ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

C. Chhabra, T. Erlebach, B. Stiller, D. Vukadinović

# Price-Based Call Admission Control in a Single DiffServ Domain

TIK-Report
Nr. 135, May 2002

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

C. Chhabra, T. Erlebach, B. Stiller, D. Vukadinović
Price-Based Call Admission Control in a Single DiffServ Domain
May 2002
Version 1
TIK-Report Nr. 135

---

Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology (ETH) Zurich

Institut für Technische Informatik und Kommunikationsnetze,
Eidgenössische Technische Hochschule Zürich

Gloriastrasse 35, ETH Zentrum, CH-8092 Zürich, Switzerland

# Price-Based Call Admission Control in a Single DiffServ Domain*

C. Chhabra, T. Erlebach, B. Stiller†, D. Vukadinović

Computer Engineering and Networks Laboratory (TIK)
ETH Zürich, CH-8092 Zürich, Switzerland
{chhabra|erlebach|stiller|vukadin}@tik.ee.ethz.ch

**Abstract**

We consider the problem of scalable call admission control (CAC) in a DiffServ network and propose a new CAC scheme. Initially, a certain fraction of the bandwidth of the network is reserved for admission-controlled traffic, and this reservation is carried out according to the hose model, a model that had previously been proposed for VPN provisioning. With this architecture, admission control can be carried out at edge routers without any need for global coordination. Following the DiffServ philosophy, routers in the core of the network are not involved in CAC or signaling and do not need to be aware of individual flows. This ensures scalability while offering deterministic bandwidth guarantees.

Assuming that different connections create different amounts of profit for the provider, we consider CAC algorithms that try to maximize the obtained profit. In the on-line version of the problem, we observe that the algorithm by Awerbuch, Azar and Plotkin can be implemented using local control and achieves a logarithmic competitive ratio. In the off-line version, where all connection requests are known to the algorithm, we give efficient approximation algorithms achieving good approximation ratios. These algorithms are useful in a scenario with advance reservations.

We also give a detailed specification of a possible implementation of the proposed CAC architecture, and present results of high-level simulations showing the relative performance of different CAC algorithms and the influence of a user model where users may sometimes be willing to accept an increased price for their connection.

## 1 Introduction

Multimedia applications such as video conferencing or tele-teaching are very sensitive to network congestion. Therefore, it seems likely that guaranteed quality of service will be an increasingly

*Research supported by the Swiss National Science Foundation under Contract No. 5003-057753/3, ANAISOFT-PCAC.

†This author is mainly with the University of Federal Armed Forces Munich, Germany, Information Systems Laboratory IIS, D-85577 Neubiberg, Germany.

important feature of communication networks. In order to achieve guaranteed quality of service, resources in the network must be allocated to each admitted connection to ensure that packets can be transmitted at the required rate without suffering packet losses due to congestion. The task of deciding for each connection request whether it is to be admitted or not (and of allocating resources to the admitted requests) is generally referred to as *call admission control* (CAC). This is a non-trivial task, since there are many optimization criteria that are partly contradictory: one wants to achieve high utilization, low rejection rates, deterministic service guarantees, high profit for the network provider, fairness, etc. Furthermore, *scalability* is an important issue, because a large number of connection requests may have to be handled by the network and so the overhead imposed by the call admission control procedure needs to be minimized.

We consider the call admission control problem from the perspective of one autonomous system (subnetwork under separate administrative control) in the Internet. The routers in this network are either *edge routers* (routers that are connected to routers in other autonomous systems or to hosts) or *core routers*. We assume that the network is a DiffServ network (see [23]), but our approach does not rely on this assumption and generalizes to other scenarios as well. We consider a network that carries two types of traffic: admission-controlled traffic (ACT) and best-effort traffic (BET). ACT is made up of connections that request a certain bandwidth and must be guaranteed to get that bandwidth if they are established. BET is made up of connections with flow control (such as TCP) that can adapt their sending rates to the currently available bandwidth and do not require any bandwidth guarantees. Only the ACT is handled by the call admission control procedure.

In this paper we propose an architecture for implementing call admission control for the above scenario in a scalable way. Requirements for the proposed architecture are:

- If a connection is admitted, it must receive guaranteed quality of service.
- Call admission control should be handled by edge routers.
- Core routers should not be involved in admission control and signaling.
- If different connections yield a different revenue for the provider, the architecture should enable price-based call admission control (PCAC), e.g. rejecting low-profit connection requests even if there is enough bandwidth.

Since core routers are not involved in admission control and signaling, a good scalability is achieved.

In our proposed architecture, described in Section 2, we make use of bandwidth reservations according to the so-called *hose model*: On every link of the network, a certain amount of bandwidth is allocated to ACT, and all routers ensure that ACT can indeed use the assigned fraction of the link bandwidth as long as there are ACT packets in the queue. BET can always use up the remaining bandwidth of the link. The allocations are done in such a way that any traffic pattern can be routed in the network provided that pre-defined ingress or egress limits at the edge routers are not violated. The main advantage of this model is that the edge routers can decide locally whether the network has sufficient bandwidth for a newly requested connection.

Then, in Section 3, we consider different algorithms for making the decisions of call acceptance or rejection in this architecture. We assume that each request is associated with a certain profit that is obtained by the provider if the request is admitted. Since different connections (e.g., connections of different DiffServ classes) may yield a different profit, an important goal of the

provider is to maximize the total profit of all admitted connections. We notice that call admission control in this architecture (with bandwidth reservations for ACT according to the hose model) is equivalent to call admission control in star networks. In the scenario where each request must be handled immediately (without knowledge of future requests), we consider the algorithm by Awerbuch, Azar and Plotkin [2] that achieves a logarithmic competitive ratio (i.e., its profit is smaller than the optimal off-line solution at most by a logarithmic factor). For the off-line case, we present approximation algorithms. These algorithms may be useful in scenarios where advance reservations are allowed.

In Section 4 we give a detailed description of how the architecture and algorithms could be implemented in a real system. We discuss which information should be stored in tables at the edge routers, how the ingress and egress router communicate during call admission control, etc. In this section, we also discuss the DiffServ aspects of our architecture in more detail.

The setup of a high-level simulation of the call admission control architecture as well as various simulation results are discussed in Section 5. We compare the profits obtained with different algorithms: the greedy algorithm (that accepts any request for which enough bandwidth is available), the original version of the algorithm due to Awerbuch, Azar and Plotkin [2], and tuned versions of the latter algorithm. We find that the tuned versions can achieve significantly higher profit than the greedy algorithm and might therefore be a good choice for application in practice. Furthermore, we consider a variation of the problem in which the call admission control algorithm can propose a higher price to the user who submits a request. The user is assumed to decide randomly whether to accept the higher price according to some probability distribution. We present simulation results showing the effects of different user models on the achievable profit.

Advantages and disadvantages of the proposed scheme as well as some issues for future work are presented in Section 6.

In the remainder of the introduction, we discuss related work on practical and theoretical aspects of call admission control and on hose model bandwidth reservations.

## 1.1 Related Work on Call Admission Control

There is a lot of previous work on call admission control mechanisms with end-to-end QoS differentiation to a class of service or flow control classes (FCClasses) and we briefly look at their advantages and disadvantages.

**RSVP signaling** [9] – The sender sends the PATH message to the receiver with QoS and traffic characteristics of the flow, each router retransmitting it to the next hop and the receiver upon receiving the PATH message sends a RESV message requesting network resources for the flow from all routers along the path. Though this approach provides strong QoS support it has the disadvantage of being not scalable as routers along the path must maintain a flow state and bandwidth must be reserved on a per flow basis on each of the routers along the path.

**Bandwidth Broker** [30] – The bandwidth broker is responsible for both the call admission control and routing, thus it is one entity doing all functions for the entire domain. So it has to perform the tasks of path computation, admission control, maintaining network topology database, signaling, resource reservation etc. Also such architectures require per domain bandwidth broker

and thus per domain bandwidth broker communication. The bandwidth broker needs to manage the network and to store information about all elements, flows and paths in the network. This is very hard for one element. Therefore, for a large network a distributed mechanism is preferable.

**Dynamic Packet State** [29] – In dynamic packet state the flow state information like reserved rate, scheduling process, signaling etc is inserted in the packet header, thus avoiding per-flow signaling and state management. The ingress router initializes the state information and inserts this information into the packet header. Core routers process each incoming packet based on the state carried on it and update its internal state and the state of the packet's header before forwarding it. Admission control is done by RSVP where the RSVP messages are processed only by edge routers and core routers maintain no per flow information. This has the disadvantage that deterministic service is provided since admission control is based only on the flow state inserted in the packet header. This also reduces utilization.

**Aggregation in Integrated Services** [5] – This approach is used to reduce the number of signaling messages in an IntServ architecture. In this technique admission control is only performed on an aggregated set of flows and thus the core routers only need to maintain reservation state of each aggregate. The RSVP protocol is used but only for this aggregate. This has the problem of the balancing act where more aggregation means less utilization and less aggregation means signaling messages remain high and thus huge scaling problems.

**End Point Admission Control Through Probing** [10, 22] – Probes are sent out in the requested flow control class and the results gathered by the egress router in terms of packet loss, jitter, transmission time etc decides whether the flow can be accepted. This has the problem of bandwidth stealing, since the probes do not show the effect of accepting the new flow on other flow control classes. Also there is an overhead in the form of large setup time.

The reason for employing End Point Admission Control is that it combines DiffServ's scalability [23] with IntServ's superior QoS [9].

**Competitive on-line algorithms** – From the theory side, much of the previous work on CAC has focussed on the development of competitive on-line algorithms. The CAC algorithm learns about a call only at the time the call wants to be established, and the admission decision must be made immediately without knowledge about future requests. The profit achieved by the on-line algorithm is compared with the optimal profit that a clairvoyant algorithm with complete knowledge of the future and with unlimited computational power could have achieved; the resulting worst-case ratio is called the *competitive ratio*. Preemptive deterministic on-line algorithms for line networks were presented by Garay et al. [13]. They assume that all bandwidth requirements and all edge capacities are equal to 1, i.e., they study the maximum edge-disjoint paths problem. For the case that all calls have the same profit, they present an $O(\log n)$-competitive algorithm for line networks with $n$ nodes. Awerbuch et al. [3] consider tree networks and present a "Classify and randomly select" paradigm for adapting algorithms to the case of arbitrary durations, bandwidth requirements and profits so that the competitive ratio gets worse by at most a logarithmic factor. A randomized, non-preemptive $O(\log d)$-competitive algorithm for trees, where $d$ is the diameter of the tree, is presented in [4]. They also discuss meshes and trees of meshes. Recently, Adler

and Azar have obtained an $O(1)$-competitive algorithm for line networks by using randomization and preemption [1]. Awerbuch, Azar and Plotkin [2] present a non-preemptive on-line algorithm for call admission in arbitrary networks and with arbitrary bandwidth requirements. Their algorithm achieves competitive ratio $O(\log nT)$, where $n$ is the number of network nodes and $T$ is the maximum call duration, provided that each call has a bandwidth requirement that is at most a logarithmic fraction of the smallest edge capacity. Most of these algorithms assume global coordination and that complete knowledge about the current state of the network is available to the entity performing CAC. For surveys of on-line call admission and routing algorithms, see [20] and [8, Chapter 13].

**Off-line approximation algorithms** – Other work has dealt with approximation algorithms for the off-line scenario, where all requests are known in advance. Kleinberg and Tardos obtain an $O(1)$-approximation algorithm for the maximum edge-disjoint paths problem in two-dimensional meshes (and a class of similar networks) [18]. For more information about disjoint paths problems, see [17]. Using an approximation technique from [7] that is based on linear programming and a coloring subroutine, Phillips et al. obtain a 4-approximation algorithm for resource-constrained scheduling, a problem related to CAC on a single link [21]. Bar-Noy et al. present a unified approach to resource allocation and scheduling using the so-called *local ratio technique* [6].

## 1.2 Related Work on Bandwidth Allocation in the Hose Model

The hose model for bandwidth reservations has been proposed in the context of VPN (virtual private network) provisioning. A VPN is an emulation of services provided by Private Networks (PN) – leased lines connecting a set of PN sites. A VPN ensures predictable and secure network connections by dedicating part of the bandwidth of a shared network to connections between VPN endpoints. With an increasing number of VPN endpoints, an alternative to the standard point-to-point (or customer-pipe) approach has been proposed – the *hose model* [11]: Given a network with link capacities and a set of VPN endpoints, one needs to know an upper limit of the amount of traffic entering and leaving each endpoint from and to the other endpoints. Then, a VPN hose is created connecting all endpoints, with preference for sharing as many links as possible. The structure of the hose can be a tree, or any other connected graph. The hose model is based on traffic aggregation from each endpoint to the set of others. The obvious advantages of aggregation are: simpler specification (the detailed traffic matrix is not needed anymore), multiplexing gain, i.e. a reduction of the needed access link capacities, and flexibility in routing traffic from one point to anywhere else.

