mode.

Table 3: Source Parameters

CIR	Committed Information Rate
ALTCIR	Alternate Committed Information Rate
PIR	Peak Information Rate
ALTPIR	Alternate Peak Information Rate
POL	Policer for first set of parameters
ALTPOL	Policer for alternate set of parameters

Table 4 shows the initial request for resources made by the client along with the alternate policy requests in case the initial request cannot be fulfilled.

Table 4: Initial and Alternate Policy Request

Source	CIR	PIR	POL	ALTCIR	ALTPIR	ALTPOL
S1	500 Kbps	500 Kbps	EF	400Kbps	450 Kbps	EF
S2	1Mbps	2Mbps	EF	750Kbps	1.5Mbps	EF
S3	1.5Mbps	3Mbps	TSW2CM	1Mbps	2Mbps	TSW3CM
S4	2 Mbps	3Mbps	TSW3CM	1Mbps	2Mbps	TSW3CM

4.2.1 Experiment 1: Exact Allocation of Resources.

In this experiment we test the DiffServ's capability to provide service when all the bandwidth is allocated. Both the BB experiment and the ARM algorithm experiment allocate less than maximum bandwidth available and result in a better utilization of bandwidth. Table 5 presents the policy tables for the three evaluations: DS, DS+BB, and DS+BB+ARM. The bandwidth used in each case is calculated by adding up the CIR. Evaluation 3, using the ARM algorithm, shows two sets of policy tables. The first table corresponds to the initial allocation made, and the second table shows the resultant run time allocations made by the ARM algorithm.

Table 5: Policy Table

Flow		S1 to D1	S2 to D1	S3 to D2	S4 to D2
Policer Type		EF	EF	TSW2CM	TSW3CM
Initial Codepoint		10	11	15	17
1.A: DS simulation	CIR	500Kbps	1Mbps	1.5Mbps	2Mbps
	PIR	500Kbps	2Mbps	----	3Mbps
1.B: Broker simulation	CIR	500Kbps	1Mbps	1.5Mbps	1.5Mbps
	PIR	500Kbps	1Mbps	----	3Mbps
1.C: ARM initial allocation simulation	CIR	500Kbps	1Mbps	1.5Mbps	1.5Mbps
	PIR	500Kbps	1Mbps	----	3Mbps
1.C: ARM runtime allocation simulation	CIR	375.0Kbps	750Kbps	1.5Mbps	1.496Mbps
	PIR	500Kbps	1Mbps	----	2Mbps

As can be seen from Table 5, the ARM algorithm improves bandwidth utilization and conserves more than 20% of the bandwidth. The bandwidth used is plotted in Figure 7.

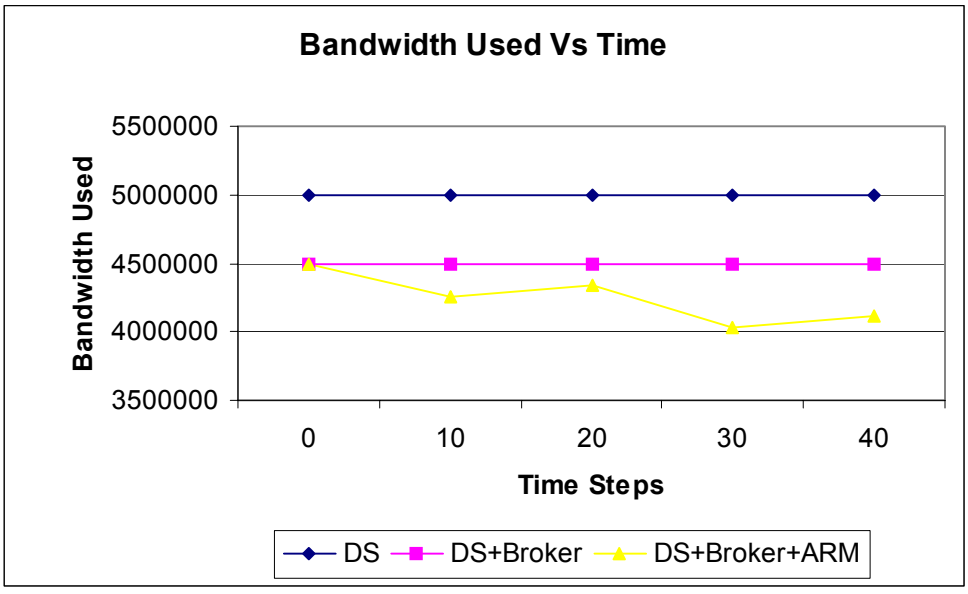

Figure 7: Bandwidth Used vs. Time

From the packet statistics shown in Table 6, it is evident that bandwidth utilization improves by using ARM. Furthermore, we also have a better mechanism to manage traffic flow and reduce the number of dropped packets.

