# A Range Based SLA and Edge Driven Virtual Core Provisioning in Diffserv-VPNs

*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*—**We recently proposed a range based SLA [KB00b] approach and edge provisioning in Diffserv capable VPN networks to customers that are unable or unwilling to predict load between VPN endpoints. With range based SLA customers specify their requirements as a range of quantitative service rather than a single quantitative value and various suitable policies and algorithms dynamically provision and allocate resources at the edges for VPN connections. However, we also need to provision the interior nodes of a transit network to meet the assurances offered at the boundaries of the network. Although deterministic guaranteed service (single quantitative value approach) provides highest level of QoS guarantees, it leaves a significant portion of network resources on the average unused. In this paper, we show that with range based SLA providers have the flexibility to allocate bandwidth that falls between lower and upper bound of the range only, and therefore, take advantage of this to make multiplexing gain in the core that is usually not possible with deterministic approach. But Dynamic and frequent configuration of interior devices are not desired as this will lead to scalability problem and also defeats the purpose to Diffserv Architecture which suggests to drive all the complexities towards edges. We, therefore, propose virtual core provisioning that only requires a capacity inventory of interior devices to be updated based on VPN connection acceptance, termination or modification.**

*Keywords*— **VPN, Differentiated Services, QoS, Resource Provisioning, Admission Control, Bandwidth Broker.**

## I. INTRODUCTION

Quality of Service (QoS) enabled IP based Virtual Private Networks [GLH+99], [MM00] are highly demanded and provisioning such services dynamically on request is a challenging problem to Internet Service Providers [BGK01]. However, the advent of Differentiated Services [BBC+98], [BBC+99] with Bandwidth Broker [NJZ99] concept and Multi Protocol Label Switching (MPLS) [FWD+99] technology makes it possible to realize such services.

With Diffserv, traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint (DSCP). As Expedited Forwarding (EF) [JNP99] Per Hop Behaviour (PHB) is considered the De facto standard to build Virtual Leased Line (VLL), classified VPN traffic is marked with the DSCP for EF. In the interior of the network, with the help of DSCP - PHB mapping [NBBB98], [BCF99], this quantitative traffic can be allocated certain amount of node resources. However, if best effort routing based default paths do not meet the requirements of requested VPN connections, MPLS can be used to create pinned paths and force VPN traffic to follow paths that are provisioned with sufficient QoS.

To provide VLL type service by exploiting these emerging technologies, we [KB00a], [BGK01], [KBG00], [GBK99] and others [QBO00],[Tea99] have proposed implementation of Bandwidth Brokers. This allows users to specify guaranteed service (i.e. a single quantitative value like 1 Mbps or 2 Mbps etc.) and based on this specification the edge routers establish VPN connections dynamically and police traffic according to the specified rate. However, providing guaranteed service exactly as specified by users has following limitations:

- although deterministic guaranteed service provides highest level of QoS guarantees, it leaves a significant portion of network resources on the average unused.
- it is expected that users will be unable or unwilling to predict load between VPN endpoints [DGG+99]. From the providers point of view also, guaranteeing exact quantitative service might be a difficult job at the beginning of VPN-Diffserv deployment [BBC+99].

To address these issues in [KB00b] we recently proposed that users specify their requirements as a range of quantitative service. For example, an user who wants to establish a VPN between stub Networks A and D (Figure 1), and is not sure whether he needs 0.5 Mbps or 0.6 Mbps or 1 Mbps, and only knows the lower and upper bounds of his requirements approximately, can specify a range 0.5- 1 Mbps as his requirement from the ISP when he outsources his service to the latter. From resource provisioning point of view ISPs can take advantage of the fact that as long as lower bound of the bandwidth is guaranteed SLA will be fulfilled, and thus provision the core in way that gains from the multiplexing effect. Core provisioning, therefore, is the main focus of this paper and complements our earlier work of edge provisioning in [KB00b].

In this paper, we propose virtual core provisioning in a Bandwidth Broker architecture where edge router selects an explicit route and signals the path through the network, as in a traditional application of MPLS. Router interfaces along these routes are pre-configured to serve a certain amount of quantitative VPN traffic. A new VPN connection is subjected to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge device, but only resource state updates of core routers interfaces in the Bandwidth Broker database, and hence the naming 'virtual core provisioning'. We propose an architecture for such provisioning and show various ways to update the database in order to support VPN connection with range based SLA. We also show how we can exploit range based SLA to simplify core provisioning and make multiplexing gain and guarantee at least lower bound of bandwidth range even under heavy VPN demand conditions. Simulation results support our claims and analysis.

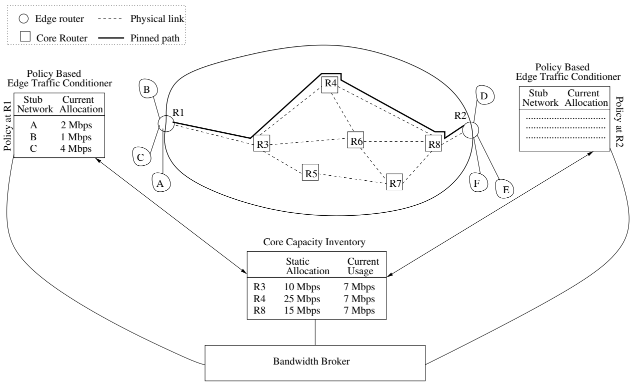


Fig. 1. Virtual Core Provisioning Architecture

## II. Virtual Core Provisioning Architecture

In diffserv enabled networks, edge provisioning drives interior (i.e core) provisioning since SLAs are contracted at the boundaries. These 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. Our Virtual Core Provisioning architecture is based on this principle where edge devices maintain all the complexities of provisioning and core devices require no explicit configuration and advance reservation states at core are maintained in a capacity inventory of Bandwidth Broker System. The architecture illustrated in Figure 1 comprises policy based edge provisioning and capacity inventory of core devices.