In [15] it was shown that for the symmetric case, when in each endpoint the in– and out–traffic limits are equal, there is a polynomial algorithm to build an optimal tree, provided that link capacities are not limited. Also they gave a $2-$approximation algorithm for the general hose graph case. For the asymmetric case, which is more realistic, the problem of finding an optimal tree becomes $\mathcal{NP}$-complete, and they gave a constant factor approximation algorithm.

In [19], three approximation algorithms are given for calculating a good hose tree in the asymmetric case. Experimental studies have shown that the relatively simple Breadth First Search algorithm performs well on large networks, while on small-size networks their Primal-Dual Algorithm outperforms Breadth First Search and Steiner tree algorithms.

The reliability of a VPN depends on the reliability of the shared network. Thus, failures of the dedicated links are a threat to the guaranteed service provisioning, the main characteristic of a VPN. In [16] a constant-factor approximation algorithm is given for an optimal restoration of the hose tree in the case of a single failure. The aim is to minimize the total bandwidth reserved on the backup edges.

# 2 CAC Architecture

We propose an architecture for realizing call admission control and quality of service guarantees in DiffServ networks. Recall that we assume that the network carries two types of traffic: ACT and BET. ACT consists of connections that request a certain bandwidth and must be guaranteed to get that bandwidth if they are established.

The basic idea is to reserve a certain fraction of the network bandwidth for ACT. This means that on every link of the network, a certain fraction of the bandwidth is assigned to ACT, and all routers ensure that ACT can indeed use the assigned fraction of the link bandwidth as long as there are ACT packets in the queue. BET can always use up the remaining bandwidth of the link.

Furthermore, in order to simplify bandwidth management, the bandwidth is reserved for ACT according to the hose model (proposed for VPN provisioning in [11]): every edge router $R$ is assigned limits $R^+$ and $R^-$ on the total rate of ACT that $R$ is allowed to send into and receive out of the DiffServ domain, respectively. At any point in time, let $T_R^+$ and $T_R^-$ denote the total rate of ACT entering the DS domain via $R$ and the total rate of ACT leaving the DS domain via $R$, respectively. Then a new connection with bandwidth requirement $x$ that enters the DS domain at $R$ and leaves it at $L$ can be admitted if and only if $T_R^+ + x \leq R^+$ and $T_L^- + x \leq L^-$. This condition can be checked locally by $L$ and $R$, i.e., only the edge routers where the connection enters and leaves the DS domain need to process the connection request.

Note that we assume for the sake of simplicity that each connection request specifies its bandwidth requirement as a single number $x$. In reality, when the transmission rate of flows is not constant over time, this number would be calculated using an effective bandwidth approach.

The hose model has been introduced as a model for reserving bandwidth for virtual private networks (VPNs). Our approach can thus be viewed as establishing a VPN in order to isolate ACT from BET.

## 2.1 Initial Configuration

The network provider determines for every edge router $R$ two values $R^+$ and $R^-$ according to the expected amount of ACT entering or leaving the network at $R$.

For every pair of edge routers $R$ and $L$, the provider determines a path $\pi_{R,L}$ from $R$ to $L$ in the network. This path is to be used by all ACT from $R$ to $L$.

On every link $(u, v)$ of the network, a sufficient amount $b_{u,v}$ of the bandwidth is assigned to ACT. This is done in such a way that any traffic pattern respecting the limits $R^+$ and $R^-$ for each edge router $R$ does not create more than $b_{u,v}$ traffic on $(u, v)$.

Every edge router stores its two values $R^+$ and $R^-$ and maintains the values $T_R^+$ and $T_R^-$, which are initially zero.

## 2.2 Admission Control

Assume that a customer submits a request for a connection that enters the network at $R$, leaves the network at $L$, and requires $x$ units of bandwidth. The following steps are carried out:

- The customer sends a reservation message to $R$.
- $R$ checks whether $T_R^+ + x \leq R^+$. If no, the request is rejected (and we stop).

- $R$ sends a message to $L$, and $L$ checks whether $T_L^- + x \leq L^-$. If no, the request is rejected (and we stop).
- There is sufficient bandwidth for the request. $L$ sets $T_L^- = T_L^- + x$ and informs $R$. $R$ sets $T_R^+ = T_R^+ + x$ and informs the customer that the request is accepted. (We will discuss in Section 2.4 how the scheme can be modified if acceptance depends on price-based criteria as well.)

Note that only the edge routers $R$ and $L$ have to process the connection request. Core routers are not involved in the admission control procedure.

When a connection from $R$ to $L$ with bandwidth $x$ is terminated, the edge routers $R$ and $L$ update their values by setting $T_R^+ = T_R^+ - x$ and $T_L^- = T_L^- - x$.

## 2.3 Bandwidth Guarantee

By construction of the scheme, it is clear that the amount of ACT on any link $(u, v)$ of the network cannot exceed the bandwidth $b_{u,v}$ assigned to ACT on that link. Therefore, it is ensured that all the accepted connection requests can indeed transmit at the rate that they have requested.

## 2.4 Price-Based Modifications

As defined above, the scheme is greedy in the sense that it accepts a connection whenever there is enough bandwidth. In order to take prices or profits into account, it is reasonable to compute a threshold $\theta$ for a connection request from $R$ to $L$ and to admit the connection only if its profit is at least $\theta$. The threshold should get larger as $T_R^+/R^+$ and $T_L^-/L^-$ get larger and depend on the requested bandwidth and duration as well. In this way, only higher-valued connection requests are accepted when the network gets congested.

## 2.5 Reconfiguration

If it turns out that the requested ACT is different from the expected amount (higher for some edge routers and smaller for others), the provider can reconfigure the limits $R^+$ and $R^-$ and adjust the bandwidths reserved for ACT accordingly. However, the provider should make sure that this happens only rarely, because the reconfiguration may involve updating all core routers and edge routers.

A more detailed explanation of the various aspects of the architecture from a systems point of view is given in Section 4.

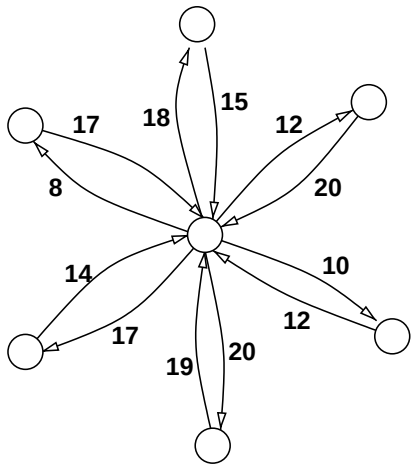


Figure 1: A network with star topology and edge capacities.

# 3 Online and Offline Approximation Algorithms

We are interested in admission control algorithms that provide provable worst-case guarantees concerning the total profit of the accepted calls. Ideally, the algorithms should be on-line (handle each call without knowledge about future calls) and *local* (only the ingress router and the egress router are involved in admission control of a call).

## 3.1 Equivalence to Networks with Star Topology

Once the bandwidth reservation for ACT has been carried out according to the hose model, the network is equivalent to a network with star topology (cf. Figure 1) for the purposes of admission control: the edge routers correspond to the leaves of the star, and the capacity of the edge from edge router $R$ to the center of the star is $R^+$ and the capacity of the reverse edge is $R^-$. Any set of connections that is feasible in the original network (with hose-model bandwidth reservation) is also feasible in the star, and vice versa. Therefore, we can assume from now on that call admission control is to be carried out in a star network. A call admission control algorithm for the star network can then be applied to the original DiffServ network without any modifications.

## 3.2 Preliminaries and Problem Definitions

The star network $G = (V, E)$ is given as a bidirected graph with the node $C$ in the center and with the set of leaves $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$. The capacities $B(e)$ of the edges $e$ are defined as $B(R_i, C) = R_i^+$ and $B(C, R_i) = R_i^-$ for all $i = 1, \ldots, n$.

A call $\sigma_i$ is given by sender $s_i \in \mathcal{R}$, receiver $r_i \in \mathcal{R}$, arrival time $a_i$, starting time $t_i$ (with $t_i \geq a_i$), duration $d_i \in \mathbb{N}$, bandwidth requirement $b_i$, and profit $p_i$. The input consists of a set $I = \{\sigma_1, \ldots, \sigma_m\}$ of $m$ calls. The (unique) path of $\sigma_i$ in the star is denoted by $path(\sigma_i)$. A call $\sigma_i$ is said to be *active* at time $t$ if $t_i \leq t < t_i + d_i$.

A subset $A \subseteq I$ is *feasible* if, for every time step $t$ and every leaf $R_j$, the sum of the $b_i$ values of all calls $\sigma_i$ in $A$ with $s_i = R_j$ that are active at time $t$ is at most $R_j^+$ and the sum of the $b_i$ values

of all calls $\sigma_i$ in $A$ with $r_i = R_j$ that are active at time $t$ is at most $R_j^-$. The profit of a feasible set $A$ is $p(A) = \sum_{\sigma_i \in A} p_i$. Our goal is to compute a feasible set $A$ of maximum profit.

Depending on when the algorithm is required to make the decision concerning admission of each call, we distinguish the following problem variants:

**Offline CAC:** Given a set of calls, compute a feasible set of maximum profit.

**Online CAC:** The calls arrive one by one. Each call is presented to the algorithm at its arrival time. The algorithm must decide to accept or reject each call immediately, without knowledge about future requests. (If several calls have the same arrival time, they are still presented to the algorithm one by one.) The goal is again to compute a feasible set of maximum profit.

For each of the problems, we also consider the special case where all edges have the same capacity. Then we call the problem *offline/online CAC with unit capacities*.

Furthermore, we consider also the special case of the problem where all calls have the same starting time and duration. This special case in mainly of theoretical interest, but it already illustrates the fundamental properties of the problem. We refer to this special case as BCAC (basic CAC).

Offline CAC with arbitrary profits is $\mathcal{NP}$-hard even if the star has only two leaves, all edges have the same capacity, and all calls have the same starting time and the same duration: This restricted version of the problem is equivalent to the KNAPSACK problem (the edge capacity corresponds to the size of the knapsack and the calls correspond to objects to be packed in the knapsack), which is $\mathcal{NP}$-hard [14]. Therefore, we are interested in efficient algorithms computing solutions that are provably close to the optimum.

For a given instance $I$, let $\mathrm{ALG}(I)$ denote the total profit of the solution computed by an algorithm and $\mathrm{OPT}(I)$ the optimal profit. The algorithm is called a $\rho$-approximation algorithm if $\mathrm{ALG}(I)$ is always at least $\mathrm{OPT}(I)/\rho$. In the online case, we also say that the algorithm is $\rho$-competitive.

For Online CAC in stars, we observe that we can employ the admission control algorithm by Awerbuch, Azar and Plotkin [2]. If there are values $F$ and $T$ such that the profit of every call $\sigma_i$ is in the interval $[b_i d_i, F b_i d_i]$ and the duration of every call is in the interval $[1, T]$, the algorithm achieves competitive ratio $O(\log(TF))$ provided that every call uses at most a $1/\log(2TF + 2)$-fraction of the bandwidth of any of its two edges.

For Offline CAC in stars, we allow arbitrary profit values, bandwidth requirements, and durations. We show that there is a constant factor approximation algorithm for unit capacities and an $O(\log \gamma)$-approximation algorithm for arbitrary capacities, where $\gamma$ is the ratio of the maximum capacity to the minimum capacity. For Offline BCAC, we get constant factor approximation algorithms even in the case of arbitrary capacities.

## 3.3 A Competitive Algorithm for Online CAC

Assume that the calls arrive in the order $\sigma_1, \ldots, \sigma_m$, i.e., we have $a_1 \leq a_2 \leq \cdots \leq a_m$. We assume that the profit of every call $\sigma_i$ is in the interval $[b_i d_i, F b_i d_i]$, where $F \geq 1$ is known to the admission control algorithm. Furthermore, we assume that the duration of every call is an integer in the interval $[1, T]$, where $T \geq 1$ is also known to the algorithm. We assume that time is slotted and all

starting times are integers as well. If the call $\sigma_i$ is accepted, it occupies bandwidth $b_i$ on the two edges of $path(\sigma_i)$ in time slots $t_i, t_i + 1, \ldots, t_i + d_i - 1$.

We employ the admission control algorithm due to Awerbuch, Azar and Plotkin. While this algorithm requires centralized control in general networks, we will see that it can be implemented locally (just involving ingress and egress router when a call request is processed) for star networks.

