Implementation of a Bandwidth Broker for Dynamic End-to-End Resource Reservation in Outsourced Virtual Private Networks

Ibrahim Khalil, *Torsten Braun*
Institute of Computer Science and Applied Mathematics (IAM)
University of Berne
Neubrückstrasse 10, CH-3012 Bern, Switzerland
ibrahim,braun@iam.unibe.ch

Abstract

As today's network infrastructure continues to grow and Differentiated Services IP backbones are now available to provide various levels of quality of service (QoS) to VPN traffic, the ability to manage increasing network complexity is considered as a crucial factor for QoS enabled VPN solutions. There is growing trend by corporate customers to outsource such complicated management services to Internet Service Providers (ISP) not only to avoid the complexities of VPN establishment and management, but also for economic reasons.

In this paper, we present methods to provide end-to-end capacity allocation to VPN connections in a single ISP domain and show the implementation of a Bandwidth Broker managing the outsourced VPNs for corporate customers that have Service Level Agreements (SLAs) with their ISPs. We also present practical configuration examples of commercial routers for enabling QoS enabled VPN tunnels and show how the Bandwidth Broker can dynamically establish tunnels when users send connection requests from the WWW interface.

Keywords

Virtual Private Network (VPN), Differentiated Services (Diffserv, DS), Quality of Service (QoS), Bandwidth Broker (BB), Service Level Agreement (SLA), Resource Provisioning, Capacity Reservation.

1 Introduction

There is a growing demand that since private networks built on using dedicated lines offer guaranteed bandwidth and latency, similar guarantees be provided in IP based Virtual Private Networks (VPNs) [9],[14]. While the Internet has not been designed to deliver performance guarantees, with the advent of Differentiated Services [3], [2], IP backbones can now provide various levels of quality of service. The Expedited Forwarding (EF) [11] Per Hop Behaviour (PHB) is the recommended method of build such an Virtual Leased Line (VLL) type point-to-point connection for VPN. This is absolutely critical to ensure that the VPN can deliver the myriad number of benefits of this rapidly growing technology.

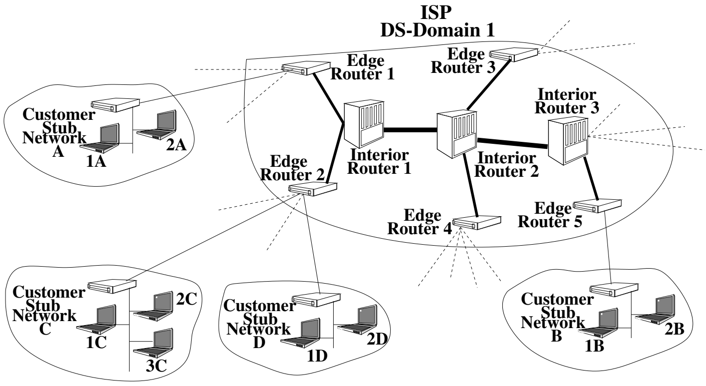

Figure 1: VPN-Diffserv Deployment Scenario

However, the complexities introduced by VPNs and the requirement to provide QoS have made the job of the ISPs and system administrators extremely difficult, and as today's network infrastructure continues to grow, there is a growing trend by corporate customers to outsource such complicated management services to Internet Service Providers (ISP). This opens the possibility for ISPs to sell VPN services to mostly corporate end users. For example, in a typical VPN-Diffserv deployment scenario (Figure 1), an ISP might offer to establish a QoS enabled VPN between stub networks A and B for an corporate end user who owns those networks and has regional offices there. To offer such services, the ISPs will, however, need a management system not only to enable the users to construct services dynamically on demand, but also able to provision the resources of it's own domain. Based on the Bandwidth Broker [15], [17], [20] approach some recent developments [19], [16], [18], [12] consider only configuring edge devices to do policing and shaping while ignoring interior provisioning of an ISP domain. Also, none of them except [12], [10] consider Dynamic VPN service creation with the BBs.

While with edge provisioning VPN connections can be allocated certain amount of resource based on SLAs (traffic contract at edge), we also need to provision the interior nodes of a transit network to meet the assurances offered at the boundaries of the network. Based on the recommendations of [2] we have, therefore, proposed a two-layered model to provision such VPN-Diffserv Networks where the top layer is responsible for edge provisioning and drives the lower layer in charge of interior resource provisioning with the help of Bandwidth Broker (BB).

In this paper, we describe the implementation of such a Bandwidth Broker (BB) not only capable of performing dynamic end-to-end admission control to setup a leased line like VPN, but also capable of managing and provisioning network resources of a separately administered Diffserv domain and cooperating with other similar domains by maintaining the topology and resource state of all nodes in the network. As World Wide Web (WWW) is widely available we provide web based interfaces as front ends where registered users can login, verify themselves and initiate a VPN based on their predefined SLA. This would obviate the need of invoking help from system administrator or ISP and at any time they can disconnect the VPN service or check their current bills.

The rest of the paper is organized as follows: section 2 gives a brief overview and concept of automated provisioning in a Diffserv domain, section 3.1 presents an edge call admission algorithm for edge provisioning and section 3.2 describes interior provisioning and end-end admission control. In Section 4 we show some examples of original VPN and QoS configuration, identify the prerequisites of a BB, and then describe various components, operational details and performance of the implemented BB. Finally, in section 5 we conclude our paper.

2 Automated Provisioning for End-to-End QoS

Provisioning in Diffserv Networks refers to determination and allocation of resources necessary at various points in the network [2]. Both quantitative, as is the case with VPN, and qualitative traffic (some assured service) are required to be provisioned at the network boundaries and in the network interior. This is achieved by a simple model [3], [4] where traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behaviour aggregates. Each behaviour aggregate is identified by a single DS codepoint. In the interior of the network, with the help of DS codepoint-PHB mapping [4], this quantitative as well as qualitative traffic can be allocated certain amount of node resources. Since we are dealing with QoS enabled VPNs, our main interest and focus will be on quantitative provisioning.