In order to provision the interior based on edge provisioning policies, 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 often known in advance 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 governed by MPLS and route pinning.

In the proposed Bandwidth Broker based virtual core provisioning architecture edge router selects a MPLS enabled pinned path for a VPN connection. Router interfaces along these routes are pre-configured to serve certain amount of quantitative VPN traffic. A new VPN connection is subjected to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration at the edge device, but only resource state updates of core routers interfaces in the Bandwidth Broker database. As shown in figure 1, an explicit path has been setup from router $R1$ to $R2$ that traverses core routers $R3$, $R4$ and $R8$. Each of these core routers is pre-configured to allocate 10, 25 and 15 Mbps of EF marked traffic. If a new stub network, say G (not shown in figure), gets hooked up to edge $R1$ and wants to have a 2 Mbps VPN connection to stub network $D$, this connection request will be accepted if edge $R1$ permits (core devices $R3$, $R4$ and $R8$ have enough capacity left to support this 2 Mbps connection). As a result of this acceptance, $R1$ will actually be configured with appropriate policing, shaping parameters, but only the current usage value for the core devices will updated (9 Mbps for each) in the core capacity inventory. This inventory only maintains actual pre-configured allocation and the amount reserved for accepted VPN connections.

It might seem that like Intserv or ATM based hop by hop approach, a VPN session is established by sending a signaling message to reserve resources for the new flow at each hop along the path, capacity reservation states are actually stored in a Bandwidth Broker based inventory and not in the core routers. Therefore, unlike the traditional intserv approach, which has the fundamental scalability limitations because of the responsibility to manage individually each traffic flow on each of its traversed routers, our virtual provisioning approach doesn't suffer from the same problem.

Virtual core provisioning algorithms operate in unison with the dynamic edge provisioning algorithms introduced in [KB00b] and update of core capacity inventory is driven by edge policy rules. This, along with the range based SLA that gives providers the flexibility to allocate bandwidth between lower and upper bound of the range only, makes the proposed Bandwidth Broker based virtual provisioning architecture advantageous to achieve multiplexing gain in the core that is usually not possible with Intserv like deterministic approach.

## III. Preliminaries

### A. A Novel Approach: Bandwidth Specified as an Interval

To overcome users difficulty in specifying the exact amount of quantitative bandwidth required while outsourcing the VPN service to ISPs, our model supports a flexible way to express SLAs where a range of quantitative amounts rather than a single value can be specified. Although it has several advantages, this also makes the edge and interior provisioning difficult. This complexity can be explained with a simple example. Referring to Figure 1, assume that edge router $R2$ has been provisioned to provide 20 Mbps quantitative resources to establish VPN connections elsewhere in the network and ISP has provided two options via a web interface to the VPN customers to select the rate of the connections dynamically: 1 Mbps or 2 Mbps. It is easy to see that at any time there can be 20 connections each having 1 Mbps, or 10 connections each enjoying 2 Mbps, or even a mixture of the two (e.g. 5 connections with 2 Mbps, 10 connections with 1 Mbps). When a new connection is accepted or an active connection terminates, maintaining the network state is simple and doesn't cause either reductions or forces re-negotiations to existing connections. If there are 20 connections of 1 Mbps, and one connection leaves then there will be simply 19 connections of 1 Mbps. Admission process is equally simple.

Now if the ISP provides a new option by which users can select a range 1Mbps - 2 Mbps (where 1 and 2 are the minimum and maximum offered guaranteed bandwidth), maintaining the state and admission control can be difficult. When there are up to 10 users each connection would get the maximum rate of 2 Mbps, but as new connections start arriving, the rate of existing connections would decrease. For example, when there are 20 connections this rate would be $\frac{20}{20} = 1$ Mbps and then at that stage if an active connection terminates the rate of every single connection would be expanded from 1 Mbps to $\frac{20}{19} = 1.05$ Mbps. This is a simple case when we have a single resource group supporting a range 1Mbps-2 Mbps. In reality, we might have several such groups to support users requiring varying bandwidth. In such cases, renegotiation for possible expansion of existing connections, admission control and maintenance of network states will not be simple. The idea presented here is illustrated in figure 2.

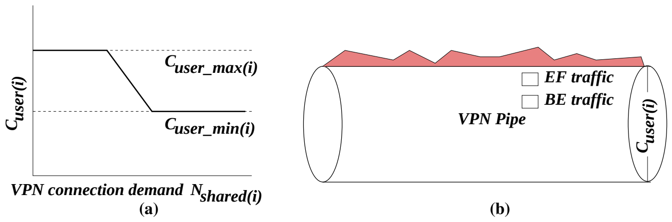


Fig. 2. The SLA approach: (a) Bandwidth is specified as an interval of $C_{user\_min(i)}$ and $C_{user\_min(i)}$ for any group $i$. Actual rate of a VPN connection $C_{user(i)}$ varies between this range but never gets below $C_{user\_min(i)}$. (b) $C_{user(i)}$ is the rate that is configured in the edge router as the policing rate. Traffic submitted at a rate higher than this rate is marked as best effort traffic or dropped depending on the policy