Table 6: Packet Statistics

1.A.DS Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	83051	79482	3518	51
10	4472	4472	0	0
11	12303	12303	0	0
15	25056	23799	1257	0
16	8026	7566	409	51
17	33083	31231	1852	0
18	111	111	0	0

CP	Codepoints
TotPkts	Total Packets
TxPkts	Transmitted Packets
ldrops	Late drops
edrops	Early drops

1.B. Broker Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	71989	71989	0	0
10	5481	5481	0	0
11	10838	10838	0	0
15	15142	15142	0	0
16	4767	4767	0	0
17	35659	35659	0	0
18	102	102	0	0

1.C. ARM Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	71989	71989	0	0
10	5481	5481	0	0
11	10838	10838	0	0
15	15142	15142	0	0
16	4767	4767	0	0
17	35631	35631	0	0
18	130	130	0	0

The legend for Table 6: cp – Codepoint; TotPkts – Total packets; TxPkts – Transmitted packets; ldrops – Late drops; and edrops – Early drops.

A graphical representation of the packet statistics is shown in Figure 8. It is clearly visible that while using the ARM algorithm, the service guarantees are maintained, and no packets are dropped. For example, in DS a total of 3569 packets (3518 ldrops + 51 edrops) are dropped while there are no packet drops for DS+Broker+ARM.

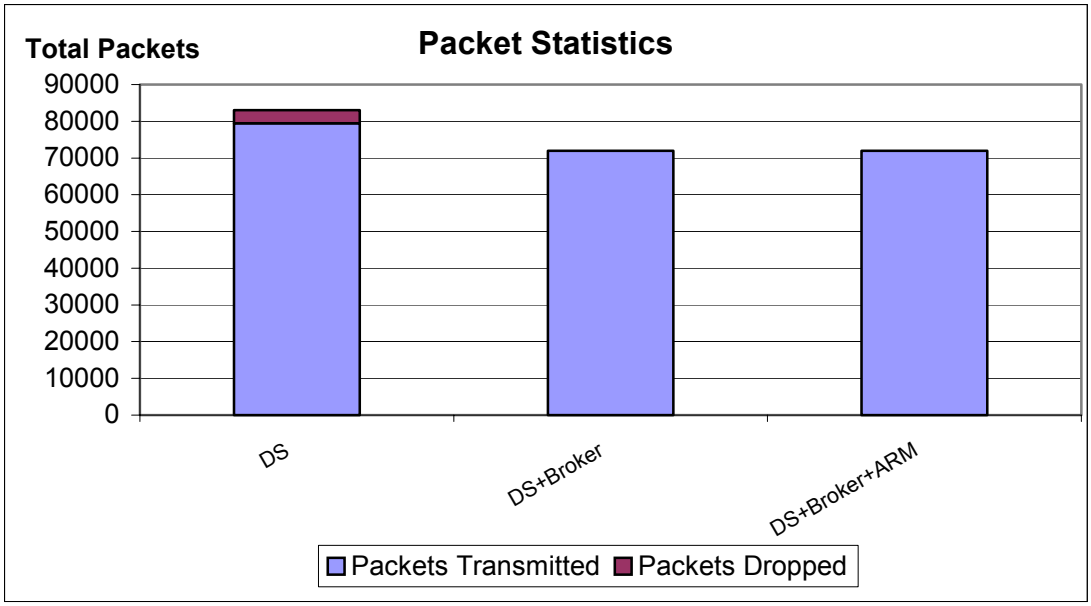

Figure 8: Packet Statistics

4.2.2 Experiment 2: Over allocation of Resources.

In the second experiment, we stress the allocation limits by over allocating. This is achieved by increasing the bandwidth requirements of Source 2 to 1.5Mbps. The Broker manages to keep the allocation under control and the ARM algorithm subsequently improves on the broker's allocation. Experiments 2.A, 2.B and 2.C in Table 7 show the DS results, broker results and the ARM algorithm results respectively.