For the sake of completeness, we describe the algorithm and give an outline of its analysis. Let $\mu = 2TF + 2$. We require that, for every call $\sigma_i$, we have $b_i/B(e) \leq 1/\log \mu$ for all $e \in path(\sigma_i)$. Let $\lambda_e(\tau, k)$ be the normalized load on edge $e$ in time slot $\tau$ just before call $\sigma_k$ is processed. Here, the normalized load of an edge is the sum of the bandwidth requirements of the accepted calls using the edge, divided by the capacity of the edge. Let

$$c_e(\tau, k) = B(e) \left(\mu^{\lambda_e(\tau,k)} - 1\right).$$

Now the admission control rule is easy to specify: Request $\sigma_i$ is accepted if and only if the condition

$$\sum_{e \in path(\sigma_i)} \sum_{\tau=t_i}^{t_i+d_i-1} \frac{b_i}{B(e)} c_e(\tau, i) \leq \frac{1}{2} p_i. \tag{1}$$

holds.

**Lemma 1** *The algorithm always computes a feasible solution.*

**Proof.** See [2]. □

Let $\mathcal{A}$ be the set of indices of the calls accepted by the algorithm.

**Lemma 2** $4 \log \mu \sum_{i \in \mathcal{A}} p_i \geq \sum_\tau \sum_{e \in E} c_e(\tau, m + 1)$.

**Proof.** See [2]. □

Let $\mathcal{Q}$ be the set of indices of the calls accepted by the optimal solution, but not by the algorithm.

**Lemma 3** $\sum_{i \in \mathcal{Q}} p_i \leq 2 \sum_\tau \sum_{e \in E} c_e(\tau, m + 1)$.

**Proof.** See [2]. □

**Theorem 1** *The algorithm for Online CAC achieves competitive ratio $O(\log(TF))$ provided that for all $\sigma_i$, we have $b_i/B(e) \leq 1/\log(2TF + 2)$ for all $e \in path(\sigma_i)$.*

**Proof.** Follows from Lemmas 1–3. □

Note that we allow that the starting time of a call is later than its arrival time. Therefore, the algorithm works in a scenario with advance reservation without any alteration.

It also follows from the arguments in [2] that the competitive ratio $O(\log(TF))$ is optimal up to a constant factor. It can also be shown that $\Omega(T^{1/k} + F^{1/k})$ is a lower bound on the competitive ratio if $b_i/B(e)$ can be as large as $1/k$ for some $e \in path(\sigma_i)$. This means that the restriction on the bandwidth requirements cannot be avoided if logarithmic competitive ratio is to be achieved.

In order to implement the admission control algorithm, it suffices if each edge router stores the values $\lambda_e(\tau, i)$ for the two edges incident to it (in the star topology abstraction of the DS network). When call $\sigma_i$ is requested, the ingress router $s_i$ can evaluate the expression $\sum_{\tau=t_i}^{t_i+d_i-1} \frac{b_i}{B(e)} c_e(\tau, i)$ for $e = (s_i, C)$, and the egress router $r_i$ can evaluate it for $e = (C, r_i)$. The sum of the two resulting values can be used to check whether (1) is satisfied and the call is to be accepted or not. If $\sigma_i$ is accepted, only the two edge routers $s_i$ and $r_i$ need to update their stored values to reflect $\lambda_e(\tau, i+1)$.

In the remainder of this report, we refer to the algorithm by Awerbuch, Azar and Plotkin simply as the *AAP algorithm* for the sake of brevity.

## 3.4 Approximation Algorithms for Offline CAC

In this section, we present approximation algorithms for Offline CAC. We allow arbitrary profit values and arbitrary bandwidth requirements.

We give a general outline of our approach to approximating CAC. First, we use a linear programming relaxation in order to obtain a fractional solution to the problem. Then we create a new set of calls (containing a certain number of copies of every original call) and partition this set into a number of bins, where each bin corresponds to a (integral) feasible solution to the original problem. Finally, we output the solution with maximum profit among all feasible solutions obtained in this way.

Our approach is inspired by the work in [7], where a linear programming relaxation and a coloring subroutine are used to schedule jobs with release times and deadlines on multiple machines such that the profit obtained from jobs that finish before their deadlines is maximized.

Let $I$ be the set of indices of the given calls. We assume that every call is such that it could be established if there was no other traffic in the network, i.e., we assume that $b_i \leq B(e)$ for all $e \in \mathit{path}(\sigma_i)$ for all $\sigma_i$ with $i \in I$. Calls violating this condition can easily be detected and discarded.

### 3.4.1 Linear Programming Relaxation

Let $X = \max_{i \in I} t_i + d_i$ be the latest ending time of any call. Consider the following linear programming relaxation of CAC, which uses a variable $x_i$ for every call $\sigma_i$ that indicates whether $\sigma_i$ is accepted ($x_i = 1$) or rejected ($x_i = 0$):

$$(LP_1) \quad \max \sum_{i \in I} p_i x_i \tag{2}$$

$$\sum_{i \in I(e,t)} x_i b_i \leq B(e), \quad e \in E, 0 \leq t < X \tag{3}$$

$$0 \leq x_i \leq 1, \quad i \in I \tag{4}$$

where $I(e,t) = \{i \in I : e \in \mathit{path}(\sigma_i), t_i \leq t < t_i + d_i\}$ is the set of all calls that use edge $e$ at time $t$. Constraint (3) says that, for any edge $e$ and any time $t$, the sum of the bandwidth requirements of accepted calls that use edge $e$ at time $t$ is at most $B(e)$, ensuring that the capacity of $e$ is not exceeded. The objective function (2) simply states that the goal is to maximize the total profit of the accepted calls. If we required that $x_i \in \{0, 1\}$ for all $i \in I$, we would have an integer linear programming formulation of CAC (but nothing would be gained because solving an ILP is an $\mathcal{NP}$-hard problem). Instead, we relax the integrality constraint and allow the $x_i$ to take arbitrary values

between zero and one (constraint (4)). The resulting linear program can be solved optimally in polynomial time, yielding values $\hat{x}_i$ for all $i \in I$. The corresponding objective value $\hat{z} = \sum_{i \in I} p_i \hat{x}_i$ is an upper bound on the optimal integral solution. As the values $\hat{x}_i$ can be fractional, we have to show how an integral solution can be derived.

One obstacle in solving $LP_1$ is the potentially large number of constraints (3). However, we can omit all constraints (3) for time steps $t$ such that no call has starting time $t$. The remaining subset of constraints (3) is denoted (3)$'$. If a solution violates the original set of constraints (3) for edge $e$ at time $t$, it also violates a constraint (3)$'$ for edge $e$ and time $t'$, where $t'$ is the largest starting time $t_i$ of a call $\sigma_i$ with $t_i \leq t$. Therefore, we simply use the constraints (3)$'$ in place of (3). Then $LP_1$ has $|I|$ variables and $O(n|I|)$ constraints. So the size of $LP_1$ is polynomial in the size of the input and $LP_1$ can be solved efficiently in polynomial time (polynomial not only in the size of $LP_1$, but also in the size of the given instance of CAC) [26].

### 3.4.2 Discretizing the LP solution

As the next step, we round the values $\hat{x}_i$ down to values $\bar{x}_i$ that are integral multiples of $\frac{1}{N}$ for some integer $N$. This can be done without losing much in the objective function, i.e., for any given $\delta > 0$ we can achieve $\bar{z} = \sum_{i \in I} \bar{x}_i p_i \geq (\sum_{i \in I} \hat{x}_i p_i)/(1+\delta)$ by choosing $N$ sufficiently large (depending on $\delta$). For a fixed $\delta > 0$, the value of $N$ can be chosen polynomial in the size of the input.

Now consider the set $I'$ of calls that contains $N\bar{x}_i$ copies of every call $i \in I$. We refer to the copies of calls also as *call instances*. Note that $\sum_{i \in I'} p_i = \sum_{i \in I} N\bar{x}_i p_i = N\bar{z}$. Our goal is to partition $I'$ into $k \leq \alpha N$ feasible solutions $A_1$, $A_2$, ..., $A_k$, where we try to have $\alpha$ as small as possible. If we have achieved that, a pigeonhole argument shows that at least one of the solutions must have total profit at least $\frac{N\bar{z}}{k} \geq \frac{1}{\alpha}\bar{z}$. Let $A_j$ be the feasible solution that maximizes the total profit among $A_1,\ldots,A_k$. Our algorithm outputs $A_j$ and, consequently, achieves approximation ratio $\alpha(1+\delta)$.

In fact, it is possible to skip the rounding to multiples of $\frac{1}{N}$ and put the fractional call instances into $I'$ directly. $I'$ is then partitioned into fractions of feasible solutions (similar to fractional graph coloring [25]), but otherwise the algorithm remains essentially the same. The advantage is that we can avoid losing a factor of $1+\delta$ due to the rounding to multiples of $\frac{1}{N}$ in this way. Nevertheless, the approach described above (with the rounding) is conceptually simpler and so we use it in this paper. For the formulation of our theorems, we assume that the fractional approach (without rounding) is used so that we do not lose the factor of $1+\delta$ in the approximation ratio.

### 3.4.3 Solving the Partitioning Problem

We want to partition $I'$ into feasible (integral) solutions to the original instance of CAC. From the way in which $I'$ was obtained from a feasible (fractional) solution to $LP_1$, we know that $I'$ contains at most $N$ call instances belonging to the same call (from constraint (4)) and that $\sum_{i \in I'(e,t)} b_i \leq N \cdot B(e)$ for all $e \in E$ and all $t = 0, 1, \ldots, X$ (from constraint (3)). We refer to the value $\sum_{i \in I'(e,t)} b_i / B(e)$ as the load of edge $e$ at time $t$. Thus constraint (3) tells us that the maximum load of an edge due to call instances in $I'$ is at most $N$.

**Partitioning for BCAC** First, we consider the case that there is a time $t \in \mathbb{N}$ such that all call instances in $I'$ include time $t$, i.e., $t_i \leq t < t_i + d_i$ for all $i \in I'$. (In particular, this is the case for

1. Maintain feasible solutions $A_1$, $A_2$, ...; initially, all $A_j$ are empty.
2. Consider the instances in group 1 in arbitrary order, and then the instances in group 2 in arbitrary order. Add each instance $i$ to the solution $A_j$ such that $A_j$ remains feasible and $j$ is minimized among all such solutions $A_j$.

Figure 2: Algorithm for groups 1 and 2.

the problem BCAC.)

We classify the call instances in $I'$ into the following groups. Group 1 contains all instances $i$ such that $b_i > B(e)/2$ for all $e \in \mathit{path}(i)$. Group 2 contains all instances $i$ such that $b_i \leq B(e)/2$ for all $e \in \mathit{path}(i)$. Group 3 contains all instances $i$ using two edges $e_1$ and $e_2$ with $B(e_1) > B(e_2)$ and satisfying $b_i \leq B(e_1)/2$ and $b_i > B(e_2)/2$.

The instances in groups 1 and 2 are treated separately from the instances in group 3. For instances in groups 1 and 2, the partitioning algorithm shown in Figure 2 is used.

Let $k_1$ be the number of non-empty solutions $A_j$ after the execution of this algorithm.

**Lemma 4** $k_1 \leq 5N$.

**Proof.** First, consider an instance $i$ from group 1 and assume that $i$ is put in solution $A_j$ by the algorithm. This means that in every solution $A_h$, $1 \leq h \leq j-1$, either an edge used by $i$ was blocked by a previous call instance, or $A_h$ contained another instance belonging to the same call. There are at most $N-1$ previous instances belonging to the same call, so at most $N-1$ solutions can be blocked for this reason. Furthermore, all previous call instances are also from group 1, so they occupy at least half the edge capacity. As the load of an edge $e$ is at most $N$, each edge can be blocked in at most $2N$ solutions. As $i$ uses two edges, at most $N-1+2N+2N$ solutions are blocked, and we have $j \leq 5N$

Now consider an instance $i$ from group 2 and assume that $i$ is put in solution $A_j$ by the algorithm. Instance $i$ is blocked from a solution $A_h$ only if $A_h$ contains another call instance from the same call or if an edge $e \in \mathit{path}(i)$ has less than $b_i$ bandwidth available in $A_h$; in the latter case, the bandwidth used on $e$ in $A_h$ is at least $B(e)/2$. So the same reasoning as above shows that $j \leq 5N$. Therefore, no instance is assigned to a solution $A_j$ with $j > 5N$. □

Now consider the call instances in group 3. Use the algorithm shown in Figure 3 to partition them into feasible solutions. We remark that the order in which this algorithm processes the call instances in group 3 is essential: if they were processed in arbitrary order, no constant approximation ratio could be achieved.

Let $k_2$ be the number of non-empty solutions $B_j$ after the execution of this algorithm.

