Implementation of a Simple Bandwidth Broker for DiffServ Networks

Chamil P. W. Kulatunga, Jesse Kielthy, Paul Malone, Mícheál Ó Foghlú

Telecommunications Software Systems Group (TSSG)
Waterford Institute of Technology, Ireland.
Email: {ckulatunga, jkielthy, pmalone, mofoghlu}@tssg.org

Abstract

To date, QoS solutions that have been developed are confined to enterprise networks due to the scalability problems in inter-domain QoS provisioning. Therefore implementing a service that guarantees end-to-end QoS across the Internet has yet to be fully realised. It is believed that such a service will need to evolve regionally, then nationally and, finally, on a global scale in the same way that the Internet became a global network. By developing such a system, regional and national Network Service Providers (NSP) can be influenced more easily to provide guaranteed QoS for its users, by enabling them to generate revenue from service differentiations in IP networks.

Implementation of a QoS management framework is required in order to create end-to-end services across multiple administrative domains. Such a framework also offers easier integration of accounting, billing and security. A Differentiated Services (DiffServ) network alone is not sufficient to address the end-to-end guarantees in network layer service provision but, significantly, it does show great potential for scalability. In order to address the aforementioned problems, a centralised, QoS, management architecture has been proposed – called a Bandwidth Broker (BB). The implementation of a fully automated BB in a single administrative domain is outlined in this paper. The importance of implementing such a BB is ease of extendibility to any other QoS mechanisms such as IntServ and MPLS as well as concepts of Inter-domain QoS, Traffic Engineering, Policy QoS and Mobility Management with QoS. This paper does not intend to investigate those particular extended concepts and restricts implementation of the BB to a single DiffServ domain.

Keywords: DiffServ, QoS Management, Edge Node, Core Node, SLA, SNMP, LDAP

1. Introduction

The first major attempt by the IETF to implement end-to-end QoS for real-time and non-real time applications was known as Integrated Services, or *IntServ* [1]. It showed that significant enhancements were needed in the IP datagram model if it is to support applications where the timeliness of data and loss of packets is critical to their efficient performance. A major problem with IntServ is that there is scalability issues associated with maintaining flow state information for resource reservations. Differentiated Services, or *DiffServ* [2], was devised by the IETF as the alternative method to IntServ for supporting services in IP networks.

But scalability of DiffServ (DS) comes at the expense of an admission control mechanism. That is where we apply the management framework [3][4] described in this paper. It supports admission control as well as the authentication and accounting processes for a DS network.

2. DiffServ Architecture

The ultimate aim of the DiffServ architecture is to simplify forwarding in the core of the network and to move the processing and profiling burden towards the network edge routing agents. DS replaces the first six bits in the IPv4 Type Of Service (TOS) byte with a DS Code Point (DCSP) [5]. The TOS byte is then called the DS field (see Figure 1).

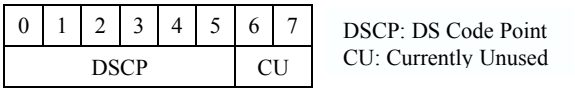

Figure 1 The DS Field

The remaining two bits are left unused to allow other networks, which require the full eight bits in the header field, to treat the packets properly.

The DS field is mapped to a specific forwarding treatment. This forwarding treatment is known as the P*er-Hop Behaviour* (PHB), and it specifies the scheduling characteristics of the packets. It does this by:

- Setting the DSCP at network edges and administrative boundaries
- Using those bits to determine how packets are treated by the routers inside the network
- Conditioning the marked packets at network boundaries in accordance with the requirements of each service

The advantage of such a scheme is that many traffic streams can be aggregated to one of a small number of Behaviour Aggregates (BA), which are each forwarded using the same PHB at the router, thereby simplifying the processing and associated storage. In addition, the fact that only information is kept about the class priorities at each hop, and not for every individual flow through the network, means that DS scales exceptionally well. In the core, routers need only inspect the DSCP in the IP packet header to determine where to send the packet next.