Table 7: Policy Table

Flow		S1 to D1	S2 to D1	S3 to D2	S4 to D2
Policer Type		EF	EF	TSW2CM	TSW3CM
Initial Codepoint		10	11	15	17
2.A: DS simulation	CIR	500Kbps	1.5Mbps	1.5Mbps	2Mbps
	PIR	500Kbps	2Mbps	----	3Mbps
2.B: Broker simulation	CIR	500Kbps	1Mbps	1.5Mbps	1.5Mbps
	PIR	500Kbps	1Mbps	----	3Mbps

2.C: ARM initial allocation simulation	CIR	500Kbps	1Mbps	1.5Mbps	1.5Mbps
	PIR	500Kbps	1Mbps	----	3Mbps
2.C: ARM runtime allocation simulation	CIR	500Kbps	750Kbps	1.497Mbps	1.498Mbps
	PIR	500Kbps	1Mbps	----	3Mbps

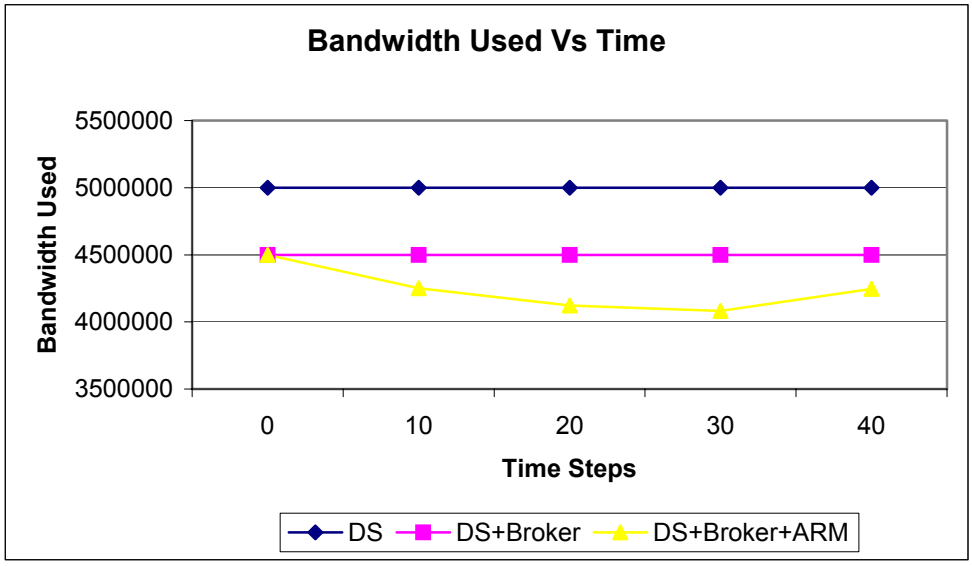

Figure 9: Bandwidth Used vs. Time

The bandwidth used over the 40 time steps duration of this experiment is plotted in Figure 9. It can be seen that some used bandwidth is reclaimed for further use, thus increasing the number of clients that can be served with guaranteed service. The packet statistics for over allocation are as shown in Table 8. Even during over allocation, the ARM algorithm manages to maintain the service levels that have been guaranteed to the customers. The packet statistics are plotted in Figure 10.

Table 8: Packet Statistics

2.A.DS Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	62292	53843	8448	1

10	9195	9195	0	0
11	14564	14564	0	0
15	19235	14998	4237	0
16	38	37	1	0
17	19190	14983	4207	0
18	70	66	3	1

2.B. Broker Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	59410	59410	0	0
10	9827	9827	0	0
11	5811	5811	0	0
15	19935	19935	0	0
16	86	86	0	0
17	23713	23713	0	0
18	38	38	0	0

2.C. ARM Packet Statistics

CP	TotPkts	TxPkts	Ldrops	edrops
All	65550	65550	0	0
10	3972	3972	0	0
11	889	889	0	0
15	24904	24904	0	0
16	74	74	0	0
17	35610	35610	0	0
18	101	101	0	0