**Lemma 5** $k_2 \leq 5N$.

**Proof.** Consider an instance $i$ from group 3 and assume that $i$ is put in solution $B_j$ by the algorithm. Let $e_q$ and $e_r$ be the edges used by $i$, with $q < r$. By the definition of group 3, $b_i \leq B(e_q)/2$ and $b_i > B(e_r)/2$. At most $N-1$ solutions $B_h$ are blocked by other instances of the same call. At most $2N$ solutions are blocked because less than $b_i$ bandwidth is available on $e_q$ (as the bandwidth occupied on $e_q$ is at least $B(e_q) - b_i \geq B(e_q)/2$ in all these solutions). At most $2N$ solutions are blocked because less than $b_i$ bandwidth is available on $e_r$: this is because every

1. Maintain feasible solutions $B_1$, $B_2$, ...; initially, all $B_j$ are empty.
2. Let $e_1, e_2, \ldots, e_n$ be the edges of the star in order of non-increasing capacities, i.e., $B(e_1) \geq B(e_2) \geq \cdots \geq B(e_n)$.
3. For $q = 1, 2, \ldots, n$, do the following: consider the call instances $i$ that use $e_q$ and no edge with smaller index than $e_q$, in arbitrary order, and add each such instance $i$ to the solution $B_j$ such that $B_j$ remains feasible and $j$ is minimized among all such solutions $B_j$.

Figure 3: Algorithm for group 3.

1. Maintain feasible solutions $A_1$, $A_2$, ...; initially, all $A_j$ are empty.
2. Consider the instances in $I'$ in the order of non-decreasing starting times. Add each instance $i$ to the solution $A_j$ such that $A_j$ remains feasible and $j$ is minimized among all such solutions $A_j$.

Figure 4: Partitioning algorithm for general CAC.

previously processed call instance using $e_r$ occupies more than half the capacity of $e_r$; so whenever a solution is blocked on $e_r$, more than $B(e_r)/2$ of the capacity of $e_r$ is occupied in that solution. Summing up, we obtain that $j \leq 5N$. No instance is added to a solution with index greater than $5N$, so we have $k_2 \leq 5N$. □

So we have partitioned $I'$ into at most $10N$ feasible solutions. At least one of these feasible solutions must have total profit at least $\bar{z}/10$. (Otherwise, the total profit of all solutions would be smaller than $N\bar{z}$, a contradiction.) The feasible solution produced by our algorithm has a total profit that is at least a 1/10-fraction of the optimal profit.

In the case of BCAC with unit edge capacities, there are no calls in group 3, and so we can partition $I'$ into at most $5N$ feasible solutions.

Summing up the preceding discussion, we obtain the following theorem.

**Theorem 2** *For Offline BCAC in star networks, there is a* 10*-approximation algorithm for arbitrary edge capacities and a* 5*-approximation algorithm for unit edge capacities.*

**Partitioning for general CAC** Now we consider the general case where the call instances in $I'$ can have arbitrary starting times and durations. We propose the greedy algorithm shown in Figure 4 for partitioning $I'$ into feasible solutions. Let $k$ be the number of non-empty solutions at the end of the algorithm.

Let $B_{\min} = \min_{e \in E} B(e)$ and $B_{\max} = \max_{e \in E} B(e)$ denote the minimum and maximum edge capacity in the network, respectively, and let $\gamma = B_{\max}/B_{\min}$ be the ratio of the maximum edge capacity to the minimum edge capacity.

We say that a call instance $i$ using edges $e_1$ and $e_2$ with $B(e_1) \geq B(e_2)$ is *big* if $b_i > B(e_1)/3$ and $b_i > B(e_2)/3$, is *small* if $b_i \leq B(e_1)/3$ and $b_i \leq B(e_2)/2$, and is *mixed* if $b_i \leq B(e_1)/3$ and $b_i > B(e_2)/2$,

**Lemma 6** $k = O(N \log \gamma)$.

**Proof.** Consider an instance $i \in I'$ and assume that $path(i)$ consists of two edges $e_1$ and $e_2$, with $B(e_1) \geq B(e_2)$. Assume that $i$ is added to $A_j$ by the algorithm. This means that for $A_h$, $1 \leq h \leq j-1$, at least one of the following conditions holds:

1. $A_h$ contains another instance of the same call as $i$.
2. Edge $e_1$ has less than $b_i$ bandwidth available at time $t_i$ in $A_h$.
3. Edge $e_2$ has less than $b_i$ bandwidth available at time $t_i$ in $A_h$.

Note that if enough bandwidth is available on an edge at time $t_i$, then it is also available in all later time steps, because the call instances are processed in order of non-decreasing starting times.

We claim that $j \leq 4.5N$ if $i$ is a small call, $j \leq 5.5N(1+\log_{1.5}(B(e_1)/B_{\min}))$ if $i$ is a mixed call, and $j \leq 6N + 5.5N(1+\log_{1.5}(B_{\max}/B_{\min}))$ if $i$ is a big call. The claim is proved by induction on the number of call instances processed by the partitioning algorithm. Distinguish the following cases:

*Case 1:* $i$ is small, i.e., $b_i \leq B(e_1)/3$ and $b_i \leq B(e_2)/2$. At most $N-1$ solutions are blocked by different instances of the same call. At most $1.5N$ solutions are blocked because $e_1$ does not have enough bandwidth available, and at most $2N$ solutions are blocked because $e_2$ does not have enough bandwidth available. Therefore, $j \leq 4.5N$, and the claim holds for call instance $i$.

*Case 2:* $i$ is mixed, i.e., $b_i \leq B(e_1)/3$ and $b_i > B(e_2)/2$. Note that $B(e_2) \leq 2B(e_1)/3$. At most $N-1$ solutions are blocked by different instances of the same call. At most $1.5N$ solutions are blocked because $e_1$ does not have enough bandwidth available. Let $h < j$ be the maximum index of a set $A_h$, $1 \leq h < j$, in which $e_2$ is used by a call instance $i'$ with $b_{i'} \leq B(e_2)/3$. Either no such $h$ exists (and we have $j \leq 5.5N$), or $h \geq j - 5.5N$.

Let $e_2$ and $e_3$ be the edges used by $i'$. If $b_{i'} \leq B(e_3)/2$, we can conclude that $h \leq 4.5N$. Thus $j \leq h + 5.5N \leq 2 \cdot 5.5N \leq 5.5N(1+\log_{1.5}(B(e_1)/B_{\min})$, where the last inequality holds because $B(e_1) \geq 1.5B(e_2) \geq 1.5B_{\min}$.

Now assume that $b_{i'} > B(e_3)/2$. This implies $B(e_3) < B(e_2)$. As $i'$ is mixed, we have by the inductive hypothesis that $h \leq 5.5N(1+\log_{1.5}(B(e_2)/B_{\min}))$. As $j \leq h + 5.5N$, we obtain $j \leq 5.5N(1+\log_{1.5}(1.5B(e_2)/B_{\min})) \leq 5.5N(1+\log_{1.5}(B(e_1)/B_{\min})$.

The claim holds for call instance $i$.

*Case 3:* $i$ is big, i.e., $b_i > B(e_1)/3$ (and thus $b_i > B(e_2)/3$). Let $h < j$ be the maximum index of a set $A_h$, $1 \leq h < j$, in which $e_1$ or $e_2$ is used by a call instance $i'$ with $b_{i'} \leq B(e_1)/3$ or $b_{i'} \leq B(e_2)/3$, respectively. We find that either no such $h$ exists (and we get $j \leq 6N$) or that $h \geq j - 6N$. In the latter case, we observe that $i'$ is a small call or a mixed call and that $h \leq 5.5N(1+\log_{1.5}(B_{\max}/B_{\min}))$ by the inductive hypothesis. As $j \leq h + 6N$, the claim holds for the big call instance $i$.

So we obtain that $k \leq 6N + 5.5N(1+\log_{1.5}(B_{\max}/B_{\min})) = O(N \log \gamma)$ □

At least one of the solutions $A_1,\ldots,A_k$ must have total profit at least $\bar{z}/O(\log\gamma)$. Thus we obtain approximation ratio $O(\log\gamma)$ for the case of arbitrary edge capacities.

In the case of unit edge capacities ($B(e) = 1$ for all $e \in E$), there are no mixed calls. A call instance $i$ with $b_i \leq \frac{1}{2}$ is blocked in at most $N-1$ solutions by other instances of the same call and in at most $4N$ solutions because an edge in $path(i)$ does not have enough bandwidth available at time $t_i$. Hence, such a call instance can be put in one of $5N$ solutions. A call instance $i$ with $b_i > \frac{1}{2}$ can be blocked in at most $4N$ solutions by other call instances with bandwidth requirement

greater than $\frac{1}{2}$, hence it can be put in one of $9N$ solutions. Thus we can partition $I'$ into at most $9N$ feasible solutions in the case of unit edge capacities.

**Theorem 3** *For Offline CAC in star networks, there is a* 9*-approximation algorithm for unit edge capacities and an* $O(\log \gamma)$*-approximation algorithm for arbitrary edge capacities, where* $\gamma$ *is the ratio of the maximum edge capacity to the minimum edge capacity.*

# 4 System Functionality and Design

From a high level perspective, the system offers a 'VPN' like service, for an aggregated flow with bandwidth guarantees into the DiffServ domain which can be viewed as a virtual link (from the edge router). Thus bandwidth reservations are not made on each link by the edge routers, but the aggregate of all outgoing links is viewed as a virtual link on which ingress and egress reservations are made. The aggregated flow is assumed to carry DiffServ traffic [23]. For each Flow Control Class (FCClass), a minimum bandwidth is guaranteed. Optionally, a maximum bandwidth guarantee can also be guaranteed. At the edge of the network, the ingress and egress routers carry out policing. In the core of the DiffServ domain, no policing is done.

The assumed model looks like in Figure 5, where the network cloud is a DiffServ Domain with all routers in the domain being DiffServ aware and a Bandwidth Broker responsible for the DiffServ Domain. The job of the Bandwidth Broker is to assign (initialize) the ingress and egress bandwidth values to each of the edge routers as in a VPN Hose and to reserve bandwidth on each of the links within the domain depending on the edge (ingress and egress) reservations. The Bandwidth Broker which has the global view of the domain, sends these reservation values to each of the routers in the domain during initialization.

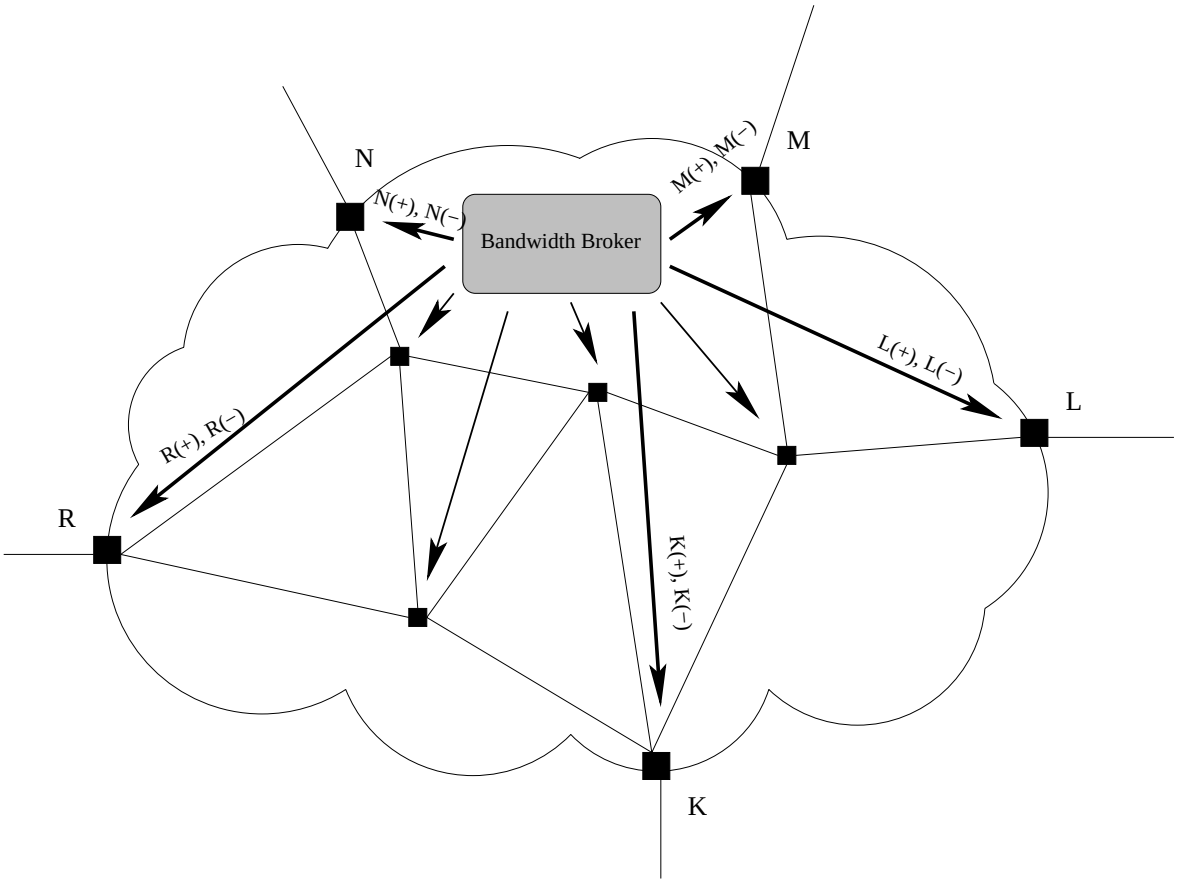


Figure 5: High Level View

## 4.1 Bandwidth Reservation

Before any traffic is let into the system, resources are pre-allocated by the ingress and egress routers for ACT in the domain. This pre-allocation specifies the amount of bandwidth of the virtual link allocated for ACT and the remaining bandwidth is allocated for BET. Once this decision is made a VPN hose is set up between the ingress and egress routers of the transmission path (by bandwidth reservations on each link by core routers as initialized by the Bandwidth Broker). Admission control is then done for the traffic passing through the VPN hose, which demands some level of QoS and traffic differentiation. The pre-allocation might be done like in the Figure 6 below, where the distribution defines the ingress and egress bandwidth for that VPN hose. Also the VPN capacity, allocated for each FCClass in the VPN hose, is pre-allocated as shown in Figure 6. These pre-allocations described above are an administrative (Bandwidth Broker) and management decision and require no special call admission control scheme. The VPN hose thus is a static entity which is set up once at each edge router end-to-end and the dynamic entity in this case would be the ACT requested by the hosts connected to these edge routers.