It is recommended [2] that quantitative traffic is provisioned first and then the remaining capacity can be allocated to qualitative traffic. However, it is expected that only a small fraction of a node resource will be provisioned for quantitative traffic. Determination of resources required at each node for quantitative traffic needs the estimation the traffic volume that will traverse each network node. While an ISP naturally knows from the SLA the amount of VPN quantitative traffic that will enter the transit network through a specific edge node and implement it by configuring appropriate traffic conditioning components in order to protect the provider's network, this volume cannot be estimated with exact accuracy at various interior nodes that will be traversed by VPN connections if we do not know the path of such connections exactly [1]. However, if the routing topology is known, this figure can be almost accurately estimated. For example, referring to Figure 1, assume that customer stub networks A and C want to establish VPN tunnel with stub network B and submit 5 and 10 Mbps quantitative traffic. Therefore, edge routers 1 and 2 will

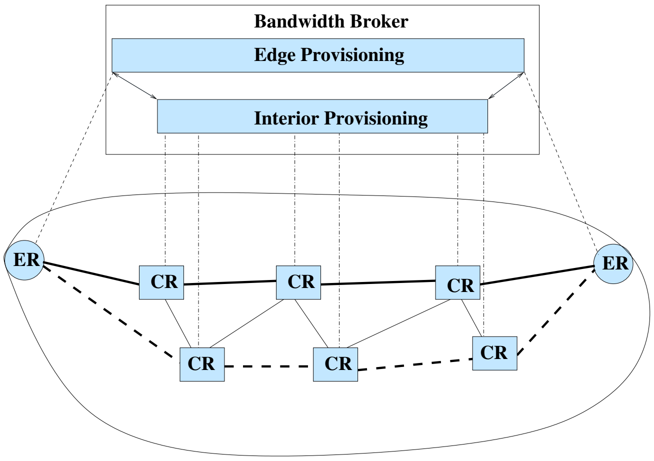

Figure 2: Layered Provisioning view of VPN-Diffserv Networks

mark the packets with DS codepoint for EF PHB and restrict the volume of quantitative traffic to 5 and 10 Mbps respectively, and since the topology of this network is simple and route follows known path, interior routers 1, 2 and 3 will need to protect these traffic by reserving at least 15 Mbps of capacity at appropriate interfaces using scheduling mechanisms as mentioned in [11]. If the default path doesn't meet the requirements of an incoming connection, alternate and various QoS routing [7], [6], [1] can also be used to find a suitable path. MPLS techniques [8] can be used to enforce the selected path.

Based on the basic needs of provisioning a VPN-Diffserv network to support quantitative service we consider the provisioning as a two layered model - the top layer responsible for edge provisioning and driving the bottom layer which is in charge of interior provisioning (Figure 2). The layers here provide the required Diffserv provisioning functionalities we have discussed earlier to create virtual leased line like services requested by customers who reside in the stub networks and outsource their services to the ISP responsible for provisioning. In the next section we will present algorithms for such provisioning applied in the Bandwidth Broker.

3 Network Resource Provisioning and Admission Control

3.1 Edge Provisioning and Call Admission at Ingress

As we have explained that provisioning is required both at the edges and in the interior and edge provisioning will drive the interior provisioning, we will first consider edge provisioning and resource sharing mechanisms required at the entry of a transit network.

First of all, ISPs need to determine the maximum amount VPN quantitative traffic that it will allow to enter the transit network from an edge router. If this figure is known incoming connections requests can be allocated the desired capacity based resource availability by performing an admission control. The job of admission control is to determine whether a VPN connection request is accepted or rejected. If the request is accepted, the required resources must be guaranteed. If C_{TOTAL} is the total quantitative capacity reserved for VPN traffic at an ingress router's interface, then $\left(C_{allocated} + C_{request} \leq C_{TOTAL}\right)$ must be true before an incoming connection request can be accepted.Here, $C_{allocated}$ is the bandwidth that is already allocated to existing tunnels, and $C_{request}$ is the requested capacity of the new incoming connection.

3.2 Interior Provisioning and End-to-End Admission

In Diffserv networks offering quantitative allocation to VPN, edge and interior provisioning are coupled with each other to a high degree in a way that each has direct influence on the other and it would not make much sense to offer guarantee only at the edges which are not met in the interior.

In order to provision the interior routes based on edge provisioning that we have described in section 3.1, we first need to know the amount of traffic that would traverse each interior node. Although provisioning a large network for such quantitative services is a difficult problem, computation of resources needed for VPN connections at various nodes can be feasible because of the following facts:

- Both ingress and egress points are known in the case of traffic submitted for quantitative VPN services. Therefore, the direction of traffic is known and traffic admitted into the network is governed by edge provisioning rules.
- Routing topology is known in advance or from interior routing protocols and stored in Bandwidth Broker database. So, VPN traffic stemming from an ingress node and directed towards an egress node traverses through some specific nodes in the interior network.

Like edge nodes, only a specific amount of bandwidth will be allocated to VPN traffic at each interior node. If a VPN connection is accepted at the edge but doesn't find enough resources provisioned for quantitative services at any of the interior nodes, the connection request will be finally rejected.

Based on idea, we will describe a simple method to estimate the capacity needed at any interior node to support traffic contract promised at the edges. Before doing that we first need to define the following terms:

- $e(I, E)$ denotes an edge pair for a VPN connection originating from ingress point I and ending at egress point E where $I \neq E$. If we have total n boundary points then $I = 1, 2, 3, ...n$ and $E = 1, 2, 3, ...n$.
- $\Re$ is the set of all edge pairs in a Diffserv domain, i.e. $\Re \subseteq [e(1,2), e(1,2), e(1,3).....e(n, n-1)]$.
- $IN(i, j)$ denotes interior routers i's jth interface where $i = 1, 2, 3, ...m$ and $j = 1, 2, ..k_i$ if we have m interior routers and any interior router i has maximum k_i interfaces.
- $\Re_{i,j}$ is the set of edge pairs that establish VPN connections which traverse through interior routers i's jth interface.
- $C(i, j)_{e(I,E)}$ is the capacity required at interior i's jth interface for VPN connection between ingress point I and egress point E.