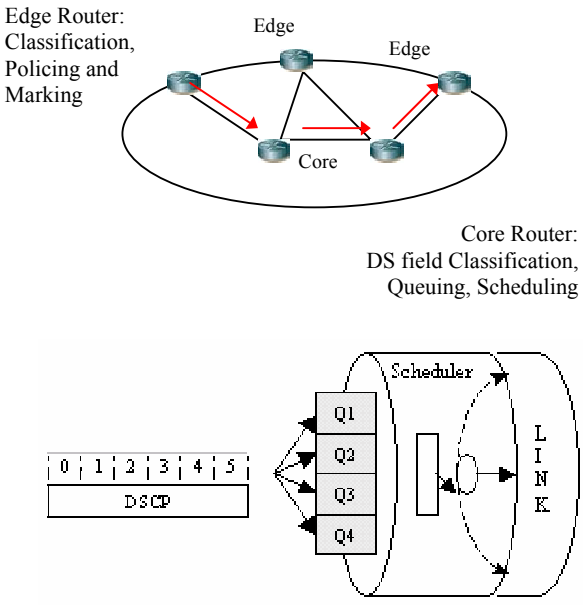

Figure 2 Scheduling Packets onto a Link

First, classification and policing are done at the ingress routers of the IP network. If the traffic does not exceed the bit-rate specified by the resource request, they are considered as *in-profile*. Otherwise, the excess packets are considered as *out-of-profile*. Third, the queue is managed by a queue management scheme.

2.1. DiffServ Traffic Classes

DS enables multiple flows to be mapped on to a single PHB. Packet flows are classified in some order before entering the network and associated with a BA. The order of classification then, will determine the level and QoS that the packets receive by the network. DS classification of packets is based on two service class types – *Expedited Forwarding* and *Assured Forwarding*.

Expedited Forwarding (EF) [6] service is the premium class of service that can be offered by the DS network. The objective is to provide tools to build a low loss, low latency, low jitter, assured bandwidth and end-to-end service through DS domains. EF requests that every network element will always service EF packets at least as fast (if not faster) than the configured rate at which the packets arrive i.e. the departure rate must equal or exceed a configurable rate. Packet drops are rare and the scheduling queues are small or empty due to this high service rate.

There is only one importance level i.e. 101110 in EF PHB (see Figure 3). Traffic flows entering the network is aggressively policed and rate shaped at the ingress nodes in order to ensure that agreed bandwidth is not exceeded.

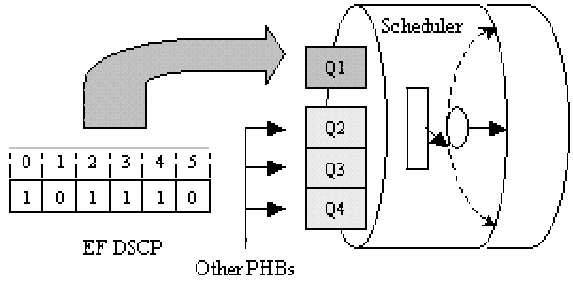

Figure 3 Scheduling EF Packets onto a Link

In EF, a simple queue management scheme suffices as all traffic with this classification has priority at the nodes and across the network. EF configuration is suitable for real-time data, where timeliness of data arrival is important.

Also, due to the fact that EF traffic will have a guaranteed bandwidth and has priority at nodes on a network, EF is suitable in congested networks to ensure application performance remains effective.

Assured Forwarding (AF) [7] is the alternative to EF. The assurance that the user of an AF service receives is that traffic is unlikely to be dropped as long as it stays within the expected capacity profile. AF has a number of PHB classes (n), and a number of Drop Precedence (DP) levels (m). Current specification defines n=4 and m=3.

This results in a total of twelve Code Points (CP) (see Table1 below).

Table 1 AF Class Code Points

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

The four AF classes define no specific bandwidth or delay constraints, other than that AF class 1 is distinct from AF class 2, and so on. The expectation is that traffic that is within the contracted rate (as measured by a token bucket) has a very much-reduced probability of being lost. When congestion is encountered at a router, higher drop precedence packets will be discarded ahead of lower drop precedence packets. Excess AF traffic is not delivered with the same probability as the traffic within the predefined profile, which means it may be demoted but not necessarily dropped (see Figure 4).