The call admission control algorithm is applied to each requesting flow in the admission controlled VPN hose. Each such flow conforming to the PCAC Algorithm and the VPN hose's pre-allocated reservation for each FCClass is admitted with the QoS level requested.

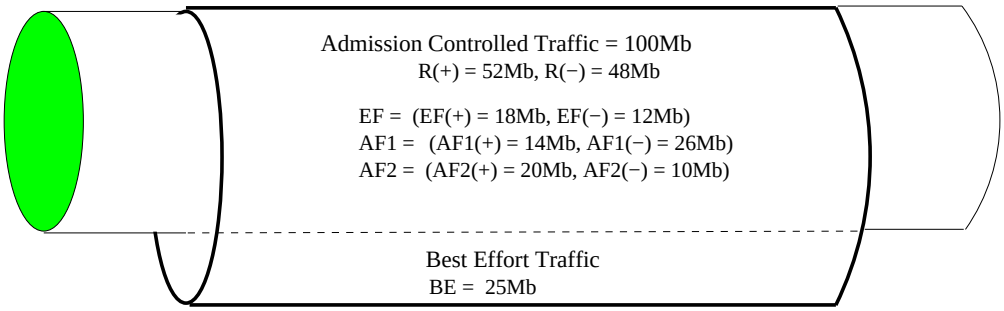


Figure 6: Example of a Virtual Private Network Bandwidth Mapping

## 4.2 VPN Hose Architecture

The basic idea is to reserve a certain fraction of the network bandwidth for ACT. This means that on every link of the network, a certain fraction of the bandwidth is assigned to ACT, and all routers ensure that ACT can indeed use the assigned fraction of the link bandwidth as long as there are ACT packets in the queue. BET can always use up the remaining bandwidth of the link.

### 4.2.1 Initial Configuration

The initial configuration is done as described in Section 2.1. The reservations during initialization are done by each of the edge and core routers as initialized by the Bandwidth Broker at initialization time as shown in Figure 5.

Every edge router stores its two values $R^+$ and $R^-$ and maintains the current ingress and egress rates $T_R^+$ and $T_R^-$ respectively, which are initially zero.

| $R^+$ |
|---|
| $R^-$ |
| $T_R^+$ |
| $T_R^-$ |
| Link Capacity |

Table 1: VPN HOSE - Allocation Table

Every edge router stores the allowed ingress and egress rates through the VPN hose for each of the FCClasses, and their current used bandwidths and available bandwidths.

### 4.2.2 Edge Router Design

The edge router is involved in the setup and maintaining of the VPN hose. The edge router is also responsible for the call admission control of the DiffServ traffic in the VPN. Thus, the information gathered and maintained by the edge router is large.

The bandwidth reserved for Admission Controlled Traffic in the VPN hose at every edge router $R$ is assigned limits $R^+$ and $R^-$ on the total rate of admission controlled traffic that $R$ is allowed to send into and receive from the DiffServ domain, respectively. At any point in time, $T_R^+$ and $T_R^-$ denote the total rate of admission controlled traffic entering the DS domain via $R$ and the total rate of admission controlled traffic leaving the DS domain via $R$, respectively. Then a new connection with bandwidth requirement $x$ that enters the DS domain at $R$ and leaves it at $L$ can be admitted if and only if $T_R^+ + x \leq R^+$ and $T_L^- + x \leq L^-$ holds. This condition can be checked locally by $L$ and $R$, i.e., only edge routers where the connection enters and leaves the DS domain need to process the connection request.

Therefore, the 'State Table (VPN HOSE - Allocation Table)' shown in Table 1 that needs to be maintained at each of the edge routers is as below. $R^+$ and $R^-$are pre-allocated and specified before the signaling and setup of the VPN hose. ACT reservation distribution is made thereafter, which is an administrative decision of the provider. $T_R^+$ and $T_R^-$ are initially zero and change after each connection is admitted or torn down. This Connection Admission is done by the PCAC Algorithm and thus PCAC is responsible for update and maintenance of the 'VPN HOSE - Allocation Table'.

Also since the entire network is DiffServ-aware edge routers have the additional task of Traffic Classification and Conditioning. The differentiated services architecture is based on a model where 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 DiffServ Code Point (DSCP). Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DSCP. The key components within a differentiated services region are traffic classification and conditioning functions, and how differentiated services are achieved through the combination of traffic conditioning and PHB-based forwarding.

The ACT in the VPN will be packets that are belonging to different Flow Control Classes (FCClass) as in DiffServ. These packets of different DiffServ classes are distinguished by their DSCP markings. The classifiers and traffic conditioners on the DS boundary nodes perform the classification and conditioning functions adhering to the Service Level Agreement and the Traffic Conditioning Agreement. Thus, packets entering the DS domain from the ingress node (router)

| FCClasses | b/w per FCClass (Mb) ($z_R$) | b/w in use (Mb) ($z_{R_{use}}$) | Available b/w (Mb) ($z_{R_{free}}$) |
|---|---|---|---|
| **EF** | **30** | **10** | **20** |
| $EF^+$ | 18 | 6 | 12 |
| $EF^-$ | 12 | 4 | 8 |
| **AF 1** | **40** | **15** | **25** |
| $AF1^+$ | 14 | 5.25 | 8.75 |
| $AF1^-$ | 26 | 9.75 | 16.25 |
| **AF 2** | **30** | **25** | **5** |
| $AF2^+$ | 20 | 16.25 | 3.25 |
| $AF2^-$ | 10 | 8.75 | 1.75 |
| **Best Effort** | - | - | - |

Table 2: Bandwidth Allocation Table

| Flow Number | FCClass | Bandwidth Guaranteed ($b_{z_R}$) | Duration (Minutes) |
|---|---|---|---|
| 1 | EF | 17 | 1440 |
| 2 | AF1 | 20 | 720 |
| 3 | AF2 | 8 | 360 |
| .. | ..... | .. | ... |

Table 3: Call Admission Control Table

undergo classification and conditioning. Traffic conditioning includes metering, marking, and shaping/dropping. Core routers in the DS domain may also perform these conditioning operations, though it is rarely the case. Within the core of the network, packets are forwarded according to the per-hop behavior associated with the DSCP. It must however be noted that in our initial model we do not consider this model and perform some high level simulations with call admission control algorithm defined later.

Each edge router thus has a 'Bandwidth Allocation Table'. A typical example of such a table is as in the Table 2. In the 'Bandwidth Allocation Table' we need to specify limits of ingress and egress traffic allowed for each of the FCClasses. Thus, each FCClass flow is a virtual VPN hose in effect and admission control can be done by just comparing any new request with these FCClass limits instead of the $R^+$ and $R^-$ values.

## 4.3 Core Router Functionality

Core routers are not involved in any call admission control decision. Core routers being DiffServ-aware are only involved in DiffServ Per-Hop behavior forwarding of all ACT packets within the VPN hose, which has a reserved bandwidth of $b(u, v)$. This reservation is carried out while the VPN hose is setup after receiving reservations from the Bandwidth Broker and while resizing is done and involves minimum work from the core routers. Core routers do not allow the amount of

ACT to exceed the reserved bandwidth $b(u, v)$, which is achieved by suitable remarking, shaping and/or policing. The hose setup requires in our case not only the ingress and egress rates $R^+$ and $R^-$ respectively from the edge router $R$, based on which the $b(u, v)$ reservations are done in the core path of the domain, but also the ingress and egress rates per FCClass need to be provided to each core router for DiffServ PHB forwarding. If needed remarking, shaping and/or policing may be added.

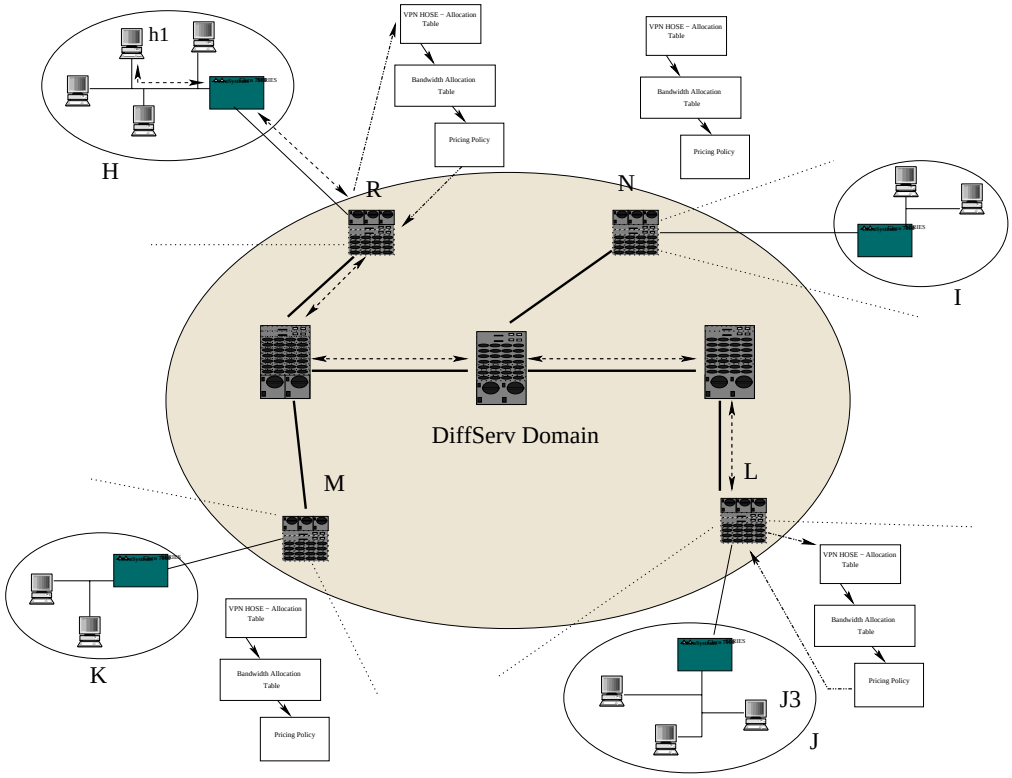


Figure 7: Virtual Private Network Scenario

## 4.4 Admission Control Scenario

In the architecture being assumed for the PCAC Algorithm, we assume a static VPN hose set up between the edge routers sending and receiving admission controlled traffic. Though these hoses can be resized dynamically and statistical multiplexing gains can be made by aggregating traffic in one direction, these issues are beyond the scope of present discussion, in order to gain further utilization by resizing and explicit routing see [19].

### 4.4.1 Greedy - Algorithm Description

In the Initialization Phase the 'VPN Hose - Allocation Table' is initialized with the hose ingress ($R^+$) and egress ($R^-$) limits. These limits are basically the initialized limits which can be changed throughout to resize the VPN hose in order to make maximum utilization of the available bandwidth. This resizing approach is outside the scope of our investigation. Initialize the ACT currently

| FCClass | Profit Factor | $t_{min}$ (Minutes) | Bandwidth |
|---|---|---|---|
| EF | 15 | 4 | 25 Kb |
| AF1 | 1 | 90 | 2 Mb |
| AF2 | 10 | 100 | 128 Kb |
| BE | $a$ | | |

Table 4: Pricing policy Table

entering and leaving the router $R$ through its VPN hose ($T_R^+$ and $T_R^-$), these are zero initially. Specify the bandwidth for ACT and BET entering and leaving the router. Setup the VPN hose from and to routers initialized with the above. Initialize the Bandwidth Allocation Table specifying the breakup of the allocated bandwidth for each of the FCClasses.