## B. The Model and Notations

In our model, we address this novel approach to SLAs and provide policies and algorithms for automated resource provisioning and admission control. However, to support such provisioning, we first start by allocating a certain percentage of resources at each node (edge and interior) to accommodate quantitative traffic. At the edge this quantitative portion is further logically divided between dedicated VPN tunnels (i.e. require 1Mbps or 2 Mbps explicitly) and those connections that wish to have rates defined by a range (i.e 0.5-1 Mbps or 1-2 Mbps etc.). This top level bandwidth apportionment is shown in Figure 7. The notations are :

- $C_T$ is the total capacity of a node interface.
- $C_{ded}$ is the capacity to be allocated to VPN connections requiring absolute dedicated service
- $C_{shared}$ is the capacity apportioned for those VPN connections who describe their requirement as a range.
- $C_{qual}$ is the remaining capacity for qualitative traffic.
- $C_{quan}$ is the capacity provisioned for quantitative traffic and is equal to ($C_{ded}$+ $C_{shared}$).

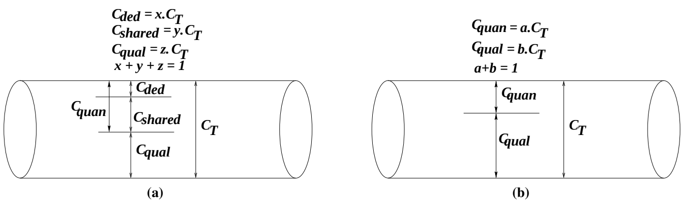


Fig. 3. Top level Bandwidth Apportionment: (a) logical partitioning at the edge, (b) logical partitioning at an interior

While at the edge $C_{quan}$ is rate controlled by policing or shaping, at the interior this $C_{quan}$ indicates that this amount of capacity will be allocated (actually protected) to quantitative traffic if need arises. All the values can be different at different nodes. This kind of logical partitioning is helpful because capacity is never wasted even if portions of resources allocated to quantitative traffic are not used by VPN connections. Unused capacity naturally goes to qualitative portion and enhances the best effort and other qualitative service. This is true both at the edge and in the interiors. $C_{shared}$, as shown in Figure 7, can be logically divided to multiple groups where each group supports a different range (Figure 4). As there might be multiple of such groups, for any group $i$ we define the following notations:

- $C_{base(i)}$ is the the base capacity for group $i$ which is shared by the VPN connections belonging to that group.
- $C_{user\_min(i)}$ is the ISP offered minimum guaranteed bandwidth that a user can have for a VPN connection.
- $C_{user\_max(i)}$ is the ISP offered maximum guaranteed bandwidth that a user can have for a VPN connection.
- $N_{shared(i)}$ is the current number of shared VPN connections in group $i$
- $C_{shared(i)}$ is the amount of capacity currently used by group $i$.
- $C_{user(i)}$ is the actual rate of active connections in group $i$ and is equal to $\frac{C_{shared(i)}}{N_{shared(i)}}$ (in section 3).
- $C_{shared\_unused}$ is the total unused bandwidth from all shared service groups.

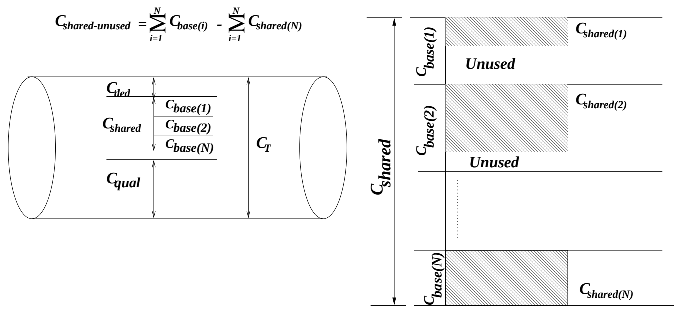


Fig. 4. Microscopic View of Bandwidth Apportionment at Edge

There are numerous sharing policies that we can apply to these shared service groups. We call them shared service groups because in reality the base capacity is shared by a certain number of VPN connections and sharing policy might allow a group to share it's resources not only among it's own connections, but also share with other groups' VPN connections in case there is some unused capacity. This may also apply to dedicated capacity. Priority can be given to certain groups while allocating unused resources. We will discuss sharing policies with examples in later sections to show how core provisioning is driven by edge based policies.

# IV. Interior Provisioning and End-to-End Admission

## A. A simple Algorithm to Update Resource Table

Like edge nodes, only a specific amount of bandwidth will be allocated to VPN traffic in each interior node. If a VPN con-

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

TABLE I
GENERALIZED RESOURCE TABLE FOR END-TO-END CONNECTION ADMISSION CONTROL

nection 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 the discussion above 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. $\Re \in [e(1,2), e(1,2), e(1,3).....e(n,n-1)]$.
- $IN(i,j)$ denotes interior routers $i$'s $j$th 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 $j$th interface.
- $C(i,j)_{e(I,E)}$ is the capacity required at interior $i$'s $j$th interface for VPN connection between ingress point $I$ and egress point $E$.
- $\theta$ is the set of interior points in Diffserv domains, i.e. $\theta \in [IN(1,2), IN(1,2), IN(1,3)....IN(m,k-1_m), IN(m,k)]$.
- $\theta_{e(I,E)} \in \theta$ 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 resource needed for all VPN connections that traverse through a router $i$'s $j$ th interface can be expressed as:

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

This is actually computed from the following matrix shown in Table I:

In table I 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 bandwidth broker.