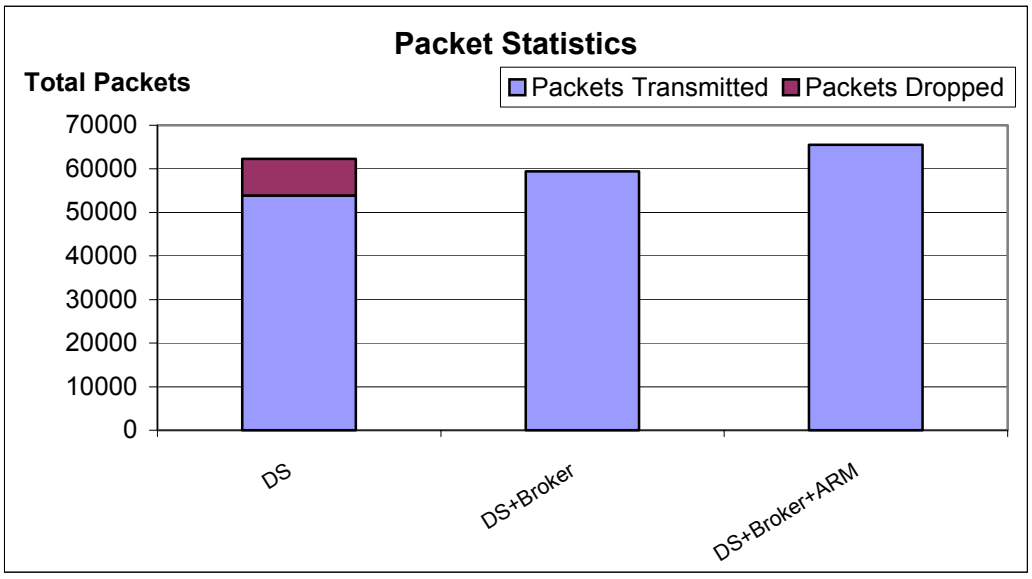

Figure 10: Packet Statistics

4.2.3 Experiment 3: Over allocation for an Extended Period of Time

For the third experiment, we extended the second experiment to stress the system by doubling the period of simulation to 80.0. It can be seen that, as the traffic increases, the broker results and the ARM algorithm are still within limits of the guarantees given, while the allocation maintains the same improved performance.

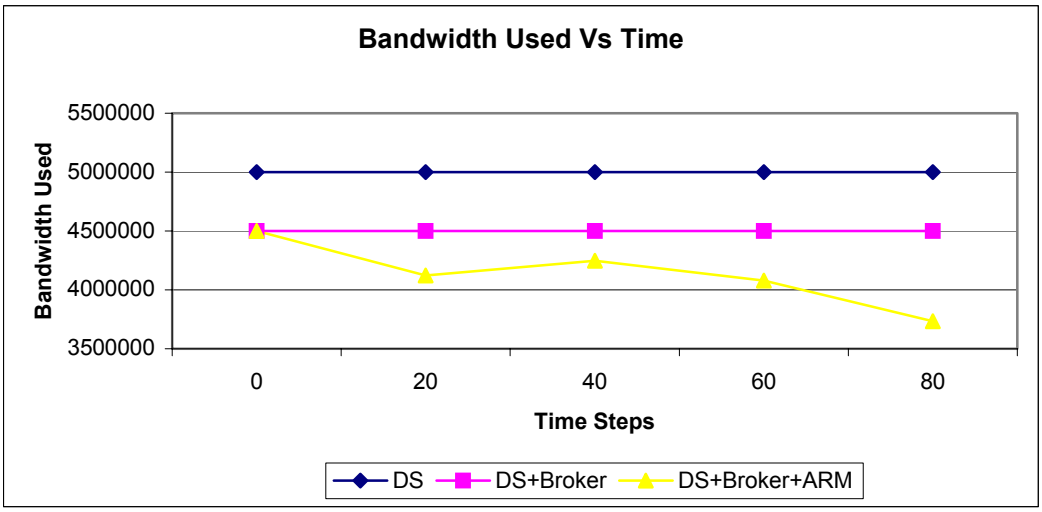

Figure 11: Bandwidth Used vs. Time

Figure 11 shows the bandwidth usage against the increased duration of the experiment. The ARM algorithm improves bandwidth utilization and conserves about 40% of the bandwidth. From the packet statistics plotted in Figure 12 we see ARM has managed to reduce number of dropped packets.