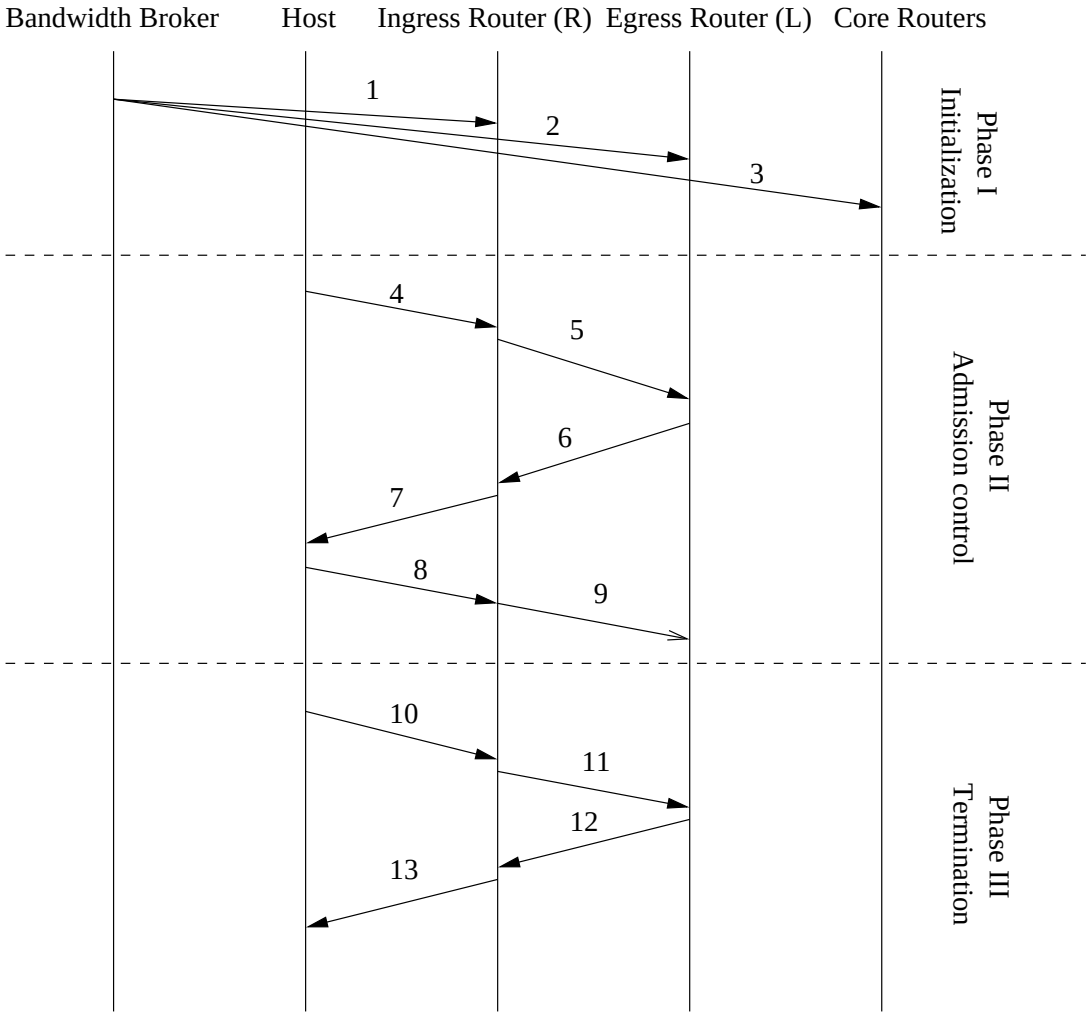


Figure 8: Message Sequence

The customer $h_i$ who is an element of all possible customers $H_R$ of edge router $R$ submits a

request for a connection that enters the network at $R$, and leaves the network at $L$, and requires a bandwidth of $b_{z_R}$ units, of FCClass $z_R$ which is an element of all possible FCClasses $Z_R$ offered by $R$, for a duration $d$ and offers a price $p$ for the connection. The edge router $R$ performs the VPN hose capacity and Bandwidth Allocation Table checks to confirm whether the request from the customer can be granted. If checks pass then a request for the same is sent to the edge egress router $L$ at the other end, which performs similar checks as $R$ and if granted, sends a confirmation. Else it sends a response offer $O_R$ to $R$ for the customer giving the possible FCClass and available bandwidth closest to what the customer requested. If the test passes at $L$ then ingress router $R$ sends the confirmation to the customer with the pricing $p$ offered by the customer. The customer can then within a time limit confirm and then the ingress and egress routers update their VPN Hose - Allocation table (Table 1), Bandwidth Allocation Table (Table 2) and Call Admission Control Table (Table 3). Alternatively if the request cannot be granted by either $R$ or $L$ a Response Offer $O_R$ which is the minimum of the response offers of $R$ and $L$ is sent to the customer $h_i$, who can then start a connection request based on the Response Offer made and admission control process starts again. Thus, only edge routers $R$ and $L$ are involved in processing the connection requests. Core routers are not involved in the admission control procedure. And the admissions are made based on the available bandwidth by the edge routers in a greedy manner.

The pricing policy can alternatively be decided just before the host is informed of the availability of requested bandwidth. This pricing policy is a management decision and is based on maximum profit for the ISP. We consider a static Pricing policy for the Greedy PCAC Algorithm where $Profit = ProfitFactor * Duration * Bandwidth$ and depending on the traffic class a Profit factor is defined. Since the profit is static policy we can consider this to be defined by the customer as part of his request, alternatively the ingress router can do this as the policy is the same and static.

Termination of a connection will again require only the ingress and egress routers $R$ and $L$ respectively to update the VPN Hose - Allocation table and Call and Admission Control Table. Also the resizing of the VPN Hose can be done at this point depending on the incoming requests and utilization. Figure 7 specifies the PCAC algorithm in detail. The dashed lines are the signaling between the ingress and egress routers where each such router has the Tables as shown which are used and updated in the PCAC algorithm.

Figure 8 above shows the three phases, Initialization, Admission control and Termination of the Greedy PCAC algorithm with message sequence. In the Initialization phase messages 1, 2 and 3 are the initializations of the edge and core routers by the Bandwidth Broker, thus setting up the VPN hose. In Admission control phase, message 4 is the request of a specific bandwidth in a FCClass by a host connected to a edge router $R$ with price ready to pay $p$. The ingress edge router $R$ checks available bandwidth in the FCClass as specified in the PCAC algorithm and sends the request to the egress router $L$, via message 5. The egress router performs similar checks and sends its acceptance or response offer decision to the ingress router by message 6. The ingress router then sends its acceptance offer with pricing policy to the host or the response offer it deems fit, if it is unable to provide the requested bandwidth in the FCClass. The host then either confirms its acceptance decision within a time period or starts with a new request considering the received response offer, by message 8 and if it is a confirmation, the ingress router updates its tables and sends confirmation to egress router by message 9 to update its tables as per the PCAC algorithm. In the Termination phase the host requests a termination of the connection it holds to the ingress router via message 10, the ingress router sends this termination signal to egress router by message

11, which responds with acknowledgment via message 12. The ingress router then confirms this termination to the host via an acknowledgment message 13.

### 4.4.2 Greedy PCAC Algorithm

The following describes the PCAC Algorithm :

**For Every Edge Router R in the DiffServ Domain.**

**Initialization**

1: **Initialize the VPN Hose - Allocation Table**
2: *Initialize the VPN Hose limits* $R^+ \leftarrow R_i^+$ *and* $R^- \leftarrow R_i^-$.
3: *Initialize the total rate of ACT entering and leaving the DiffServ domain via* **R**. $T_R^+ \leftarrow 0$ *and* $T_R^- \leftarrow 0$ *respectively.*
4: *Initialize the bandwidth reserved for Admission Controlled Traffic.*
5: *Initialize the bandwidth reserved for Best Effort Traffic.*
6: *Setup the* **VPN Hose** *from the router with the initialized Hose limits.*
7: **Initialize the Bandwidth Allocation Table**
8: *Specify the bandwidth allocation for each of the FCClasses in both the rate entering and leaving the DiffServ domain via* **R**.
9: **Initialize the Pricing Policy Table**

**Admission Control**

1: *Request from a host* $h_i \in H_R$ *for admission of Flow Control Class* $z_R \in Z_R$, *for duration* $d$ *and bandwidth* $b_{z_R}$ *through egress router* $L$, *and optionally price ready to pay* $p$.
2: *Router performs VPN hose capacity checks from the* **VPN Hose - Allocation Table** *for* $R^+$ *and from the* **Bandwidth Allocation Table** *for ingress available bandwidth per FCClass* $(z_{R^+{}_{free}})$.
3: ***if*** $(b_{z_R} \leq z_{R^+{}_{free}})$ ***then***
4: *Send the request message to Egress Router (L) at other end*
5: ***else***
6: *The Request from* $h_i$ *is* **rejected** *and a* **Response Offer** $O_R(h_i)$ *is made.*
7: *Terminate after sending the* **Response Offer** $O_R(h_i)$ *to host* $h_i$
8: ***end if***
9: *Router L performs VPN hose capacity checks from the* **VPN Hose - Allocation Table** *for* $L^-$ *and from the* **Bandwidth Allocation Table** *for egress available bandwidth per FCClass* $(z_{L^-{}_{free}})$.
10: ***if*** $(b_{z_R} \leq z_{L^-{}_{free}})$ ***then***
11: *sends* **Accept** *to* $R$
12: ***else***
13: *sends* **Reject** *to* $R$ *with its* **Response Offer** $O_L(h_i)$.
14: ***end if***
15: ***if*** *receives* **Reject** *response from the Egress Router* ***then***

16: *sends the lower of the two* **Response Offers** ($O_R(h_i)$ **and** $O_L(h_i)$) *to host* $h_i$.
17: ***else***
18: *Checks the* **Pricing Policy** *for the requested bandwidth* ($b_{z_R}$) *in that FCClass* ($z_R$)*and sends the appropriate* **Pricing** *and* **Admission Decision** *to the host* $h_i$.
19: ***end if***
20: ***if*** *host* $h_i$ *accepts the* **Pricing** *and* **Admission Decision** *and informs router R of confirmation within a period of time* ***then***
21: *Updates* **VPN Hose - Allocation Table** *and sets* $T_R^+ = T_R^+ + b_{z_R}$.
22: *Updates the* **Ingress Call Admission Control Table** *with*

- *Flow number*
- *FCClass of the bandwidth requested* ($z_R$)
- *Bandwidth guaranteed for that flow* ($b_{z_R}$)
- *Duration requested (in minutes)*

23: *Informs the egress router L which updates its* **VPN Hose - Allocation Table** *and sets* $T_L^- = T_L^- + b_{z_R}$.
24: *Updates the* **Egress Admission Control Table** *with*

- *Flow number*
- *FCClass of the bandwidth requested* ($z_L$)
- *Bandwidth guaranteed for that flow* ($b_{z_R}$)
- *Duration requested (in minutes)*

25: ***else***
26: *Host* $h_i$ *requests new bandwidth after receiving the Response Offer, Admission Control starts again.*
27: ***end if***

**Response Offer**

1: *Checks* **Bandwidth Allocation Table** *for available bandwidth.*
2: ***if*** $z_{R^+{}_{free}} \geq 0$ ***then***
3: *Store* ($z_{R^+{}_{free}}$) *in Response Offer* $O_R(h_i)$ *for host* $h_i$.
4: ***end if***
5: ***if*** *Other FCClasses* ($z_{R^+{}_{free}} \leq b_{z_R}$ ) ***then***
6: *Store the FCClasses with available bandwidths for all other FCClasses with less than or equal to the requested bandwidth i.e.* ($z_{R^+{}_{free}} \leq b_{z_R}$). [1]
7: ***end if***
8: *Return* **Response Offer** $O_R(h_i)$

[1] Specifying the bandwidths for these FCClasses in value not more than that requested.

**Termination**

1: *Request from a host $h_i$ to terminate connection from $R$ to $L$ with bandwidth $b_{z_R}$.*
2: *Updates* **VPN Hose - Allocation Table** *and sets $T_R^+ = T_R^+ - b_{z_R}$.*
3: *Updates* **Ingress Call Admission Control Table** *by recomputing and setting the bandwidth in use and available bandwidth.*
4: *Sends the flow number and termination signal to the egress router $L$ which updates its* **VPN Hose - Allocation Table** *and sets $T_L^- = T_L^- - b_{z_R}$*
5: *Egress router $L$ updates the* **Egress Call Admission Control Table** *by recomputing and setting the bandwidth in use and available bandwidth.*
6: *host $h_i$ is informed of the Termination Confirmation.*

## 4.5 The PCAC Algorithm by Awerbuch, Azar and Plotkin

Message sequence for the AAP PCAC algorithm (cf. Section 3.3) is the same as for greedy only that the final decision to the requesting host is made on the basis of available bandwidth by both the ingress and egress routers and also on the threshold calculated by them based on the load on each of the links along the path. Since in our case for the admission control scenario the network reduces to a star network the threshold is calculated based on the load on the links of the ingress and egress routers. If the price the host is ready to pay is less than the threshold, though there is enough bandwidth available, the request will be rejected. Later we will consider user models where customers are asked if they are ready to pay a higher price (equal to the calculated threshold) in case the request is reject though there is enough bandwidth available.