There are numerous ways to use this matrix for all admission and resource provisioning. This matrix is basically a representation of resources currently reserved for quantitative traffic at various interior nodes for VPN traffic stemming from edges. For admission control purpose, ISPs can define a similar matrix where each cell represents upper bound value $C(i,j)_{upper}$ for quantitative traffic reservation. $C(i,j)_{upper}$ can be exactly equal to $C_{quan}$ as shown in Figure 7(a) or an over-estimated value of $C_{quan}$ to take the advantage of multiplexing effect in the interior routers where several connections are bundled and allocated an aggregated capacity. For example, if in reality $C_T = 500$, and $C_{quan} = 0.2C_T = 100$ Mbps for an interior router $i$'s $j$th interface, ISP can set $C(i,j)_{upper} = 1.5C_{quan} = 150$ Mbps to gain from multiplexing and 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. In fact, end to end admission can be presented as follows:

$if\left(N_{shared(i)} \leq \frac{C_{base(i)}}{C_{user\_min(i)}}\right)$
$\{$
*compute* $C_{user(i)}$;
$if\left(C(i,j)_{upper} > C(i,j)_{computed} + C_{user(i)}\right)$
*for all* $\theta_{e(I,E)} \in \theta$
$\{$
*accept connection request*;
$C(i,j)_{e(I,E)} = C(i,j)_{e(I,E)} + C_{user(i)}$ *for* $\theta_{e(I,E)} \in \theta$
*allocate and provision resources*;
$\}$
$\}$

Here $C(i,j)_{computed}$ is the most recent updated value of $C(i,j)$. This is because, a connection arrival, for example, might trigger changes in existing connections and if such things happen then $C(i,j)$ is computed taking these changes into consideration before end-to-end admission algorithm can decide correctly. The same algorithm can be repeated for alternate routing paths (also stored in the topology database) if the default path or MPLS based pinned path doesn't satisfy the requirements.

## V. SPECIFIC CASES OF CORE CAPACITY INVENTORY UPDATE

Based on the dynamic edge provisioning policies a new connection arrival or departure of a connection might require existing connections to reduce current rates or re-negotiate for possible expansion. Actually, such arrival or departure might force several connections to change rate not only at the edges but also in interior nodes on connection by connection basis. Although this poses some difficulties ISPs need to maintain up-to-date interior network state. Here we will present the possible cases that might happen in a network.

- case I: A new connection request arrives triggering reductions of existing VPN connections at the ingress edge.
- case II: A new call arrives which doesn't cause changes of existing VPN connections at the edge.
- case III: A call departs leaving extra capacity at the edge (as

| | $IN(1,1)$ | $IN(1,2)$ | $IN(1,3)$ | $IN(2,1)$ | $IN(2,2)$ | $IN(2,3)$ |
|---|---|---|---|---|---|---|
| $e(1,2)$ | - | 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 II

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 | - | - | - | - |
| $e(1,3)$ | - | - | 9.67 | - | 9.67 | - |
| $e(1,4)$ | - | - | 19.33 | - | - | 19.33 |
| $e(2,1)$ | 0 | - | - | - | - | - |
| $e(2,3)$ | - | - | 15 | - | 15 | - |
| $e(2,4)$ | - | - | 25 | - | - | 25 |

TABLE III

RESOURCE TABLE AFTER RELINQUISHING 1 MBPS OF CAPACITY FROM GROUP 2

unused resources) but the active connections don't need to use any portion of it.

- case IV: A call departs leaving extra resources for existing connections to be shared at the edge.

### A. Case I

In such a case, when a new connection request arrives existing connections of that group or other group(s) have to reduce their rate at the ingress because of respective sharing policy. From the resource management point of view reduction of rates of existing connections do not cause renegotiation in the interior of the network. Only the new connection negotiates at various interior points between it's ingress and egress point and if it finds sufficient resource at all points then the request is accepted and resource table for the interior is updated for this call. We will present a detailed example of this case that will explain the analysis and algorithms presented in this section

Consider a scenario as shown in Figure 5. 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 $IN(2,3)$

| | $IN(1,1)$ | $IN(1,2)$ | $IN(1,3)$ | $IN(2,1)$ | $IN(2,2)$ | $IN(2,3)$ |
|---|---|---|---|---|---|---|
| $e(1,2)$ | - | 0 | - | - | - | - |
| $e(1,3)$ | - | - | 10.67 | - | 10.67 | - |
| $e(1,4)$ | - | - | 19.33 | - | - | 19.33 |
| $e(2,1)$ | 0 | - | - | - | - | - |
| $e(2,3)$ | - | - | 15 | - | 15 | - |
| $e(2,4)$ | - | - | 25 | - | - | 25 |

TABLE IV

UPDATED RESOURCE TABLE AFTER IS ACCEPTED

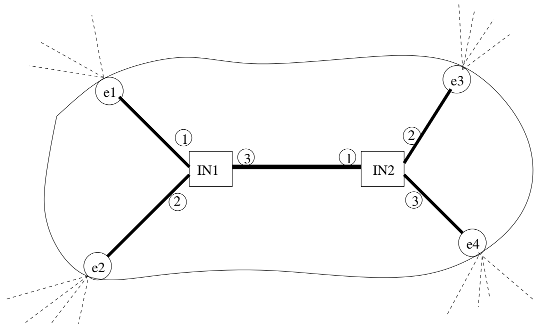


Fig. 5. Topology of Network for Example IV.1

For this example, however, only $C(1,3)_{upper}$, $C(2,2)_{upper}$ are of interest if we consider only unidirectional QoS allocation. Consider that at ingress point $e1$ capacity sharing policies are:
Group 1: $N_{shared(1)} = 6, C_{shared(1)} = 6.1 = 6$ Mbps, $C_{base(1)} = 10$ Mbps, $C_{user\_min(1)}$ = 0.5 Mbps, $C_{user\_max(1)}$= 1 Mbps and
Group 2: $N_{shared(2)} = 12, C_{shared(2)} = 12.2 = 24$ Mbps, $C_{base(2)} = 20$ Mbps, $C_{user\_min(1)}$ = 1 Mbps, $C_{user\_max(1)}$= 2 Mbps

A detailed traffic distribution before the arrival of a VPN connection request in group 1 is:
Group 1: 2 connections towards $e3$ , 4 connections towards $e4$
Group 2: 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:

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

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

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, and $C(2,3) = 20 + 25 = 45$ Mbps.