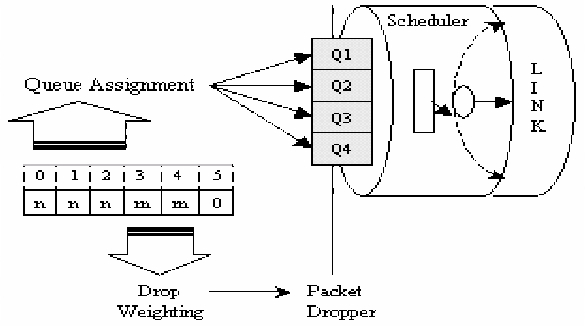

Figure 4 Scheduling AF Packets onto a Link

An AF implementation attempts to minimize long-term congestion within each class, while allowing short-term congestion resulting from bursts. This requires an active queue management algorithm. Many queuing schemes have been devised for AF e.g. Random Early Detection (RED), Generic Random Early Detection (GRED), Weighted RED (WRED), RED with In/Out (RIO) and it is important that the correct scheme be chosen so as to improve the efficiency and performance of the network.

As the rate of delivery of packets is uncertain, AF is unsuitable for applications where high QoS levels are critical to their performance. AF is perfectly suitable for applications where some degree of latency is acceptable, for example non real time audio and video streaming.

3. BB Management Framework

The ultimate goal of network QoS support is to provide users and applications with high quality data delivery services. However, from a router's view point, QoS support is made of three basic parts: defining packet treatment classes, specifying the amount of resources for each class, and sorting all incoming packets into their corresponding classes. DS addresses both the first and third issues above: it specifies traffic classes as well as provides a simple packet classification mechanism, routers easily sort packets into their corresponding treatment classes by the TOS value, without having to know which flows or what types of applications the packets belong to.

The BB addresses the second issue by keeping track of the current allocation of marked traffic and interpreting new requests in the light of the policies and current allocation. A BB will be in charge of both the internal affairs and external relations regarding resource management and traffic control. Internally, a BB may keep track of QoS requests from individual users and applications, as necessary, and allocate internal resources according to the domain's specific resource usage policies. Externally, a BB will be responsible for setting up and maintaining service agreements with the BBs of neighbouring domains to assure QoS handling of its border-crossing data traffic but beyond the scope of this paper.

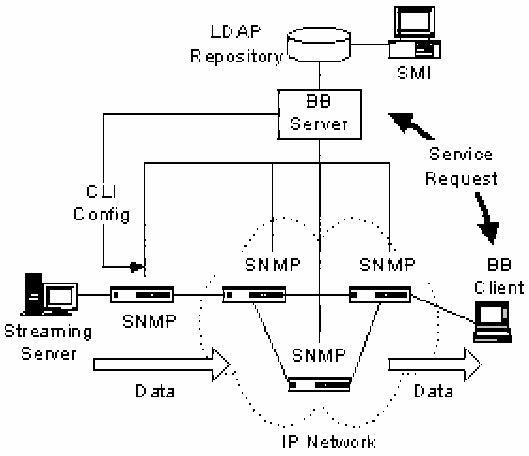

Figure 5 BB System Configuration

When a client desires an allocation for a particular flow, a request is sent to the BB (see Figure 5). This request includes details of source/destination IP

addresses, source/destination ports, requested rate and burst size, and finally the time duration for the session. A BB first authenticates the credentials of the requester, and then verifies there exists unallocated bandwidth sufficient to meet the request. If a request passes these tests, the available bandwidth is reduced by the requested amount and the flow specification is recorded in the LDAP repository.

The design of the BB system is not an innovative concept but it is the foundation work to look at the future of the QoS in the Internet. The idea of a BB was introduced as part of the DS architecture. The BB plays several roles in administering a differentiated services resource management. Two important aspects are:

Intra-domain resource management deals with allocation of resources within a single network or domain. Service Level Agreements (SLAs) are negotiated with customers and bandwidth allocation requests are satisfied or rejected depending on availability of resources. Also information about the DSCP to be set is passed on to the edge router and about the setting of classes to the egress router depending on requirements.

Inter-domain resource management is concerned with provisioning and allocating resources at network boundaries between two domains. The BB communicates with the ingress and egress border routers to configure traffic conditioners within the routers. A bilateral SLA, specifying the amount and types of traffic each side agrees to send and/or receive, must be established on the boundary between two domains.

Conducive to the implementation of the BB, a procedure is presented for the work required in developing a DS-enabled network using both hardware and software routers.

4. Implementation

4.1. DiffServ Implementation

A Cisco 3745 multi-service access router [8] is used as the edge router in the DS domain and Linux-based routers running with Zebra routing software [9] are used as core routers. OSPF (supported by Zebra) is used for the intra-domain routing protocol. Cisco IOS is used to implement DS functionalities at the ingress router and Linux traffic control (TC), which comes with iproute2 utilities [10], is used to configure DS core functionalities.