	$IN(1,1)$	$IN(1,2)$	$\dots$	$IN(m,k)$
$e(1,2)$	$C(1,1)_{e(1,2)}$	$C(1,2)_{e(1,2)}$	$\dots$	$C(m,k_m)_{e(1,2)}$
$e(1,3)$	$C(1,1)_{e(1,3)}$	$C(1,2)_{e(1,3)}$	$\dots$	$C(m,k_m)_{e(1,3)}$
$e(1,4)$	$C(1,1)_{e(1,4)}$	$C(1,2)_{e(1,4)}$	$\dots$	$C(m,k_m)_{e(1,4)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$e(n,n-1)$	$C(1,1)_{e(n,n-1)}$	$C(1,2)_{e(n,n-1)}$	$\dots$	$C(m,k_m)_{e(n,n-1)}$

Table 1: Generalized Resource Table for end-to-end Connection Admission Control

- θ is the set of interior points in Diffserv domains, i.e. $\theta \subseteq [IN(1,2), IN(1,2), IN(1,3)....IN(m, k-1_m), IN(m,k)]$.
- $\theta_{e(I,E)} \subseteq \theta$ is the set of interior interfaces that are traversed by VPN connections having ingress point I and egress point E.

Therefore, $C(i, j)$, the resources needed for all VPN connections that traverse through a router i's j th interface can be expressed as:

$$C(i,j) = \sum_{\Re_{i,j} \subseteq \Re} C(i,j)_{e(I,E)}$$

and is actually computed from the following matrix shown in Table 1 :

In Table 1 each cell represents $C(i, j)_{e(I,E)}$. The horizontal labels indicate interfaces of interior routers and the vertical labels denote ingress/egress edge pairs. Not all cells carry numerical values since only a few of the interfaces are met by VPN traffic for a certain edge pair. Therefore, many of the cells will actually contain null values. Information regarding which interfaces are met by a VPN flow is extracted from the routing topology database used in the Bandwidth Broker.

There are numerous ways to use this matrix for call admission and resource provisioning. This matrix is basically a representation of resources currently allocated for quantitative traffic at various interior nodes for VPN traffic stemming from edges. However, ISPs again need to determine the amount of quantitative capacity that each interior router's interface will support. Let us say $C(i, j)_{upper}$ the capacity reserved at interior router i's jth interface to support the VPN traffic. If the actual capacity reserved at router i's jth interface is 50 Mbps, ISP might well set $C(i, j)_{upper}$ to slightly higher than 50 by knowing the fact not all connections will be sending at the highest rate at the same time. So, setting this value depends on how much risk ISPs want to take.

Whenever a new VPN connection request is at an ingress point destined towards an egress point, all the valid cells (not containing null values) are checked row-wise for that edge pair. If the capacity at each of interfaces are enough ,i.e. does not exceed the upper bound values even after being accepted, then with this acceptance all the cells are updated to show the most recent reservation. The end to end admission control can now be presented as follows:

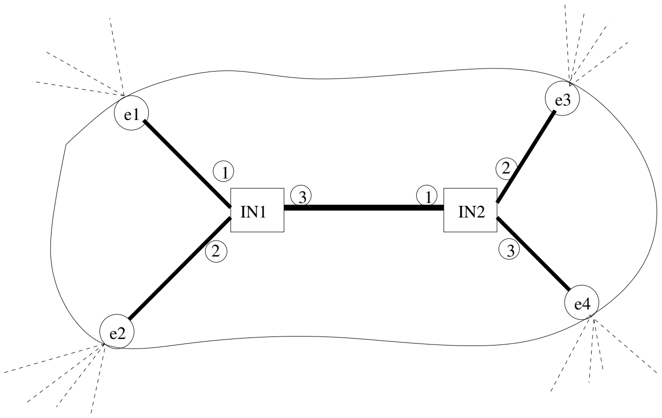

Figure 3: Topology of Network for Example 3.2.1

if edge admission OK

{

$if\left(C(i,j)_{upper} \geq C(i,j) + C_{request}\right)$ *for all* $\theta_{e(I,E)} \subseteq \theta$

{

accept connection request;

$C(i,j)_{e(I,E)} = C(i,j)_{e(I,E)} + C_{request}$ *for* $\theta_{e(I,E)} \subseteq \theta$

allocate and provision resources;

}

}

The same algorithm can be repeated for alternate routing paths (also stored in the topology database) if the default path doesn't satisfy the requirements.

Example 3.2.1

To explain the analysis and algorithms presented in this section we will consider a scenario as shown in Figure 3. In this simple case we have only two interior routers and four edge routers . For QoS allocation only uni-directional traffic flow guaranteeing and policing VPN traffic from $e1$ and $e2$ towards $e3$ and $e4$ is taken into consideration. Assume that quantitative capacity reserved by ISP at various interfaces are as follows:
$C(1,1)_{upper}$ = 50 Mbps at $IN(1,1)$, $C(1,2)_{upper}$ = 50 Mbps at $IN(1,2)$, $C(1,3)_{upper}$ = 80 Mbps at $IN(1,3)$
$C(2,1)_{upper}$ = 75 Mbps at $IN(2,1)$, $C(2,2)_{upper}$ = 50 Mbps at $IN(2,2)$, $C(2,3)_{upper}$ = 40 Mbps at $IN(2,3)$

For this example, however, only $C(1,3)_{upper}$, $C(2,2)_{upper}$, are of interest if we consider only unidirectional QoS allocation. A detailed traffic distribution before the arrival of a 1 Mbit VPN connection request at edge router $e1$ is:
1 Mbit connections: 2 connections towards $e3$, 4 connections towards $e4$
2 Mbit connections: 4 connections towards $e3$, 8 connections towards $e4$

At the same time, VPN connections stemming from ingress point $e2$ and having egress at $e3$ and $e4$ require 15 Mbps and 25 Mbps respectively, leading to the overall capacity matrix as follows:

	$IN(1,1)$	$IN(1,2)$	$IN(1,3)$	$IN(2,1)$	$IN(2,2)$	$IN(2,3)$
$e(1,2)$	-	0	-	-	-	-
$e(1,3)$	-	-	10	-	10	-
$e(1,4)$	-	-	20	-	-	20
$e(2,1)$	0	-	-	-	-	-
$e(2,3)$	-	-	15	-	15	-
$e(2,4)$	-	-	25	-	-	25

Table 2: Resource Table Before Connection Arrival

	$IN(1,1)$	$IN(1,2)$	$IN(1,3)$	$IN(2,1)$	$IN(2,2)$	$IN(2,3)$
$e(1,2)$	-	0	-	-	-	-
$e(1,3)$	-	-	11	-	11	-
$e(1,4)$	-	-	20	-	-	20
$e(2,1)$	0	-	-	-	-	-
$e(2,3)$	-	-	15	-	15	-
$e(2,4)$	-	-	25	-	-	25