An arrival of request in group 1 for a connection towards $e3$ will allow that connection and all other existing connections in group 1 to have 1 Mbps at the ingress because $C_{base(1)} - C_{shared(1)} = 10 - 6 = 4$ Mbps and this means that group 1 hasn't used all it's base bandwidth and new connection can have th e maximum offered bandwidth 1 Mbps. This, however, reduces the share of each connection in group 2 to $\frac{23}{12}$ Mbps as that group has borrowed $C_{shared(2)} - C_{base(2)} = 24 - 20 = 4$ Mbps. Therefore, with the newly computed rates for existing connections and without taking the new connection request into consideration of computation, we have: $C(1,3)_{computed} = (2 + \frac{23}{12} * 4) + (4 + \frac{23}{12} * 8) + 15 + 25 = 69$ Mbps. Also, $C(2,2)_{computed} = (2 + \frac{23}{12} * 4) + 15 = 24.67$ Mbps. Resource table after relinquishing 1 Mbps of capacity from group

2 is shown in Table III.

Now application of end-to-end admission algorithm shows that $C(1,3)_{upper} > C(1,3)_{computed} + C_{user}(1)$ and $C(2,2)_{upper} > C(2,2)_{computed} + C_{user}(1)$. 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 IV.

### B. Case II

Consider a similar scenario of the previous example, but assume that before the arrival of a VPN connection request in group 1 (at $e1$) towards $e3$ or $e4$, we have $N_{shared(1)} = 5$ $(i.e. C_{shared(1)} = 5$ Mbps and $N_{shared(2)} = 10$ $(i.e. C_{shared(2)} = 20$ Mbps. Since no existing connections are modified at the edge, the resource table (core capacity inventory) keeping track of interior resources do not need to be updated before admission process for the requested connection can take place. However, the new connection request must check all the appropriate interior points before being finally admitted. Once accepted, the core capacity inventory is updated.

### C. Case III

This is a case when a call departs triggering no changes of existing connections in that group and also in other groups. In the previous example if $N_{shared(1)} = 10$ $(i.e. C_{shared(1)} = 10$ Mbps) and $N_{shared(2)} = 10$ $(i.e. C_{shared(2)} = 20$ Mbps) and a VPN connection departs from group 1, neither group 1 nor group 2 needs to change the rate of active connections. Interior points through which the connection was established are detected and the resource table is updated accordingly.

### D. Case IV

When a VPN tunnel is disconnected leaving extra resources for existing connections to be shared at the edge, the expandable connections having a rate less than $C_{user\_max(i)}$ need to renegotiate for possible expansion at each appropriate interior nodes. To illustrate this we will consider example presented in the previous example from the point where we stopped in that example. The final state at the edge $e1$ was:
Group 1: $N_{shared(1)} = 7, C_{shared(1)} = 7.1 = 7$ Mbps and
Group 2: $N_{shared(2)} = 12, C_{shared(2)} = 12.\frac{23}{12} = 23$ Mbps

Obviously, we had the interior resource state as is found in Table IV. Now assume that a connection departs from group 1. That leaves 1 Mbps of unused capacity that can be used to expand the existing connections in group 2. For this simple case although it is quite clear that all the existing connections will be allowed to expand to 2 Mbps and we will eventually return to the starting point of example in case I, there will be cases when not all the connections in a group will find sufficient resources at each of their appropriate interior nodes to make an end-to-end renegotiation successful. In such a case connections in the same group will have different rates. This is because, although the connections in the same group can have equal resources at the edge, this is very unlikely that connections traversing through different transit path in interior network will find equal resources on the respective path. While some connection may find only minimum offered bandwidth, others might still find maximum offered bandwidth on an end-to-end basis.

Therefore, we need to look at each connection individually and apply the end-to-end admission algorithm of section IV the same way we had earlier described in example of case I. Once again, we first have to decide how to share the unused capacity and who should have the priority to grab this resource. Such fairness issues were discussed in detail in [KB00b]. For simplicity, group with lowest base capacity has the highest priority.Since the connections might have varying rates, capacity consumed by a certain group can be $C_{shared(i)} = \sum_{i=1}^{N_{shared(i)}} C_{user(i,l)}$. $C_{user(i,l)}$ is the rate of the $l$-th connection of group $i$ where $l = 1, 2, 3......N_{shared(i)}$ and $i = 1, 2, 3......N$. Some or all of the existing connections in each group which needs to expand are also sorted according to the rate $C_{user(i,l)}$.

We will basically consider two cases. Firstly, we need to check condition $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\right)$. Here, we try to do equal expansion to all connections regardless of their current rate $C_{user(i,l)}$ by offering the addition of $\frac{C_{unused}}{N_{shared(i)}}$ to each of the connections. The goal, as usual, is to bring the rate of the expandable connections equal or close to $C_{user\_max(i)}$. Therefore, if the condition is true, then the connection is considered for possible expansion. But before we can do that, we have to check if this expansion is permitted along all the interior nodes between the VPN end points (ingress and egress). Positive answers for all the nodes finally leads to end-end expansion. $C_{unused}$ is updated as $C_{unused} = C_{unused} - \frac{C_{unused}}{N_{shared(i)}}$.