### 4.5.1 AAP PCAC Algorithm

The following describes the AAP PCAC Algorithm :

**For Every Edge Router R in the DiffServ Domain.**

**Initialization**

1: **Initialize the VPN Hose - Allocation Table**
2: *Initialize the VPN Hose limits $R^+ \leftarrow R_i^+$ and $R^- \leftarrow R_i^-$.*
3: *Initialize the total rate of ACT entering and leaving the DiffServ domain via* **R**. *$T_R^+ \leftarrow 0$ and $T_R^- \leftarrow 0$ respectively.*
4: *Initialize the bandwidth reserved for Admission Controlled Traffic.*
5: *Initialize the bandwidth reserved for Best Effort Traffic.*
6: *Setup the* **VPN Hose** *from the router with the initialized Hose limits.*
7: **Initialize the Bandwidth Allocation Table**
8: *Specify the bandwidth allocation for each of the FCClasses in both the rate entering and leaving the DiffServ domain via* **R**.
9: **Initialize the Pricing Policy**

**Admission Control**

1: *Request from a host* $h_i \in H_R$ *for admission of Flow Control Class* $z_R \in Z_R$*, for duration* $d$ *and bandwidth* $b_{z_R}$ *through egress router* $L$*,and price ready to pay* $p$*.*
2: *Router performs VPN hose capacity checks from the* **VPN Hose - Allocation Table** *for* $R^+$ *and from the* **Bandwidth Allocation Table** *for ingress available bandwidth per FCClass* $(z_{R^+{}_{free}})$*.*
3: ***if*** $(b_{z_R} \leq z_{R^+{}_{free}})$ ***then***
4: *Send the request message to Egress Router (L) at other end*
5: ***else***
6: *The Request from* $h_i$ *is rejected, a* **Reject** *is sent to the host.*
7: ***end if***
8: *Router L performs VPN hose capacity checks from the* **VPN Hose - Allocation Table** *for* $L^-$ *and from the* **Bandwidth Allocation Table** *for egress available bandwidth per FCClass* $(z_{L^-{}_{free}})$*.*
9: ***if*** $(b_{z_R} \leq z_{L^-{}_{free}})$ ***then***
10: *sends* **Accept** *to* $R$
11: ***else***
12: *sends* **Reject** *to* $R$*.*
13: ***end if***
14: ***if*** *receives* **Reject** *response from the Egress Router* ***then***
15: *sends the* **Reject due to lack of Bandwidth decision** *to host* $h_i$*.*
16: ***else***
17: *Checks the* **Pricing Policy** *for the requested bandwidth* $(b_{z_R})$ *in that FCClass* $(z_R)$ *and sends the appropriate* **Pricing** *and* **Acceptance or Rejection Decision** *to the host* $h_i$*.*
18: ***end if***
19: ***if*** *host* $h_i$ *accepts the* **Pricing** *and* **Admission Decision** *and informs router* $R$ *of confirmation within a period of time* ***then***
20: *Updates* **VPN Hose - Allocation Table** *and sets* $T_R^+ = T_R^+ + b_{z_R}$*.*
21: *Updates the* **Ingress Call Admission Control Table** *with*

- *Flow number*
- *FCClass of the bandwidth requested* $(z_R)$
- *Bandwidth guaranteed for that flow* $(b_{z_R})$
- *Duration requested (in minutes)*

22: *Informs the egress router* $L$ *which updates its* **VPN Hose - Allocation Table** *and sets* $T_L^- = T_L^- + b_{z_R}$*.*
23: *Updates the* **Egress Admission Control Table** *with*

- *Flow number*
- *FCClass of the bandwidth requested* $(z_L)$
- *Bandwidth guaranteed for that flow* $(b_{z_R})$

- *Duration requested (in minutes)*

*24:* ***else***
*25:* *Terminate current request.*
*26:* ***end if***

**Pricing Policy**

*1:* *Calculate the* **Threshold** $\theta$ *using the AAP Algorithm.*
*2:* ***if*** *(Price offered by customer* $\mathbf{p} \geq$ **Threshold***)* ***then***
*3:* *Return* **Accept**.
*4:* ***else***
*5:* *Return* **Reject**.
*6:* ***end if***

**Termination**

*1:* *Request from a host* $h_i$ *to terminate connection from* $R$ *to* $L$ *with bandwidth* $b_{z_R}$.
*2:* *Updates* ***VPN Hose - Allocation Table*** *and sets* $T_R^+ = T_R^+ - b_{z_R}$.
*3:* *Updates* ***Ingress Call Admission Control Table*** *by recomputing and setting the bandwidth in use and available bandwidth.*
*4:* *Sends the flow number and termination signal to the egress router* $L$ *which updates its* ***VPN Hose - Allocation Table*** *and sets* $T_L^- = T_L^- - b_{z_R}$
*5:* *Egress router* $L$ *updates the* ***Egress Call Admission Control Table*** *by recomputing and setting the bandwidth in use and available bandwidth..*
*6:* *host* $h_i$ *is informed of the Termination Confirmation.*

## 4.6 Pricing Policy and Tariff Scheme

In any price-based algorithm the pricing defines the most important factor for determining for an incoming call the price to be paid by the user. While in principle two different pricing policies may be distinguished, (a) the dynamic pricing or (b) the static pricing, their consequences with respect to the question of maximization of the profit are directly visible. In case (a) the profit maximization will require a dynamic approach, based on the current load of incoming calls, since dynamic prices will raise as the load increases.

For further details on work on pricing, pricing models, and charging and accounting technology the reader is referred to [28], [12], and [27].

The users' demands on pricing predictability and their practical concerns, even though the economic optimality can be achieved theoretically in some cases, lead to the selection of static pricing models for PCAC in case of the Greedy algorithm. The two AAP algorithms take into account a congestion-based price increase for an incoming call. Once the pricing policy is selected, the tariff scheme defines the set of parameters utilized to base the pricing decision on. These parameters for the case of the VPN-based hose model are (i) the duration of the call going on

and (ii) the bandwidth required for that call. Those two technical parameters, called Quality-of-Service (QoS) parameters for the hose model, are measurable by appropriate equipment located in the DiffServ network's edge routers. Though scalability concerns are always risen, this approach determines the minimal effort to be put into place, especially if it is compared with the traditional telephone network. Based on those observations, the pricing scheme can be visualized as a three dimensional view per FCClass, the admission controlled traffic of a DiffServ Flow Control Class (cf. Figure 9). The relation of pricing-relevant management information to the bandwidth reservation in a VPN hose (being part of the management table) depicted on the y-axis are based on the selection of the concrete pricing choice of the duration/volume QoS parameter pair or the bandwidth utilized.

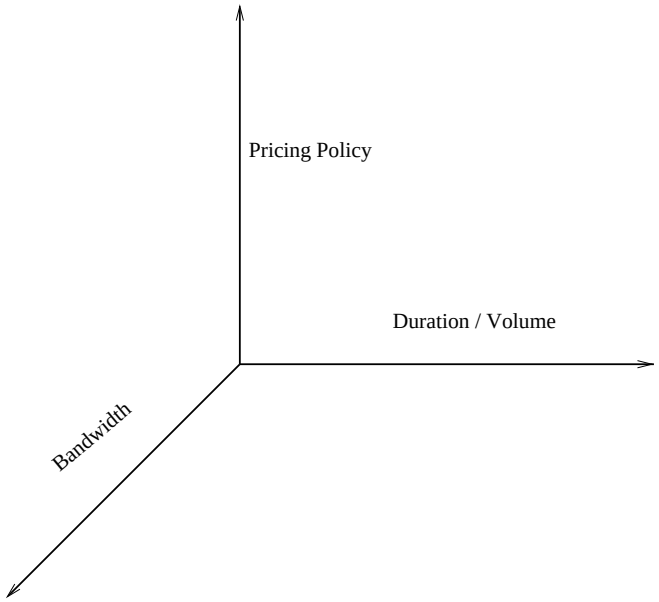


Figure 9: Overview on Pricing Dimensions for PCAC: Parameters and Management Information

For practical purpose, each such FCClass shows an associated 'Profit Factor', which is used to calculate the profit on each requested call depending on the duration and bandwidth utilized by each such call. This is given by the relation $Profit = ProfitFactor * Duration * Bandwidth$. As stated above, the pricing policy for the Greedy PCAC algorithm is a static policy and for the two AAP ones it is a congestion-based one. Therefore, the acceptance of a call may be technology-wise possible, *e.g.*, enough bandwidth is available, but economy-wise it is not granted, since the profit gained from the provider would be below a certain threshold.

# 5 Simulations

We simulated the performance of the previously described system from a high-level point of view. As the basis of our simulation, we used the software written by Schaffhauser and Hauser [24] for the simulation of CAC algorithms with global coordination. We extended the software mainly by adding the hose model bandwidth reservation algorithm and the edge router based CAC algorithms (Greedy, AAP, Tuned AAP, Tuned AAP with User Model).

The focus of our simulation study was a single domain carrying three different classes of traffic. We calculated a feasible hose tree for given maximal in– and out–demands at the access points. The simulation scenario is explained in detail in the following subsections.

## 5.1 Topology Model

The model is one autonomous system (AS) domain containing $5-20$ nodes, in which edge-routers perform call admission control. The topology model for the AS used in our simulations is a simplified SWITCH topology graph (Swiss academic network, Figure 10, see www.switch.ch) containing 8 nodes, where each node is an edge-router. We consider the scenario in which the provider knows

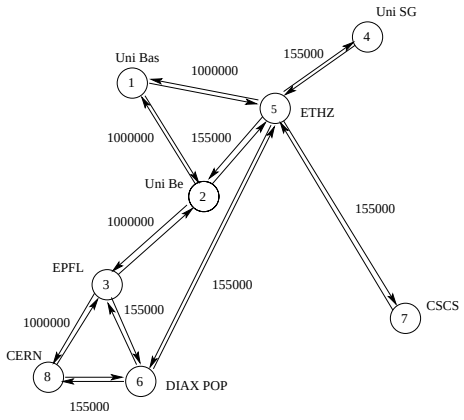


Figure 10: Simplified SWITCH network - AS domain used for our simulation.

the maximal amount of traffic which enters and leaves the AS through each edge router. For our simulations, these values were also taken from the SWITCH web-pages based on the daily statistics in each access point (see Table 5). The link capacities in this network were obtained from the same source.

## 5.2 Traffic Characteristics

The requests are generated in an off-line manner and then stored into a file. Each request contains the following parameters: *sender, receiver, requested bandwidth, duration of connection* and *price.* The *sender* and *receiver* are chosen uniformly at random among all edge-routers in the given AS.

We consider three different DiffServ traffic classes - EF, AF1 and AF2. The EF class represents Voice over IP, AF1 is a high quality video-stream (e.g. movie on demand), and AF2 is a low quality video-stream (e.g. video-conference). Consequently, bandwidth requests for each of them are

| Point ID | $R^+$ value | $R^-$ value |
|---|---|---|
| 1 | 1260 | 4930 |
| 2 | 3360 | 9390 |
| 3 | 2910 | 5950 |
| 4 | 8460 | 16890 |
| 5 | 100780 | 85130 |
| 6 | 55000 | 50000 |
| 7 | 5720 | 14220 |
| 8 | 14900 | 16080 |

Table 5: In– and out–traffic limits (in Kb) for the edge-routers.

- for EF – chosen uniformly at random between 8 and 64 Kb
- for AF1 – constant 2 Mb
- for AF2 – constant 128 Kb.

Each *bandwidth* request is thus non-VBR, but is equal to an effective bandwidth for the given connection.

The average durations for the traffic classes are (based on the assumptions above) - EF - 3 min, AF1 - 90 min and AF2 - 90 min.

The *durations* of requests are negative exponential random variables, and request arrivals are modeled as Poisson processes in each of the senders independently.

The *price* is a linear function of the bandwidth and duration and is calculated as $Price = Profit\ factor * Bandwidth * Duration$, where each traffic class has a different profit factor: EF – 15, AF1 – 1 and AF2 – 10.

## 5.3 Initialization

We used the algorithm given in [19] for provisioning Virtual Private Networks. More precisely, the algorithm computes a hose-model bandwidth reservation in tree form on top of the given network. As was stressed in [19], a tree as a solution is not always optimal, but it has some other advantages regarding implementation of the algorithm and ease of routing in the tree.

This hose provisioning is done in the first initializing phase of our simulation, and in reality it should be done by a Bandwidth Broker or any other entity knowing all in– and out–demands on the edge-routers, as well as the AS domain topology and link capacities.