Table 3: Updated Resource Table After Connection is Accepted

$$C = \begin{array}{c c} & \begin{array}{cccc} e1 & e2 & e3 & e4 \end{array} \\ \begin{array}{c} e1 \\ e2 \end{array} & \left[\begin{array}{cccc} 00 & 00 & 10 & 20 \\ 00 & 00 & 15 & 25 \end{array}\right] \end{array}$$

By extracting relevant data from the topology database for this simple network the resource table can be easily seen as in Table 2:

Clearly, $C(1,3) = C(1,3)_{e(1,3)} + C(1,3)_{e(1,4)} + C(1,3)_{e(2,3)} + C(1,3)_{e(2,4)}$ = 10+20+15 +25 = 70 Mbps. Similarly, $C(2,2) = 10 + 15 = 25$ Mbps.

With the arrival of 1 Mbit connection request at $e1$ towards $e3$, application of end-to-end admission algorithm shows that $C(1,3)_{upper} > C(1,3) + C_{request}$ and $C(2,2)_{upper} > C(2,2) + C_{request}$. Therefore, the new connection request is accepted when promise made at edge is also guaranteed in the interior and resource table is updated as shown in Table 3.

4 Dynamic Configuration Model

In this section we will identify the prerequisites for dynamic VPN service activation on demand by examining some simple configuration examples of widely used Cisco routers and based on those need and admission control mechanisms described in earlier sections, define the essential components of Bandwidth Broker that we have implemented. We will also briefly describe the operational details and performance of the implemented broker.

4.1 Example of QoS enabled VPN Configuration

In this section, referring to the experimental setup of Figure 4 we will present a configuration example in routers when an administrator needs to setup a 1 Mbps tunnel between the routers 130.92.70.101 and 130.92.66.141 for source 172.17.0.103 and destination 172.20.0.100 to meet the demand of a certain user. We need to mention that traffic flowing from source 172.17.0.103 traverses interior interfaces 130.92.70.1 and 130.92.66.1 before reaching the destination. Assume that both of these interfaces have been configured to support 10 Mbps and 15 Mbps of premium service (EF traffic)

To setup such a tunnel the administrator needs to configure appropriate commands for classification, IKE (Internet Key Exchange) policy, various IPSec tunneling/encryption methods that might be used with the tunnel, commands that specify peer

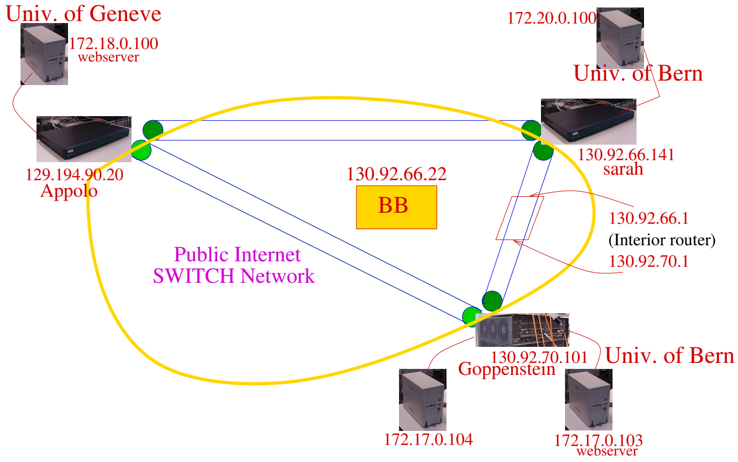

Figure 4: Experimental Setup of VPN

router's necessary identification to establish the tunnel. The classified flows also need to be possibly policed and marked at the inbound and shaped at the outbound on aggregated basis. Basically, there are mainly two functionalities that need to be provided by the edge routers: VPN tunneling, QoS (policing, shaping). For the requested tunnel to be established, the administrator, therefore, configures Router 130.92.70.101 (Cisco 7206) and 130.92.66.141 (Cisco 2611) as follows:

Configuration of Router 130.92.70.101

```
/*IKE Policy commands specifying hash method, Key sharing method,
key of peer */
1  crypto isakmp policy 1
2  hash md5
3  authentication pre-share
4  lifetime 500
5  crypto isakmp key lab-tunnel address 130.92.66.141
/*Various tunneling/encryption methods that can be
used with the requested tunnel*/
6  crypto ipsec transform-set ah-md5-hmacANDesp-des
   ah-md5-hmac esp-des
7  crypto ipsec transform-set ah-md5-hmac ah-md5-hmac
8  crypto ipsec transform-set ah-rfc1828 ah-rfc1828
9  crypto ipsec transform-set ah-sha-hmac ah-sha-hmac
10 crypto ipsec transform-set esp-encryption esp-des
/*Commands to create tunnel between routers 130.92.70.101 and
  130.92.66.141 forsource 172.17.0.103 and destination 172.20.0.100*/
11 crypto map cati-tunnel 143 ipsec-isakmp
12 set peer 130.92.66.141
13 set transform-set ah-md5-hmacANDesp-des
14 match address 143
/*Aggregate traffic shaping rate at the outbound.
Referred as C_TOTAL in section 3 */
15 interface FastEthernet0/0
16 traffic-shape group 150 10000000 1000000
   1000000 1000
17 crypto map cati-tunnel
/*policing indvidual VPN connection at the inbound./
Referred as C_request in section 3*/
18 interface FastEthernet1/0
19 rate-limit input access-group 143 1000000 2000000
   8000000 conform-action set-prec-transmit 1
   exceed-action drop
/*Classifying all VPN packets that originate from 130.92.70.101.
Used for aggregate sharing */
20 access-list 150 permit ip
    host 130.92.70.101 host any
/*Classifying the requested traffic/
21 access-list 143 permit ip
    host 172.17.0.103 host 172.20.0.100
```