The second case, if found true, will also lead to similar end-to-end expansion. It says that even if $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} > C_{user\_max(i)}\right)$, $C_{user(i,l)}$ might be less than $C_{user\_max(i)}$. This implies that equal expansion might cause the current rate to exceed the maximum offered rate, but otherwise, is less than the maximum offered rate, and therefore, eligible for end-to-end expansion. So, the connection in question is expanded to $C_{user\_max(i)}$ and unused resource is updated as $C_{unused} = C_{unused} - [C_{user\_max(i)} - C_{user(i,l)}]$. The end-to-end admission algorithm can be presented as :

*for each ordered group* $i$ *where* $i = 1, 2, 3......N$
$\{$
*compute* $C_{shared(i)} = \sum_{i=1}^{N_{shared(i)}} C_{user(i,l)}$
*sort connections* $l = 1, 2, 3......N_{shared(i)}$ *according to rate* $C_{user(i,l)}$
*for* $i = 1$ *to* $N_{shared(i)}$
$\{$
$if\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\right)$
$\{$
*do end-to-end admission at interior points*
*if OK then expand connection to* $C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}}$
$C_{unused} = C_{unused} - \frac{C_{unused}}{N_{shared(i)}}$
$N_{shared(i)} = N_{shared(i)} - 1$
$\}$
*else if* $\left(C_{user(i,l)} < C_{user\_max(i)}\right)$
&& $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} > C_{user\_max(i)}\right)$
$\{$
*do end-to-end admission at interior points*
*if OK then expand connection to* $C_{user\_max(i)}$
$C_{unused} = C_{unused} - [C_{user\_max(i)} - C_{user(i,l)}]$
$N_{shared(i)} = N_{shared(i)} - 1$
$\}$
$\}$
$\}$
$\}$

Now let's go back to the example again. We are to find out what happens if a connection terminates from group 1.As is easily seen, this will make the resource table look like as shown in Table III. Now scanning through all the connections of group 2 and applying condition $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\right)$ of the above algorithm (actually doing admission test at each interior point in a similar way as explained in example of case I) we see that $C_{user(2,1)} + \frac{1}{12} \leq 2$, $C_{user(2,2)} + \frac{1}{12} \leq 2$, . . . . . ., $C_{user(2,11)} + \frac{1}{12} \leq 2$, $C_{user(2,12)} + \frac{1}{12} \leq 2$. Since re-negotiations of all connections are successful in the example, the resource table will finally look like what we have previously seen in Table II.

With all the examples in this section we have clearly showed how core capacity inventory can be updated based on edge provisioning policies. The four cases that we have explained with examples or referred to previous examples outlines all possible states that a node might have with a connection arrival or termination. Although we didn't show with example how a connection could choose alternate route in case the primary route doesn't meet admission criterion, it is easily understood that application of the same end-to-end admission algorithm will produce the desired result should the latter (i.e. alternate route(s)) have sufficient resources.

## VI. Simplified Core Update

To maintain exact capacity reservation states of core interfaces the update cases presented in the previous section require significant amount of computation in the bandwidth broker system and makes the VPN connection acceptance or expansion complicated in certain situations. In case I, to admit a new connection existing connections not only reduced rates at edges, but the core capacity inventory were updated for every single connection at appropriate interfaces. Even worse, in case IV, existing connections were required to renegotiate for capacity expansion at several core interfaces and a success in renegotiation triggered several core capacity updates.

Although the purpose of virtual core update is to make reservation at core accurate and consistent with edge provisioning, such complexities can actually be avoided while still guaranteeing the bandwidth promised at edge. In fact, we can simply update the appropriate core interfaces with the minimum guaranteed bandwidth each time a VPN connection is accepted and releasing the same if terminated. This is done by taking advantage of the fact that in range based SLA only lower bound capacity needs to be guaranteed and multiplexing effect in the core leaves enough room to adopt more aggressive approach and actually accommodate more connections than is possible if $C_{user(i)}$ is used for virtual core updates.

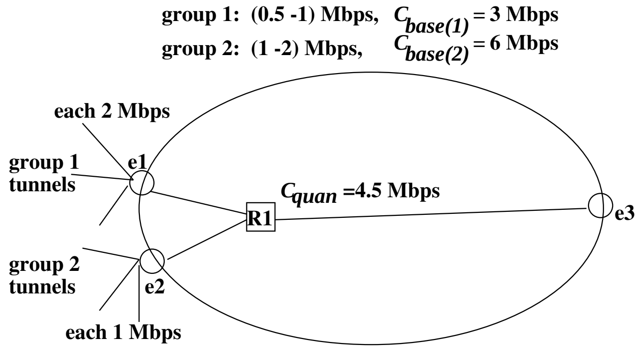


Fig. 6. Worst case Scenario. If all connections send traffic at max. configured rate some of them might not get minimum guaranteed capacity

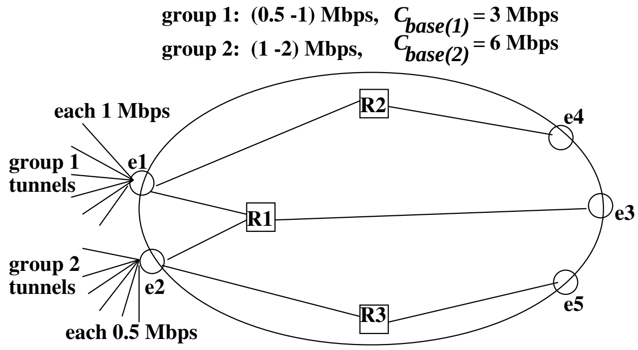


Fig. 7. Heavy VPN demand. Arrival of more connections make sure that old connections get at least min. guaranteed bandwidth