Cisco IOS version 12.2(15)T9 is installed on the Cisco router. All the interfaces used on the test-bed have 100Mbps link speed. 10 Mbps of this is configured for EF service. EF configuration of the output interfaces of Cisco router is described in the following steps.

Step 1: A class, named as *EFC*, is created that matches packets marked with DSCP 46 that use for EF PHB.

```
class-map match-all EFC
      match ip dscp ef
```

Step 2: A policy is created that applies to the traffic going through the above class. It meters the traffic and filter to 10 Mbps of committed information rate (cir) and 15 Kbytes of committed burst size and 15 Kbytes excess burst size. Any packets that conform to these parameters are transmitted, while packets that exceed the specified rates or violate the parameters are dropped.

```
policy-map EFP
    class EFC
          police cir 10000000 bc 15000 be 15000
          conform-action transmit
          exceed-action drop
          violate-action drop
```

Step 3: Finally, a policy-map is applied to the corresponding interface. It is applied as traffic going out from the interface.

```
interface FastEthernet 0/0
    service-policy output EFP
```

Linux TC utility is used for configuration of DS functionalities for the Linux based routers. Traffic filtering capabilities should be enabled in the kernel. The following shell script shows the TC configuration of 10Mbps for EF PHB.

```
tc qdisc add dev eth1 handle 1:0 root dsmark
indices 64 set_tc_index

tc filter add dev eth1 parent 1:0 protocol ip
prio 1 tcindex mask 0xfc shift 2

tc qdisc add dev eth1 parent 1:0 handle 2:0 cbq
bandwidth 100Mbit cell 8 avpkt 1000 mpu 64

tc class add dev eth1 parent 2:0 classid 2:1 cbq
bandwidth 100Mbit rate 10Mbit avpkt 1000 prio 1
bounded isolated allot 1514 weight 1 maxburst 10

tc qdisc add dev eth1 parent 2:1 pfifo limit 5

tc filter add dev eth1 parent 2:0 protocol ip
prio 1 handle 0x2e tcindex classid 2:1 pass_on

tc class add dev eth1 parent 2:0 classid 2:2 cbq
bandwidth 100Mbit rate 90Mbit avpkt 1000 prio 7
allot 1514 weight 1 maxburst 21 borrow split 2:0
defmap 0xffff

tc qdisc add dev eth1 parent 2:2 red limit 60KB
min 15KB max 45KB burst 20 avpkt 1000 bandwidth
100Mbit probability 0.4

tc filter add dev eth1 parent 2:0 protocol ip
prio 2 handle 0 tcindex mask 0 classid 2:2
pass_on
```

4.2. Central Repository

OpenLDAP server [11] on Linux is used as the central repository. The directory structure for the BB database is sub-divided into two categories - *User* and *Network* (see Figure 6). The User category contains user-dependent data and the Network category contains network-dependent data. The detailed attributes for each context are not shown in the figure.

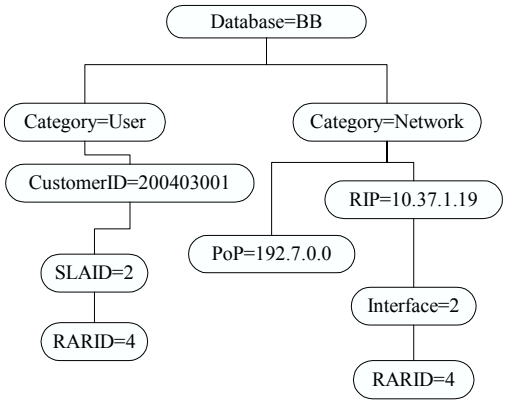

Figure 6 Repository Directory Structure

The LDAP server reads the configuration file *slapd.conf*. Schema for attributes used in BB system should be included in the file as follows.

```
include /etc/openldap/schema/BB.schema
database ldbm
suffix "Database=BB"
directory /var/lib/ldap
rootdn "Database=BB"
rootpw "***"
```

The initial configuration of the database can be added to the server using the LDAP Data Interchange Format file (in this case BB.ldif) that has the initial configurations.

4.3. Services Management

In services management, a customer can be defined as an individual, organisation or neighbouring operators in an inter-domain scenario but these requirements are beyond the discussion of this paper. Before starting an SLA for guaranteed QoS with customers, a BB Service Provider (BBSP) should define at least one service in the underlying network infrastructure in this care EF service. DS-enabled routers are used in BB administrative domains as described in above section for implementation of EF service. Prior to starting any service, a customer should agree an SLA with the BBSP with the details of the service requirements.