Configuration of Router 130.92.66.141

```
1  crypto isakmp policy 1
2  hash md5
3  authentication pre-share
4  lifetime 500
5  crypto isakmp key lab-tunnel address 130.92.70.101
6  crypto ipsec transform-set ah-md5-hmacANDesp-des
   ah-md5-hmac esp-des
7  crypto ipsec transform-set ah-md5-hmac ah-md5-hmac
8  crypto ipsec transform-set ah-rfc1828 ah-rfc1828
9  crypto ipsec transform-set ah-sha-hmac ah-sha-hmac
10 crypto ipsec transform-set esp-encryption esp-des
11 crypto map cati-tunnel 145 ipsec-isakmp
12 set peer 130.92.70.101
13 set transform-set ah-md5-hmacANDesp-des
14 match address 145
15 interface FastEthernet0/0
16 traffic-shape group 150 15000000 1500000
   1500000 1000
17 crypto map cati-tunnel
18 interface FastEthernet1/0
19 rate-limit input access-group 145 1000000 2000000
   8000000 conform-action set-prec-transmit 1
   exceed-action drop
20 access-list 150 permit ip
   host 130.92.66.141 host any
21 access-list 145 permit ip
   host 172.20.0.100 host 172.17.0.103
```

The comments in the configuration example of router 130.92.70.101 already explain briefly which part of the scripts are responsible for the various tunneling and QoS functions. Note that in line 5 we have peer router's ip address and the *shared secret key* is the same. ISPs might actually enter the same for other peers in a domain even before establishing tunnels for between any two peers. We can also see in lines 6-10 numerous tunneling/encryption methods have been defined and only one of these is used (line 13) for the desired tunnel that is specified by the commands in lines 11-14. Also note that in line 21 traffic is classified with a specific number (access-list number) and the same number is used in lines 11, 14 and 19. Although it is not necessary to maintain the same number in lines 11 and 14 at least, this is a must in line 19 where classified packets are policed and marked. However, by maintaining the same number we can maintain a unique tunnel ID.

With all these explanations now let us examine what we need to do in order to establish another 2 Mbps tunnel between the same routers for source 172.17.0.104 and destination 172.20.0.100. If we consider only uni-directional QoS then we need policing/marking command only in router 130.92.70.101. Here the interface of 130.92.70.101 supports C_{TOTAL} =10 Mbps for quantitative VPN traffic. Earlier we accepted a 1Mbps connection i.e. $C_{allocated}$ = 1 Mbps and now $C_{request}$=2 Mbps. By performing the admission check we see that $C_{allocated} + C_{request} < C_{TOTAL}$.

Since only a 1 Mbps connection was installed previously, end-to-end admission test also signals positively, i.e. an interior router interface 130.92.66.1 also passes the admission check for this uni-directional QoS support. We can, therefore, accept the connection. All we need to do is to add the following routing commands in router 130.92.70.101 and also a similar set of commands in router 130.92.66.141.

```
crypto map cati-tunnel 144 ipsec-isakmp
set peer 130.92.66.141
set transform-set ah-md5-hmacANDesp-des
match address 144
interface FastEthernet1/0
rate-limit input access-group 144 2000000 2000000
 8000000 conform-action transmit exceed-action drop
access-list 144 permit ip host 172.17.0.103
 host 172.20.0.100
```

4.2 Prerequisites for Dynamic Configuration

We have seen from the above example how routers are actually configured (manually) and what needs to be done in the routers if additional tunnels need to established between two peer routers. Based on this experience of section 4.1 and admission control methods presented in section 3, we will now try to identify the prerequisites of an automated system like a Bandwidth Broker capable of dynamic service configuration and mimic a system administrator. These are:

- *User and Request Validity*: The system should be able to identify the user and the validity of a request.
- *VPN Readiness*: Edge routers should be *VPN Ready*. This means that IKE policy commands from line 1 to 5 and command for specifying tunneling/encryption methods (lines 6 - 10) should be pre-configured in the VPN edge routers so that when a new tunnel is configured it can choose one of previously configured tunneling/encryption methods (for example, in line 13 `ah-md5-hmac esp-des` is chosen).
- *Resource management of edge and interior routers*: Each edge and interior router need to be pre-configured to support a certain amount of quantitative traffic as specified by ISP. The system should also be able to track the resource usage so that admission control can be performed to have a well provisioned system.
- *Topology maintenance*: The system should store the topology of the network. When a connection request arrives the system should know which transit path (i.e. which interior routers the VPN connection will traverse) will be followed.
- *Management of existing connections*: The same connection that is already established cannot be established with a different tunnel ID.
- *Management of interfaces*: The system should be intelligent enough to identify which routers and interfaces should be configured.
- *Remote configuration of routers*: The system should be able to configure the appropriate routers to activate the service immediately without the invocation of a human operator.

4.3 The Essential Components of Bandwidth Broker

Based on the above requirements our Bandwidth Broker (Figure 6) has been developed to establish VPN tunnels dynamically on customers' request and also to allocate QoS to them. The BB interacts with specialized configuration daemons (CD) *(remote configuration of routers*) when a certain user request arrives to setup a tunnel and the BB has to decide whether it can allocate enough resources to meet the demand of that tunnel. The main components of BB are:

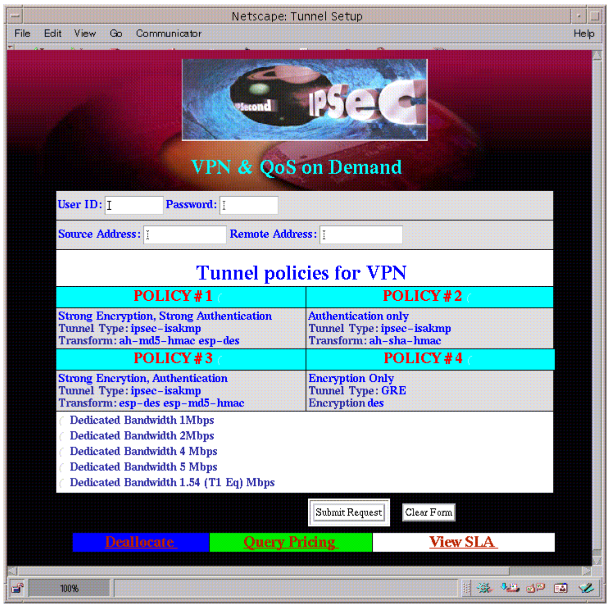

Figure 5: VPN Web interface

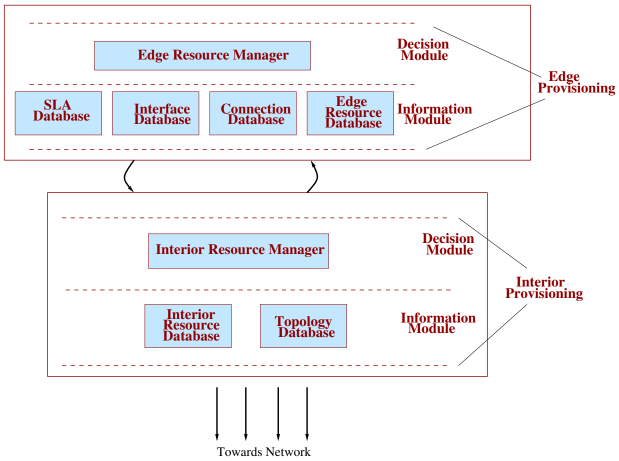

Figure 6: Components of a BB for Resource Provisioning

The **SLA database** (*user and request validity*) does contain not only the user's identification, but also specifies the maximum amount and type of traffic he/she can send and/or receive for a tunnel. As we are concerned about closed user groups, a SLA also contains the boundary of a valid VPN area. This perimeter of the valid VPN area and are put in this database as source and remote stub address'. User authentication process prohibits malicious users to setup unauthorized tunnel and access network resource illegally. The SLA, however, allows a user to add new VPN areas to his old contracted list of valid VPN areas. It contains the following tuple:

<User ID, Password, Maximum BW in Mbps, Source Stub Address, Remote Stub Address>

The **interface database** (*Management of interface*) contains necessary records of edge routers that are used as tunnel endpoints for the outsourced VPN model. In such a model since some customer stub networks are connected to the ISP edge router we need to specify which stub networks are connected to a particular edge router. Also, an edge router might have one or more inbound and outbound interfaces which also need to be specified for each stub network that is actually connected to a particular inbound interface of a router. This is important because normally at the inbound interface tunnels are policed on individual basis and at the outbound they are shaped on aggregated basis. At the same time, outbound interfaces are also used as the tunnel endpoints. Finally, a tunnel map to which all defined tunnels are attached is also part of the record in this database that is activated at the outbound interface of the router. The tuples are:

$<$ *stub network, edge router, generic router name, inbound interface, outbound interface, tunnel map name* $>$

The **connection database** (*management of existing connections*) contains a list of currently active VPNs. When a request arrives for a new VPN connection or termination of an existing connection, BB can check if that connection already exists or not and then make its decision. The storage of connections detail indicates how much resources have been consumed by VPN users at various edge and interior nodes.

$<$*user id, source address, src tunnel ID, rmt address, rmt tunnel ID, bandwidth, activation time*$>$

The **edge resource database** (*resource management of edge routers*) contains information regarding resource provisioned for different router interfaces and used (allocated) capacity to existing VPN connections. The difference between the two is the spared capacity that can be allocated to incoming connections. Referring to the notations used in section 3.1 C_{TOTAL} is total quantitative capacity dedicated to serve the VPN connection requests, $C_{allocated}$ is the allocated capacity to existing connections. The tuples are, therefore:

$<$ *edge router,* C_{TOTAL}, $C_{allocated}$ $>$

The **interior resource database** (*resource management of interior routers*) database also maintains records of quantitative resource provisioned and current resource consumption of various router interfaces. The tuples are :

$< IN(i,j)$, $C(i,j)_{upper}$, $C(i,j) >$

The **tunnel ID database** (*resource and connection management*) maintains a list unique tunnel IDs and their status for each edge router. An ID that is available is marked as 1, and the one that is used is marked as 2. The tuples are:

$<$ *edge router, Tunnel ID, Status* $>$

The **topology database** (*topology maintenance*) contains default and alternate routing paths for various VPN source destination pairs. By extracting information from this database and combining that with interior information a resource table can be created to support end-to-end admission control.

4.4 Operational Details

To show the operational details we will consider the example of section 4.1 where an administrator established a 1 Mbps tunnel for source 172.17.0.103 and destination 172.20.0.100 by manually configuring the appropriate routers. Now we will see how the BB mimics the administrator and dynamically establishes the same VPN connection when an user *castispp* sends his request from WWW. Figure 8 shows all the communications involved in setting up a VPN connection between two stub networks and Figure 9 and 10 shows the partial entries of the databases (of section 4.3) for experimental VPN setup of Figure 4. We will describe the operational details by referring to the communication marked on Figure 8 or Figure 9 and 10. Considering each communication in turn :

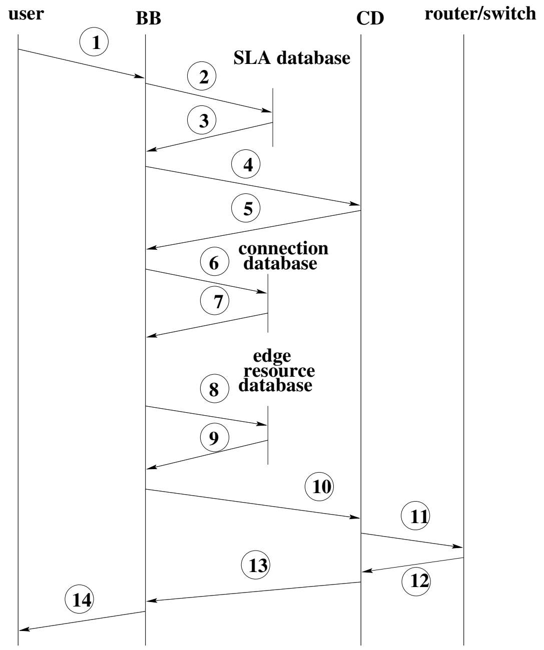

Figure 7: VPN Connection Acceptance : Only Edge Admission case

The request (step 1) contains user id *catispp*, user password, source (172.17.0.103) and remote address (172.20.0.100) for the tunnel, amount of bandwidth (1 Mbps) and encryption/tunneling method (policy #1 on webpage). The BB contacts the SLA database that is responsible for validating the user and his request. If the user is identified correctly, his source and remote address conforms the contract, and also the bandwidth requested is less than or equal to the agreed traffic contract, it sends a positive response (steps 2,3). After this validation the BB sends signal to the configuration daemon (CD) to check it's status. The status can be busy, available, or down. Only in the case of availability the user request can be processed further (steps 4,5). The BB then contacts the connection database to check the existence of an exactly similar tunnel. This is because between a source and destination only one tunnel can remain active (steps 6,7). Before edge admission control BB finds which edge routers should be configured by checking the interface database (dotted line). The BB asks the edge resource database to allocate a tunnel of a certain amount. The resource database responds to the BB and either allocates the resource or denies based on resource availability (steps 8,9). If resources are available then the BB asks the tunnel ID database for an ID (dotted line in Figure 8). If end-to-end resource allocation is necessary then the topology database (dotted line) and interior resource databases are invoked (steps 10, 11 in 8(b)). If everything goes fine routing scripts are created and appropriate routers are configured instantaneously (remaining of the steps).