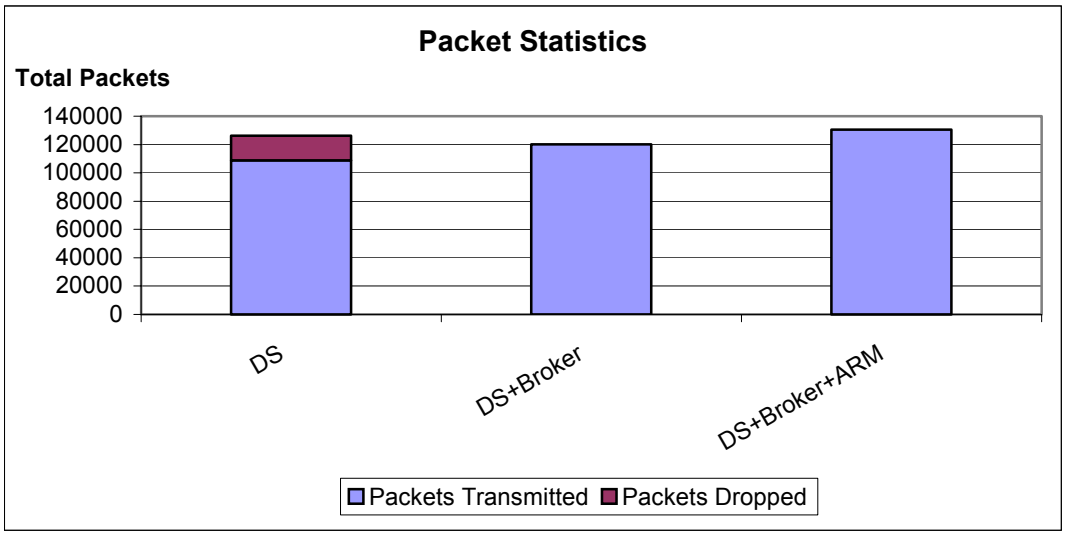

Figure 12: Packet Statistics

5 Conclusion

There is a need for guaranteed services for real time media and mission critical traffic that cannot be provided by standard IP. The Differentiated Services framework provides a suitable, scalable and less complex approach for providing these guarantees. With the help of the bandwidth broker agent, a level of intelligent resource provisioning is achieved. In this paper we presented the Active Resource Management (ARM) algorithm to further improve the level of resource allocation provided by any Differentiated Services domain. ARM reallocates the bandwidth reserved for specific clients when not used by them, to other clients. Thus ARM enables better usage of the limited bandwidth that is available. Furthermore flow throughput increases as ARM enables a larger number of flows to request resources. Along with an intelligent bandwidth broker[32, 33] an appropriate number of flows are admitted so that the network is not unduly loaded. Traffic in the Differentiated Services domain is now effectively controlled at the edge router keeping the core routers simple with little or no modification. The presented design is scalable as no state is maintained in the routers.

ARM has been implemented and evaluated using the NS-2 simulator. The experimental evaluation presented show that ARM saves at least 20% of the bandwidth allocated to individual flows, and up to a maximum of 50%. We are currently enhancing ARM to better understand application usage profiles to enable better allocation policies. We are also integrating our work with a content aware bandwidth broker architecture[34] to further optimize allocations for multimedia flows.

6 Acknowledgement

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) and by DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