The BBSP uses a web interface to enable easier administration of the client SLAs. Through the Services Management Interface (SMI), the BBSP can register new users and manipulate SLAs. An Apache web server [12] is used to facilitate this. PHP enabled with LDAP and SNMP functions [13] is used for scripting. In addition, the SMI can be extended to provide value added services such as accounting and billing, report generation and network trend analysis.

Users should initially be registered with the BB system in order to get QoS-enabled services from the BB. Registration is done through a third party communication, or on-line, but this is not part of the consideration of this design. Once a user is registered with the BB service they are given a user-name, password and a customer ID. In this system, a customer ID should be unique to the system. After that, a customer can agree the SLAs with the service provider under the given customer ID. Customer ID is always used to identify the users to make them unique rather that using alphanumerical user-name or any other identity (see Figure 7).

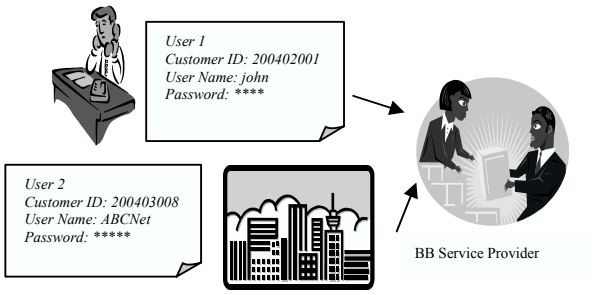

Figure 7 Authorisations and User ID to the System

After registering a customer, the BBSP should enter the user registration to the system. The web interface allows the administrator to enter the user-name and password into the system repository. The customer ID is automatically generated by the system in YYYYMMXXX format, where YYYY and MM represent the current year and the month and XXX a sequence number for the month. Once a month is completed, XXX is set back to 001 by the system.

Any user can agree more than one SLA for different services with different parameters under his customer ID. This ID makes it easier to identify the customer for billing and other administrative purposes. For example, user will get one bill for separate SLAs he has agreed under this single customer ID. It also eases the front-end administrative works such as customer care, and enables the provision of discounts or service promotions for high usage customers using this unique numerical ID from the

system. In the central repository this numerical identification is very useful for per customer identifications.

When an SLA is entered, a customer is informed with their system-generated SLA ID. This is unique per-customer and is automatically incremented in sequence.

Service Level Agreement (SLA)	
Customer ID	*200403001*
SLA ID	*6*
Service Type	*EF*
Maximum Rate	*500 Kbps*
Maximum Burst Size	*10 KByte*
Start Date	*20040101000000*
End Date	*20051231235959*

Figure 8 Customer's SLA

When the BBSP adds an SLA, first it should insert the customer ID (see Figure 8). The next three parameters specify the service and the maximum resources the user is offered. The Service Type is the service that the BB will implement. BBSP activates this agreed service at the ingress boundary of the BB domain. Then, the customer should agree for service specific parameters provided by the network operator. For EF services, only the maximum bandwidth and the maximum burst size are agreed depending on the total network capacity and the user requirements. Customers can never use a bandwidth or burst size above the limits unless the SLA is revised and agreed with the BBSP. The BBSP should be aware of the limitations of the total network capacities and maximum EF service capacities when agreeing the SLAs.

Service is only restricted with a valid time period of the SLA in this design. In addition to the time period, customer can be restricted by his SLA to authorised users, protocols, source and destination addresses or ports. The BBSP should monitor the SLAs and should not oversubscribe a service. All SLAs should be well within the network capabilities and utilisation. In addition to adding SLAs, BBSP can view all SLAs in the system and delete or update an existing SLA as required. Monitoring of the offered QoS (i.e. monitoring the application performance during resource allocation) is beyond the scope of this design.

Note: In order to enable PHP for LDAP and SNMP functionalities *extension=php_ldap.dll* and *extension=php_snmp.dll* should be uncommented in *php.ini* file for Windows.

4.4. End User System

A user should request resources for allocation for the flow duration from the BB. A TCP socket is used for customer and BB server communication. Only a user having a valid SLA ID in the system can make the resource requests.