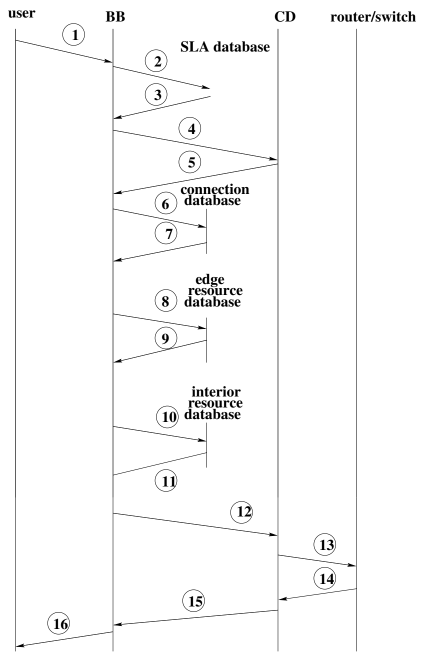

Figure 8: VPN Connection Acceptance: End-to-End Case

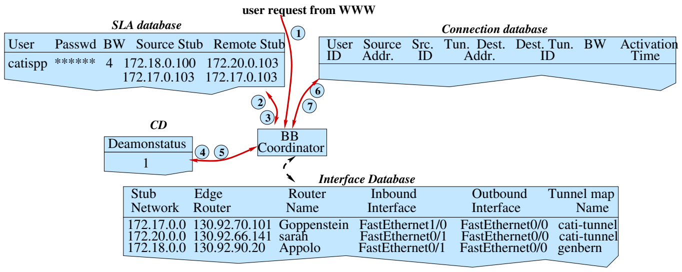

Figure 9: Partial Entries for **SLA** , **Interface Databases**. **Connection Database** does not contain any entry as no connection exists before request arrival. As *catispp* is a valid user with a valid request, **CD** status is OK, and the requested tunnel was not established before, BB proceeds further. It finds the ingress and egress routers to be 130.92.70.101 and 130.92.66.141 by a simple lookup in **interface database**.

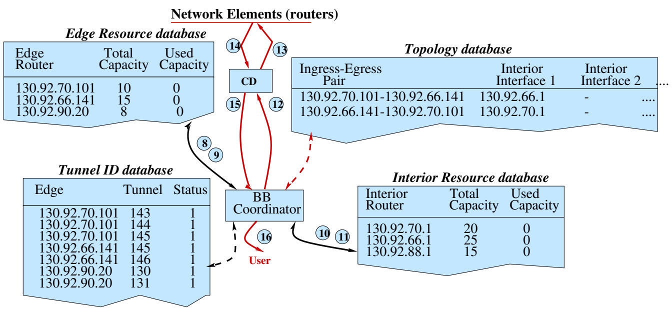

Figure 10: Partial Entries for **Edge Resource**, **Interior Resource**, **Tunnel ID** and **Topology Databases**. Admission control at 130.92.70.101 and 130.92.66.141 show that QoS can be supported in both directions at least at the edges. BB picks up tunnel ID 143 for ingress router and 145 for the egress router from the available lists in the **tunnel ID database**. For interior provisioning BB reads **Topology database** and performs admission control at 130.92.66.1 and 130.92.70.1 to check the possibility of supporting QoS in both directions on end-to-end basis.

Stream	Stream Size (bytes)	Frame Size	Frames/ Second	Bit Rate (Per Sec.) only video	Mux Rate (Per Sec) with audio	I:P:B Ratio
heuris	20595900	352x240	29.97	1.152	1.4112	15:44:41
sayit	78021332	352x240	29.97	1.856	2.4576	15:41:42

Table 4: Sample MPEG-I Bitstreams

4.5 Performance Results

For the demonstration of performance, we played some MPEG-I streams over various VPN tunnels. We selected two public domain bit streams that we believe constitute a reasonable data set.Table 4 presents the characteristics of the bit streams. Here, the bitstreams have different bit rate requirements.We believe the best metric to judge the performance of the QoS VPN tunnels is to measure to what extent the required bit rate is achieved while playing over those tunnels in real time.

Figure 11(a) and 12 (a) show the bit rate distributions of the MPEG streams of *heuris* and *sayit*. These rates consider the video frames only although the overall rates are higher. We first established a 1.5Mbps and 2.5 Mbps VPN tunnel and played *heuris* and *sayit* respectively between routers Goppenstein and sarah for webserver (source) 172.17.0.103 and station 172.20.0.100 running MpegTV [13] player. As we can see in Table 4 that *heuris* and *sayit* require 1.411 Mbps and 2.457 Mbps respectively, output traces of the received traffic as shown in Figure 11(b) and 12 (b) verifies that the allocation was adequate to transmit the streams smoothly.

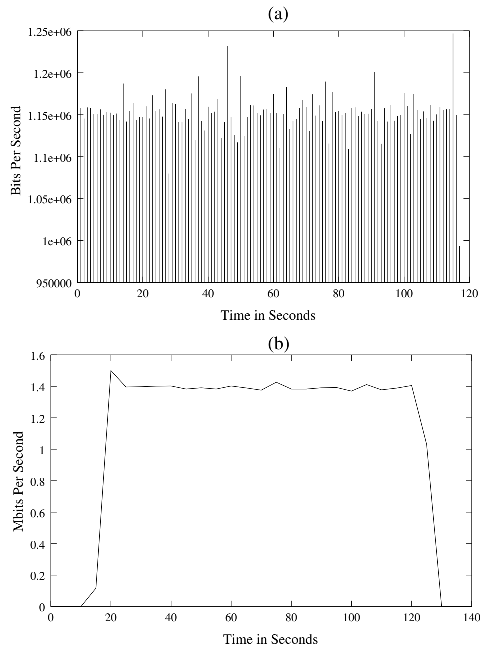

Figure 11: (a) Bit Rate Distribution of Video Framely Only for *heuris*, (b) Output Traces over a 1.50 Mbps Tunnel

The main intention here is not to find out the influence of buffer size or burst length on shaping or policing algorithm to illustrate performance changes, but rather to show the readers that such QoS mechanisms work with VPN tunnels established through our managed Bandwidth Broker system.