7 References

[1] Chris M. IP QoS: traveling in the first class on the Internet. IEEE Internet Computing March 1999; 3(2)
[2] Blake S., Black D., Carlson M., Davies E., Wang Z., and Weiss W. An Architecture for Differentiated Services. RFC 2475. December 1998.
[3] The network simulator (NS-2) Homepage. http://www.isi.edu/nsnam/ns/
[4] Nichols K., Jacobson V. and Zhang L. A Two Bit Differentiated Services Architecture For The Internet. RFC 2638. July 1999.
[5] Braden B., Ed B., et al. Resource Reservation Protocol (RSVP) - Version 1 Functional Specification. IETF; RFC-2205, September 1997
[6] Charny A., EF PHB Redefined. Internet Draft. November 2000.
[7] Heinanen J., Baker F., Weiss W., and Wroclawski J. Assured Forwarding PHB Group. RFC 2597. June 1999.
[8] Differentiated Services testing at TF-TANT. http://www.cnaf.infn.it/~ferrari/tfng/ds/
[9] CSIRO and Autralian Academic and Research Network (AARNET) testbed.
[10] University of Kansas IPQoS project. http://qos.ittc.ukans.edu/
[11] Massachusetts Institute of Technology (MIT) Differentiated Services page. http://diffserv.lcs.mit.edu/
[12] Pieda P., Ethridge J, Baines M. and Shallwani F. A Network Simulator Differentiated Services Implementation. Open IP, Nortel Networks. http://www7.nortel.com:8080/CTL/
[13] Reichmeyer F., Ong L., Terzis A., Zhang L., and Yavatkar R. A Two-Tier Resource Management Model for Differentiated Services Networks. Internet Draft. November 1998.
[14] Qbone signaling design team. http://qbone.internet2.edu/bb/
[15] Hoo G., Johnston W., Foster I., and Roy A. QoS as middleware: Bandwidth broker system design. *Poster session for the 8th High Performance Distributed Computing conference (HPDC '99)*, Redondo Beach, California, August 1999.
[16] Policy Based Network Management Software by Orchestream. http://www.orchestream.com
[17] Policy Based Bandwidth control and application prioritization software by Extremeware. http://www.extremenetworks.com/products/datasheets/entmngr.asp
[18] Intel®, Policy-Based Network Management (PBNM). http://www.intel.com/labs/manage/pbnm/index.htm
[19] Foster I., Kesselman C., Lee C., Lindell B., Nahrstedt K., Roy A. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocations. *Proceedings of the Globus Retreat (IWQoS '99)*, June 1999. http://www.globus.org/documentation/incoming/iwqos.pdf

[20] Roy A, End to End Quality of Service for High-End Applications. PhD Dissertation, August 2001. http://www.cs.wisc.edu/~roy/publications/
[21] Veciana G., Kesidis G. and Walrand J. Resource management in wide-area ATM networks using effective bandwidths. *IEEE Journal on Selected Areas in Communications May 1995*, 1081-1090.
[22] Ogawa J., and Nomura Y. A Simple Resource Management Architecture for Differentiated Services. *Proceedings of Internet Society (INET June 2000)*, 315-438.
[23] Semeria C. Supporting Differentiated Service Classes: Active Queue Memory Management. White Paper, Juniper Networks Inc. http://www.juniper.net/techcenter/techpapers/200021.html
[24] Szabó R., Henk T., Rexhepi V., Karagiannis G. Resource Management in Differentiated Services (RMD) IP Networks. *Proceedings of the International Conference on Emerging Telecommunications Technologies and Applications, ICETA 2001*, 64- 80, Kosice, Slovak Republic.
[25] Simple Network Management Protocol Homepage. http://www.snmp.com
[26] Durham D., Boyle J., Cohen R., Herzog S., Rajan R., Sastry A. The COPS (Common Open Policy Service) protocol. Internet Draft 2748. January 2000.
[27] Heinanen J., Finland T., and Guerin R. A Single Rate Three Color Marker. Internet draft. May 1999.
[28] Heinanen J., Finland T., and Guerin R. A Two Rate Three Color Marker. Internet draft. May 1999.
[29] Clark D., and Fang W. Explicit Allocation of Best Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking August 1998;* 6(4): 362-373.
[30] Fang W, Seddigh N., and Nandy B. A Time Sliding Window Three Color Marker. Internet draft. March 2000.
[31] Welsh B., *Practical programming in Tcl and Tk*; Third Edition, Prentice Hall: 1999. (ISBN 0130220280).
[32] K. Nahrstedt, J. M. Smith, The QoS Broker, IEEE Multimedia, Vol 2, No 1, pp. 53-67, 1995.
[33] CA*net II Differentiated Services Bandwidth Broker System Specification, British Columbia Institute of Technology (BCIT), Technology Centre Group for Advanced Information Technology, 1998.
[34] M. Mahajan, M. Parashar, Content Aware Bandwidth Broker, Proc. of Fourth Annual International Workshop on Active Middleware Services. Network Services Session. 83 - 90, Edinburgh. Scotland, July 2001

Biographies

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

Ananthanarayanan Ramanathan received his MS from the Department of Electrical and Computer Engineering at Rutgers University. His research interests include parallel & distributed computing, and software engineering.

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. Manish received a BE degree in Electronics and Telecommunications from Bombay University, India and MS and Ph.D. degrees in Computer Engineering from Syracuse University.