The first window of the GUI allows the user to enter the customer ID, user name and password. Once authorisation is granted, the resource request GUI is displayed to the user to enter the resource request (see Figure 9). In the event of an authorisation failure the user is informed with an authorisation failure message and the connection from the client to the server will be discontinued.

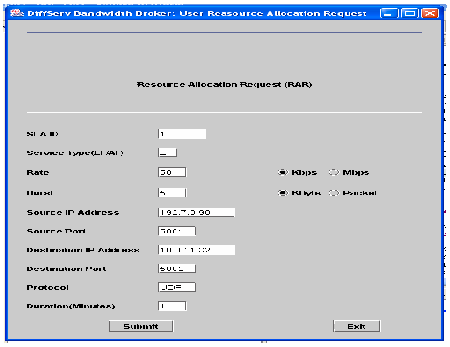

Figure 9 Resource Request GUI

Once a user is authorised to make a resource request to the BB, the customer should enter the information that the BB needs to make an end-to-end resource allocation.

Any request should be made under an SLA the customer has previously agreed. If the entered SLA ID does not match the customer ID, the user will be informed of the mismatch and asked to re-enter the request. Service type, rate and burst size, and the required time period should be entered for the resource allocation. In addition, source and destination IP addresses, port numbers and the protocol type should be entered for flow identification. The flow identification information is used as per-flow classification at the DS edge router for traffic policing.

The BB domain's Point-of-Presences (PoP) have been defined in the LDAP central repository which allows access to the network. If a PoP is found for the specified source, or if it does not recognise the port numbers or protocol type, then it should reject the request. Source or destination IP addresses should be entered in dotted decimal format and inconsistent addresses are also rejected.

Admission control failure due to non-compliance of the SLA, or lack of enough resources, will result in the session being terminated between the user and the BB server and the user being informed of the reason. In the event of a success of resource admission, the user is informed, through the use of a GUI, of the allocated resources flow information, with the specified start and elapsed time of the allocation period. A TCP session is maintained between the client and the BB server during the period of allocation. At the end of the allocation period, the user is informed of the resource termination and session is ended. If a user wants to end an allocation while it is progressing he can exit via the client side window and the server will gracefully de-allocates resources.

Requesting resources in advance can be implemented with the system. Then the user could issue requests with start and end time rather than duration. During processing, the BB server should consider the exact start and end times rather than relating to the current system time. For simplification, the design does not consider allocations for future sessions.

4.5. Bandwidth Broker Server

Basically, the BB Server (BBS) is listening for service requests from users on the specified TCP port. Java [14] is used for the implementation of BB system. Once a request is received, a parallel thread is generated which processes the request, which allows the main server to continue to listen for other requests. This section describes the functions of the BB server and how the server handles the events in processing a request. The functions of the BBS can be broken down as follows:

- Starting a thread to delete expired SLAs from the system
- Listening for resource requests on a specified TCP port
- Upon accepting a request, the request is then processed
- Continue to listen for other client requests

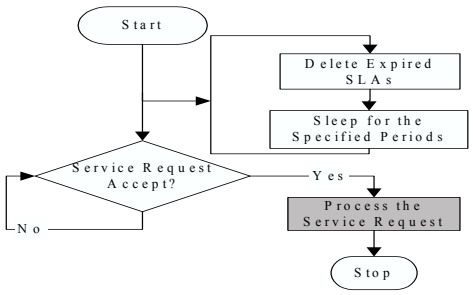

Figure 10 Flow Diagram of BB Server Main Functions

Processing a service request is described in detail in the following sections (see Figure 10). The BBS runs a thread that delete expired SLAs from the database in regular intervals. It checks the end date of a SLA and compare with the system date. The SLA deletion period can be set to a specified period.

Once the BB server accepts a request from a client, multi-threading capability is used to run a parallel thread instance, which allows the server to continue to listen for other requests. The accepted request then undergoes several stages within the thread instance. For a successful allocation, the thread will end after the period of resource allocation. In a failure, the thread will result in providing no resource allocation.

4.6. Processing of a Service Request

The server reads the password and user name from the database for the given Customer ID, and relates it to the user name and password sent by the client. If user name or password does not match, the server sends an authorisation failure message to the user and ends the thread.