We will explain with an example here before presenting simulation to support our idea. Consider a scenario where edge $e1$ accommodates group 1 requiring (0.5-1) Mbps with $C_{bases(1)}$= 3 Mbps. Another edge $e2$ supports group 2 requiring (1-2) Mbps with $C_{bases(1)}$= 3 Mbps. Core interface 1 is configured to allocate 4.5 Mbps premium traffic. (i.e $C(i,j) = C(i,j)_{upper} = 1.C_{quan}$=4.5 Mbps ). Currently, three 1 Mbps VPN connections at $e1$ and another three 2 Mbps connections are active. As we update core capacity inventory with $C_{user\_min(i)}$ rather than $C_{user(i)}$, each time a (1-2) Mbps connection gets accepted we increment $C(i,j)$ (for core interface 1) with $C_{user\_min(2)}$=1, and also similarly for (0.5- 1) Mbps connection acceptance. Although the the probability of acceptance increases (i.e blocking probability decreases), in the worst case if all the accepted connections send traffic at the maximum configured rate at the same time, some connections might not even get the minimum guaranteed bandwidth.

However, by law of large number, as more connections are accepted at edge, probability of each connection getting the minimum bandwidth increases. This is true for our example where acceptance of 3 more connections of existing types at both $e1$ and $e2$ (destine towards $e3$ and $e4$) ensures that every single accepted connection gets the lower bound of the bandwidth range even in the worst case.

## VII. Simulation

In this section, we present simulation results to show average rate achieved by accepted VPN connections in a relatively large network under different demand conditions. Simulation studies presented here obviously consider simplified core update cases and confirms earlier analysis presented in the previous section.

Recent trend on achieving multiplexing gain relies on the assumptions that connections (flows) are statistically independent and smoothed by deterministic regulators at the connections input to the network since statistical characterization of traffic sources is not often reliable [BBLO00], [RRR98]. Not surprisingly, this exactly resembles our case. VPN connections are rate controlled based on provisioning policies at provider edge. In fact, many of the results derived in those will, therefore, be valid in our case too. One interesting result is: by statistically multiplexing rate controlled (at edge) traffic in the core network number of accepted connections can be three times higher than that of Generalized Processor Sharing or any other deterministic service discipline [RRR98].

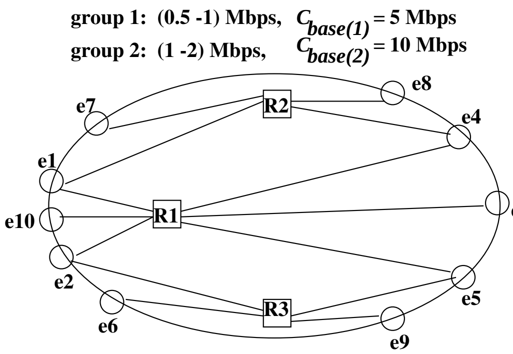


Fig. 8. Experimental Setup for Simulation

The simulation setup that we consider for our experiment is as shown in figure 8. This network has 10 edge nodes and a total of 10 core interfaces from 3 core routers. Each edge node can accept a maximum of 10 connections from each group when sending at lower bound rate. As there are 10 edge nodes, a total of 150 Mbps might enter the transit network at a time. Since there are 10 interior interfaces, we configure each interface with 15 Mbps on average.

Figure 9 plots average bandwidth achieved by 20 connections from each group over a period of 1 hour. During this one hour period 70 connections from each group were actively sending traffic between a range of minimum and maximum allowable bandwidth (i.e 0.5-1 Mbps for group 1 and 1-2 Mbps for group 2) to the network. However, the 40 connections (20 from each group) selected for plotting were accepted at edge to send traffic at the highest possible rate and were actually spraying traffic at that rate (i.e. 1 and 2 Mbps for group 1 and 2 respectively). Figure 10 also shows average of 20 connections, but in this case 60 connections from each group were active. Obviously, average rate improved slightly in this case. What is important to note here is: although we provision and update the core with less capacity than that is needed for maintaining exact core capacity inventory, accepted VPN connections were receiving almost the upper bound capacity.

One fundamental drawback of deterministic service is that, by its very nature, it must reserve resources according to a worst case scenario, and hence has limits in its achievable utilization. To overcome the utilization limits of deterministic service, statistical multiplexing must be used assuming that worst case scenario will quite rarely occur. The worst case scenario is a bit different in our case. This might happen when a core interface is configured to support minimum guaranteed bandwidth no matter what edge allocates to accepted connections, and all the connections start sending at their fullest configured rate. However, as the number of accepted connections increases, probability that worst case might happen starts diminishing. This is shown in Figure 11 where we plot the average of 30 accepted connections from each group where each connection was configured with lower bound capacity at the edge and number of total accepted connections during the 1 hour measurement period was 85 from each group. This also confirms our previous analysis in section VI.