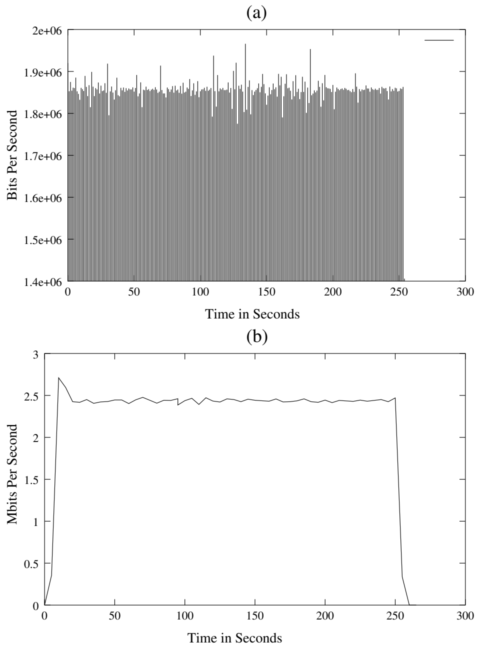

Figure 12: (a) Bit Rate Distribution of Video Framely Only for *sayit* , (b) Output Traces over a 2.5 Mbps Tunnel

5 Summary and Conclusion

In this paper, we have described the implementation of a Bandwidth Broker that is not only capable of performing dynamic end-to-end admission control to setup a leased line like VPN, but also capable of managing and provisioning network resources of a separately administered Diffserv domain. As more users are likey to outsource VPN services to ISPs, the system allows registered users to establish and terminate QoS enabled VPN tunnels dynamically from WWW. As today's network infrastructure continues to grow, the ability to manage increasing network complexity is a crucial factor for QoS enabled VPN solutions. It is estimated that almost 40 percent of the total VPN budget in an organization is spent for deploying and management of VPN and network analysts advocate outsourcing the VPN services to the ISPs [21]. Based on these industry needs we have developed our system to be deployed by ISPs that would not only alleviate the pain of corporate administrators who often need human resources and huge amount of time in such complex implementations, but also benefit the service providers form the economic point of view.

The current version of our implementation supports a single ISP domain, but we believe with some modifications the system can be easily tailored to provide end-to-end service across multiple domains and we are currently working on it.

6 Acknowledgement

The work described in this paper is part of the work done in the project Charging and Accounting Technologies for the Internet (CATI) [5] funded by the Swiss National Science Foundation (Project no. 5003-054559/1 and 5003-054560/1). The implementation platform has been funded by the SNF R Equip project no. 2160-053299.98/1 and the foundation Förderung der wissenschaftlichen Forschung an der Universität Bern.The authors would also like to thank David Billard and Noria Foukia of University of Geneva for their generous help in the experimental setup.

References

[1] Gerald R. Ash. Routing Guidelines for Efficient Routing Methods. Internet Draft `draft-ash-itu-sg2-routing-guidelines-00.txt`, October 1999. work in progress.

[2] Y. Bernet, J. Binder, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. Internet Draft `draft-ietf-diffserv-framework-02.txt`, February 1999. work in progress.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weis. An Architecture for Differentiated Services, December 1998. RFC 2475.

[4] S. Brim, B. Carpenter, and F. Le Faucheur. Per Hop Behavior Identification Codes. Internet Draft `draft-ietf-diffserv-phbid-00.txt`, October 1999. work in progress.

[5] CATI. Charging and Accounting Technologies for the Internet. http://www.tik.ee.ethz.ch/˜cati/.

[6] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network Magazine*, November/December 1998.

[7] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet, August 1998. RFC 2386.

[8] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, and P. Cheval. MPLS Support of Differentiated Services. Internet Draft `draft-ietf-mpls-diff-ext-02.txt`, October 1999. work in progress.

[9] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks. Internet Draft `draft-gleeson-vpn-framework-03.txt`, 1999. work in progress.

[10] M. Günter, T. Braun, and I. Khalil. An Architecture for Managing QoS-enabled VPNs over the Internet. In *Proceedings of the 24th Conference on Local Computer Networks LCN'99*, pages p.122–131. IEEE Computer Society, October 1999.

[11] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding phb, June 1999. RFC 2598.

[12] Ibrahim Khalil, Torsten Braun, and M. Günter. Implementation of a Service Broker for Management of QoS enabled VPNs. In *IEEE Workshop on IP-oriented Operations & Management (IPOM'2000)*, September 2000.

[13] MPEGTV. Mpeg tv home page. http://www.mpegtv.com/.

[14] Karthik Muthukrishnan and Andrew Malis. Core MPLS IP VPN Architecture. Internet Draft `raft-muthukrishnan-mpls-corevpn-arch-00.txt`, 2000. work in progress.

[15] K. Nichols, Van Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet, July 1999. RFC 2638.

[16] QBONE. The Internet2 QBone Bandwidth Broker, 2000. http://www.internet2.edu/qos/qbone/QBBAC.shtml.

[17] Olov Schelen. *Quality of Service Agnets in the Internet*. PhD thesis, Lulea University of Technology, August 1998.

[18] Benjamin Teitelbaum and et al. Internet2 QBone: Building a Testbed for Differentiated Services. *IEEE Network*, September/October 1999.

[19] Andreas Terzis, Jun Ogawa, S. Tsui, Lan Wang, and Lixia Zhang. A Prototype Implementation of the Two-Tire Architecture for Differentiated Services. In *IEEE RTAS'99*, 1999.

[20] Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang. A Two-Tier Resource Management Model for the Internet. In *IEEE Global Internet'99*, December 1999.

[21] VPNcon. Vpncon spring 2000. http://www.vpncon.com/spring/springvpn.htm.