If authorisation is successful, the user sends the resource request to the server. Once the server gets the resource request from the user it reads the service type, maximum rate and maximum burst for the SLA ID from the repository. In addition it checks the start date with the current system date to check whether the SLA is valid for the time that the request is made. If the requested rate and burst size is less than the agreed maximum rate and burst size, the server moves to the next function. Then the server reads and sums the rates and bursts for that SLA for the duration of the request from the repository for already allocated flows. If the difference of maximums agreed in SLA and summations of allocations for that SLA are less than the requested rate or burst, the server rejects the request and informs the client of its decision. If both are accepted, the server moves to the topology-aware admission control process.

The first function of the topology-aware admission control process is to find the edge router in respect of the user in the BB domain. Any specified source IP addresses (in resource request) should be able to communicate through one of the defined PoPs, otherwise the BB will not process a request. Next the server reads the routing table of the edge router row by row.

This process is done for each entry in the routing table until a match is found for the destination and the output interface is found. Next the allocated flows at the output interface from the repository are read to check whether the new request can be allocated the EF service based on

its bandwidth and burst size. If that interface can accommodate the request, and the destination is directly connected to the network, the topology-aware admission control process will terminate successfully and the user will be notified. If the next-hop connects to another network through a gateway, the next router is found using next-hop gateway of the routing entry. This process is done until it finds the destination domain.

When the topology-aware admission control is successful, the BB server enters that service request into a users SLA and also enters a record at each interface that the traffic will be going through. Then it performs the edge router traffic-policing configuration as described in the configuration management section. A user is informed of the success of the admission and that they are permitted to begin using the resource allocation.

At the end of the flow duration, the BB server removes the edge-policing configuration and if the flow continues after this, it will be given best-effort service. The BB server deletes the resource allocation information from the central repository. Then the user is informed about the termination of the flow allocation and record is written into an XML file for accounting proposes.

4.7. Path Discovery

ip MIB's *ipRouteTable* is used to read the routing tables for the topology-aware process. *ipRouteDest* and *ipRouteMask* are processed row by row. After logically ANDing the net mask with the provided destination IP address, this is compared with the *ipRouteDest*:

```
(Destination from Request)AND (Mask from Routing
Table) = (Destination from Routing Table)
```

That process is done until a match is found, and if no match is found, the default route is selected as the next route (see Figure 11).

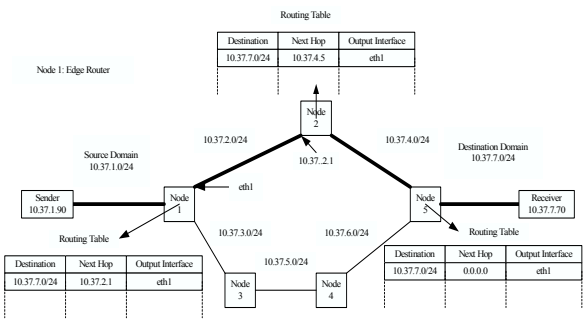

Figure 11 Path Discovery

Once a match is found, the corresponding next interface is read using *ipRouteIfIndex*. This is where the bandwidth availability checks are performed for EF service. If there are enough resources the router checks whether that is the destination hop for the path. This is done using *ipRouteType*, which shows directly connected links and, if it is not the last hop, the router reads the gateway to the next hop using *ipRouteNextHop* and find the next router. The same procedure is done at each node until it finds the destination hop. In order to determine the admission for the interface of the next hop, the router reads the EF configured bandwidth and burst size. interfaces MIB and CISCO-CLASS-BASED-QOS-MIB can be used for this purpose if all routers support QoS MIB implementations. All interfaces are considered at 100Mbps and EF bandwidth as 10Mbps throughout this design.

Note: Net-SNMP [15] is installed at each Linux router to performing SNMP agent functionalities. NetSNMPJ [16] is installed on Windows that contains Java libraries for SNMP.

4.8. Configuration Management

As has been previously stated, a Cisco router is used as the edge router in the DiffServ domain. Classification, metering, policing and marking functionalities are carried out on the input interface to the router. In addition EF is implemented on the output interfaces of the Cisco router (see Figure 12). When a flow is admitted, the BB server configures the edge router using a Telnet API as in following steps:

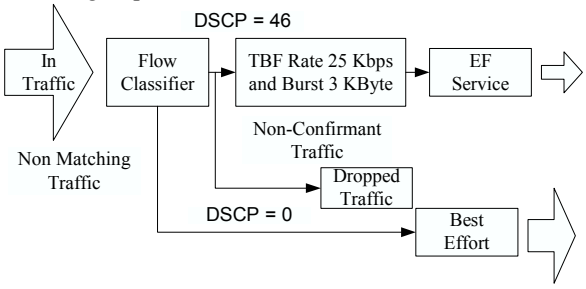

Figure 12 DiffServ Edge Policing for BB System

Step 1: An ip access-list is created for flow classification. Where RARID is the system generated ID for this flow, PRTCL, SIP, DIP and SPort are the requested protocol, source IP address, destination IP address and source port number respectively:

```
ip access-list extended AL-RARID
    permit PRTCL host SIP host DIP eq SPort
```

Step 2: A class-map is created in order to apply classified flow with traffic metering, policing and marking:

```
class-map match-all CM-RARID
    match access-group name AL-RARID
```

Step 3: The created class-map is applied to the policy that has already been created and applied to the ingress interface as an input policy:

```
policy-map EF-PMs
    class CM-RARID
        police cir 1000000 bc 5000 be 5000
        conform-action set-dscp-transmit ef
        exceed-action drop
        violate-action drop

    class class-default
        set dscp default
```

Within this policy, any flow that does not have a corresponding match even though they are marked packet with an EF DSCP by the application or host OS, are remarked to receive best-effort default service. Traffic is strictly policed for EF service by dropping non-conformant traffic. This policy is already applied at the ingress interface of the router as follows.

```
interface FastEthernet0/1
    service-policy input EF-PMs
```

The BB server should remove policing configurations relating to the flow, while permitting traffic to go through the best-effort class if a flow continues after the policy has ceased. First it should remove the class from the policy-map that applies it, and then remove the class-map and ip access-list from the configuration script respectively. A Telnet API is used to configure edge router in command line.

5. Conclusion and Future Works

This is a robust system that runs in a laboratory test-bed environment. There are practical difficulties in applying this to a commercial network environment. One example is giving users an opportunity to request bandwidth, burst size and port numbers to the end user. But with the popularity of such a system those difficulties can be minimised by allowing users to submit requests with standard - for example, requesting a videoconference using Microsoft NetMeeting in medium size window, the BB should map this high level request into detailed parameters as required.

The main objective of the basic implementation of the BB system is to further extend it in order to deploy it in real network environment. Some of the aspects that are considered for the future for the BB is to deploy it using IPv6 and use MPLS with Internet traffic engineering. A major consideration is the scalability of the system when deploying it at Internet wide scale. Deployment of another service based on the performance and cost of premium service is another consideration. This work will be based on DiffServ AF PHB.

References

[1] Braden, R. et. al. 1994 *RFC 1633 Integrated Services in the Internet Architecture: an Overview* [Online] Available: http://www.ietf.org/rfc/rfc1633.txt?number=1633/ [2003, 25 February]

[2] Blake, S. et al. 1998 RFC 2475, *An Architecture for DiffServ* [Online], Available: http://www.ietf.org/rfc/rfc2475.txt?number=2475/ [2002, October 16]

[3] Internet2 Consortium, QBone (http://qbone.internet2.edu)

[4] DANTE, Service Quality across Independently Managed Networks (SEQUIN) (http://www.dante.net/sequin)

[5] Nichols, K. et al. 1998 *RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers* [Online], Available: http://www.ietf.org/rfc/rfc2474.txt?number=2474/ [2003, March 5]

[6] Davie, C. et al. 2002 *RFC 3246, An Expedited Forwarding PHB (Per-Hop Behavior)* [Online], Available: http://www.ietf.org/rfc/rfc3246.txt?number=3246/ [2003, February 13]

[7] Heinanen, J. et al. 1999 *RFC 2597 Assured Forwarding PHB Group* [Online], Available: http://www.ietf.org/rfc/rfc2597.txt?number=2597/ [2003, February 18]

[8] Cisco Systems (http://www.cisco.com)

[9] GNU Zebra Routing Software (www.zebra.org)

[10] Differentiated Services on Linux, Traffic Control (tc) (http://diffserv.sourceforge.net)

[11] Lightweight Directory Access Protocol, OpenLDAP (http://www.openldap.org)

[12] Apache HTTP Server (http://httpd.apache.org)

[13] Script Programming, PHP (http://www.php.net)

[14] Sun Java, Sun Microsystems (http://java.sun.com)

[15] Simple Network Management Protocol, NetSNMP (http://www.netsnmp.org)

[16] Java Library for SNMP, NetSNMPJ (http://netsnmpj.sourceforge.net)