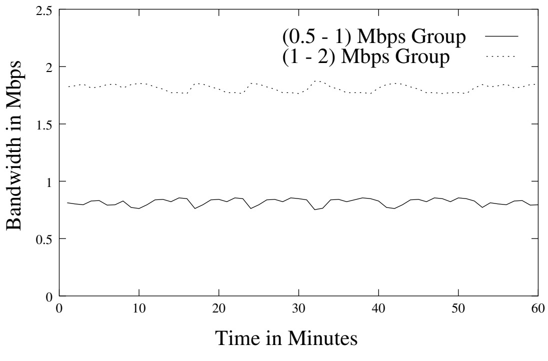


Fig. 9. Simulation Result 1: Average of 20 connections. Total accepted connections 70

## VIII. Summary and Conclusion

In this paper, we have proposed virtual core provisioning in a Bandwidth Broker architecture for QoS enabled VPN connections. As users of such connections are unable or unwilling to predict load between the VPN endpoints we recently proposed that customers specify their requirements as a range of quantitative service in the Service Level Agreements (SLAs) for VPN connections. We show how we can exploit range based SLA

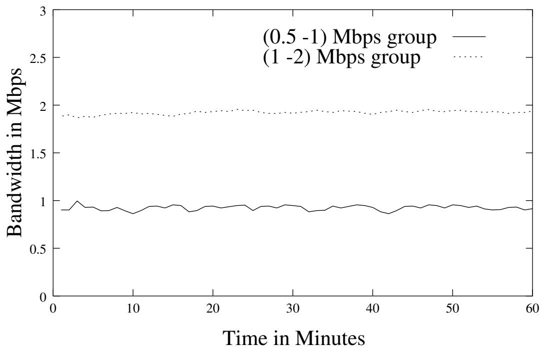


Fig. 10. Simulation Result 2: Average of 20 connections. Total accepted connections 60

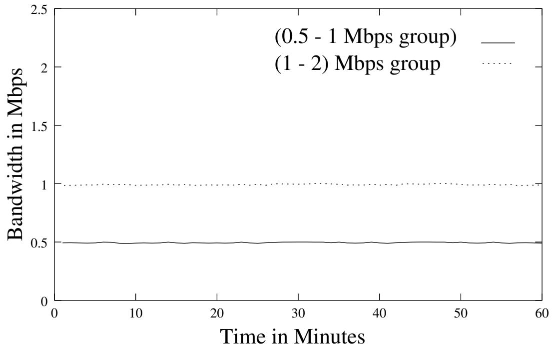


Fig. 11. Simulation Result 3: Average of 30 connections. Total accepted connections 85

to simplify core provisioning and make multiplexing gain and guarantee at least lower bound of bandwidth range even under heavy VPN demand conditions. Simulation results support our claims and analysis.

In our virtual core provisioning architecture edge router selects an explicit route and signals the path through the network, as in a traditional application of MPLS. Router interfaces along these routes are pre-configured to serve certain amount of quantitative VPN traffic. A new VPN connection is subjected to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge device, but only resource state updates of core routers interfaces in the Bandwidth Broker database, and hence the naming 'virtual core provisioning'.

The centralized BB in it's role as a global network manager maintains information about all the established real-time VPN tunnel and the network topology, and can thus select an appropriate route for each real-time connection request. If a pinned path or pre-selected alternate routes fail to reserve requested resources for a VPN connection, QoS routing can be then efficiently used. Since the objective of any routing algorithm is to find a qualified path with minimal operational overheads centralized BB based QoS routing might be very effective.This is an issue we have not addressed and can be a future research topic.

## References


[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weis. An Architecture for Differentiated Services, December 1998. RFC 2475.

[BBC+99] 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.

[BBLO00] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Statistical service assurances for traffic scheduling algorithms. *IEEE Journal on Selected Areas in Communications*, 18(12), December 2000.

[BCF99] 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.

[BGK01] Torsten Braun, M. Günter, and Ibrahim Khalil. Management of quality of service enabled vpns. *IEEE Communications Magazine*, 39(5), May 2001.

[DGG+99] N.G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K.K. Ramakrishnan, , and Jacobus E. Van der Merwe. A Flexible Model for Resource Management in Virtual Private Networks. *SIGCOMM'99 Conference*, August 1999.

[FWD+99] 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.

[GBK99] 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.

[GLH+99] 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.

[JNP99] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding phb, June 1999. RFC 2598.

[KB00a] I. Khalil and T. Braun. Implementation of a Bandwidth Broker for Dynamic End-to-End Resource Reservation in Outsourced Virtual Private Networks. *The 25th Annual IEEE Conference on Local Computer Networks (LCN)*, November 9-10 2000.

[KB00b] Ibrahim Khalil and Torsten Braun. Edge Provisioning and Fairness in DiffServ-VPNs. *IEEE International Conference on Computer Communication and Network (I3CN)*, Oct 16-18 2000.

[KBG00] 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.

[MM00] Karthik Muthukrishnan and Andrew Malis. Core MPLS IP VPN Architecture. Internet Draft `raft-muthukrishnan-mpls-corevpn-arch-00.txt`, 2000. work in progress.

[NBBB98] K. Nichols, S. Blake., F. Baker, and D. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers, December 1998. RFC 2474.

[NJZ99] K. Nichols, Van Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet, July 1999. RFC 2638.

[QBO00] QBONE. The Internet2 QBone Bandwidth Broker, 2000. http://www.internet2.edu/qos/qbone/QBBAC.shtml.

[RRR98] M. Reisslein, K. W. Ross, and S. Rajagopal. Guaranteeing statistical qos to regulated traffic: The multiple node case. *In Proceedings of 37th IEEE Conference on Decision and Control (CDC),Tampa*, December 1998.

[Tea99] Benjamin Teitelbaum and et al. Internet2 QBone: Building a Testbed for Differentiated Services. *IEEE Network*, September/October 1999.