The basic idea for the admission control is first to check availability, i.e. if there is enough bandwidth for the request, and if yes, to take into account the price of the given request, and based on this, accept or reject the request.

We compared Greedy, AAP, Tuned AAP and Tuned AAP with User Model algorithms while changing the traffic mix (the percentage of each class in one set of requests). More details about the algorithms will be given in the next subsection.

## 5.4 Algorithms

- **Greedy** The Greedy Algorithm has only an availability criterion - it accepts a request whenever there is enough bandwidth. We used it for the sake of comparison because it is frequently used in practice.

- **AAP** This is the algorithm from [2]. Besides availability, AAP takes into account the profit of each request - it has a threshold variable that depends exponentially on the congestion of the edges on the requested path in the network, and if the profit of the request is less than the calculated threshold, the request will be rejected although there is enough bandwidth. This exponential function has a parameter $\mu$ which is equal to $2TF + 2$ (where $T$ is the maximal duration of a request and $F$ is the ratio between the highest and the smallest profit factor) in the standard AAP algorithm, and is used in the worst-case analysis. However, in practice, better results can be obtained by decreasing this parameter.

- **Tuned AAP** is the AAP algorithm with different values of parameter $\mu$.

- **Tuned AAP with User Model** is a family of Tuned AAP algorithms with particular user's behavior - namely, a user can in some cases accept an even higher price than expected in order to reach the calculated threshold in the Tuned AAP algorithm.

We run **Greedy, AAP** and **Tuned AAP** for the following values of $\mu$:

- $\mu \in \{1-4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 54002\}$, where from $1-4$ the interval between two points was 0.5, for the plots in Figure 11.

- $\mu \in \{1-4, 8, 16, 32, 64, 128, 256\}$, where from $1-4$ the interval between two points was 0.5, for the left plot in Figure 12.

- $\mu \in \{1-4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\}$, where from $1-4$ the interval between two points was 0.5, for the middle and right plot in Figure 12.

- $\mu \in \{1-8, 16, 32, 64, 128, 256, 512, 1024, 4096, 54002\}$, where from $1-8$ the interval between two points was 0.5, for the plots in Figures 14 and 15.

In our setting, we have for the standard AAP algorithm that $\mu = 2TF + 2 = 54002$, because we fixed the maximum call duration to be 180 minutes (with 10 time slots per minute, this gives $T = 1800$) and the minimum and maximum profit factors were 1 and 15, respectively.

## 5.5 Comparisons Regarding Traffic-Mix

We ran simulations for different traffic mixes. We considered the following cases:

- What happens if the whole request set consists of only one traffic class?

- What happens if one class is slightly more represented than the other two (e.g. 40% of requests in EF class, and AF1 and AF2 are 30% each)?

We looked at the relation between the tune parameter $\mu$ in the AAP algorithm and the profit obtained for Greedy, AAP and Tuned AAP Algorithms. Since the Greedy and AAP algorithm are independent of $\mu$ they are represented by two horizontal lines. The higher profit obtained by Greedy can be explained by the too optimistic behavior of the AAP algorithm. When choosing $\mu$ between 1.5 and 4 for the first traffic mix and between 2 and 3 for the second traffic mix, Tuned AAP outperforms Greedy, see Figure 11.

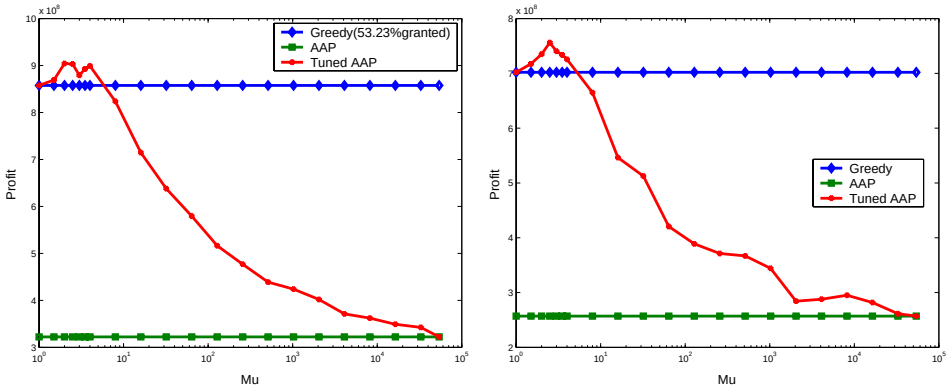


Figure 11: EF = 30%, AF1 = 30%, AF2 = 40% and EF = 30%, AF1 = 40%, AF2 = 30%, Number of requests = 3000.

We also performed simulations with pure traffic with 6000 requests for each of the runs, the results of which are in Figure 12 below. Evidently, the greedy algorithm outperforms the other two.

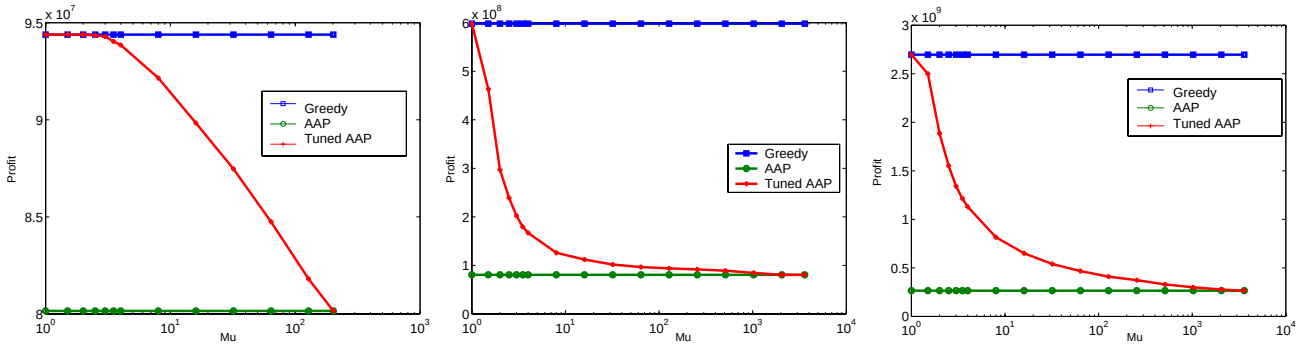


Figure 12: Left: EF = 100%, AF1 = 0%, AF2 = 0%; middle: EF = 0%, AF1 = 100%, AF2 = 0%; right: EF = 0%, AF1 = 0%, AF2 = 100%, Number of requests = 6000.

## 5.6 User Model

In the following we will define the user's behavior when he/she is willing to negotiate with the network about the price of his/her request.

We look in more details at the causes of rejection in the previously explained AAP algorithm: either the rejection is caused by the lack of bandwidth, or it can be that the profit from a request is too small, i.e. smaller than the threshold calculated by the algorithm. In the first case the rejection is unavoidable, while in the latter case there is no physical limit. Thus, a feasible solution in the latter case can be obtained also if the decision "reject" is changed to "accept" in case the user accepts to pay for the request a price equal to the threshold instead of the *in advance fixed price*.

We define two simple models of the users' behavior. The first one is represented by a family of step functions $f_{step}^{mark,val}$ – for a particular scope of ratios between the threshold and the profit, the user decides with a fixed probability whether to accept or reject the increased price. The second one is represented by a family of s-shape function $f_s^{a,b}$ – for a ratio $x$, the probability of acceptance is equal to $f_s^{a,b}(x)$ . Both families are given below, see Figure 13. The user decision is a Bernoulli random variable with probability $f_{\cdot}^{\cdots}(x)$.

$$f_{step}^{mark,val}(x) = val(i), \text{ if } mark(i-1) < x \leq mark(i);$$

$$f_s^{a,b}(x) = 1/(1 + \exp(a * (x - b)));$$

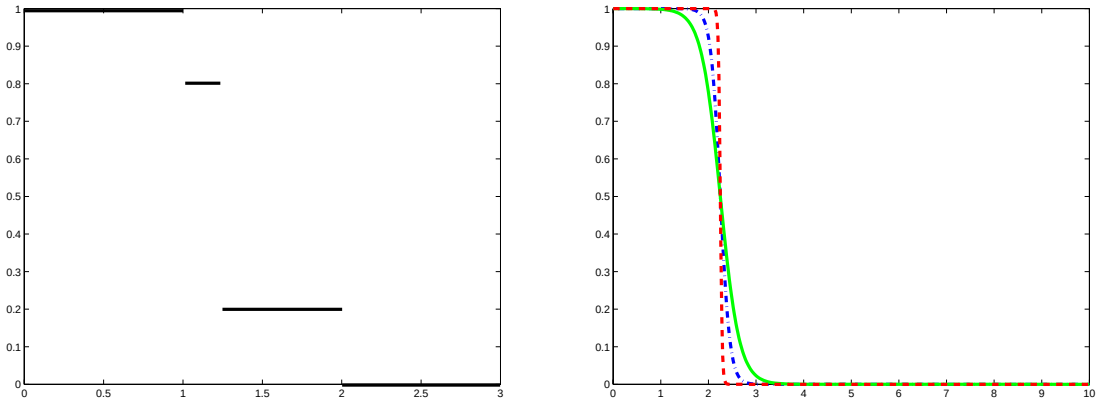


Figure 13: On the left side is an example of a step function with parameters $mar(1, 1.25, 2, \infty)$,$val(1, 0.8, 0.2, 0)$ and on the right are 3 s-shape functions with parameters $(5, 1.25), (10, 1.25), (50, 1.25)$ respectively.

### 5.6.1 Comparisons Regarding User Model

We observed two particular traffic mixes – one on the left side in Figures 14 and 15 contains 30% of EF, 30% of AF1, and 40% of AF2 requests, and on the right side it contains 40% of EF, 30% of AF1, and 30% of AF2 requests. The set of requests contains 3000 requests. Each point on the plots is obtained averaging three consecutive experiments.

We looked at the relation between tuning parameter $\mu$ in the AAP algorithm and the absolute profit obtained by the algorithm with and without user model. "Algorithm with user model" means

that if there is enough bandwidth and the ratio between threshold and request profit is some $x \geq 1$, the request would be accepted with probability $f(x)$, where $f$ is a specific user model's function.

In Figure 14, we compared the absolute profit obtained by greedy algorithm, tuned AAP algorithm without user model, tuned AAP algorithm with step user model's function from Figure 13, and tuned AAP algorithm with s-shape function with parameters $(10, 1.25)$.

The results show that for the parameter $\mu$ chosen between 1.5 and 5 for the first traffic mix, and between 2.5 and 5 for the second traffic mix, the AAP algorithm outperforms the greedy algorithm. In the first case the largest profit is obtained for $\mu = 5$ and the s-shape user model's function, and in the second case for $\mu = 3$ and the step user model's function. Thus, the use of the AAP algorithm with carefully chosen value of $\mu$ leads to the increased profit.

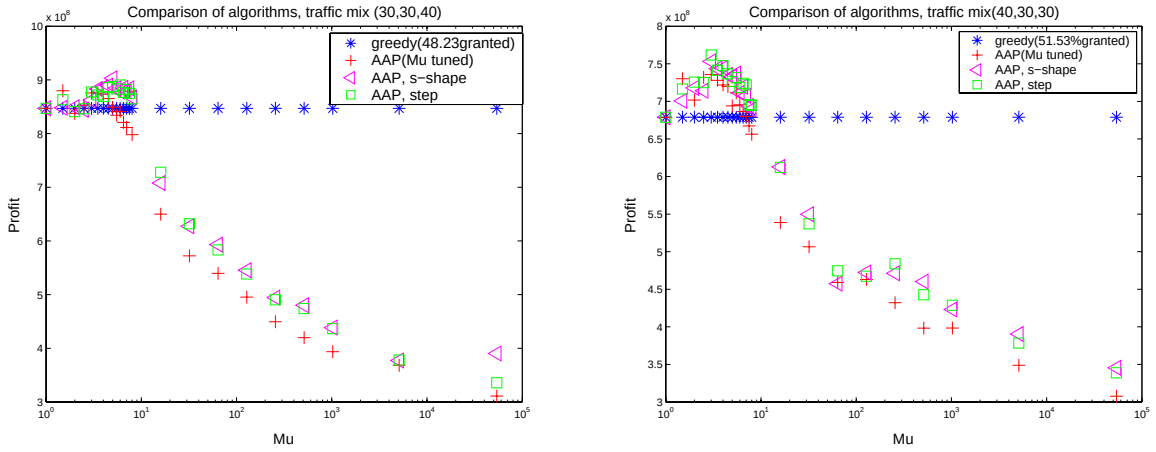


Figure 14: Comparison of the following algorithms: greedy, tuned AAP, tuned AAP with $f_s^{10,1.25}$, and with $f_{step}^{[1,1.25,2,\infty],[1,0.8,0.2,0]}$.

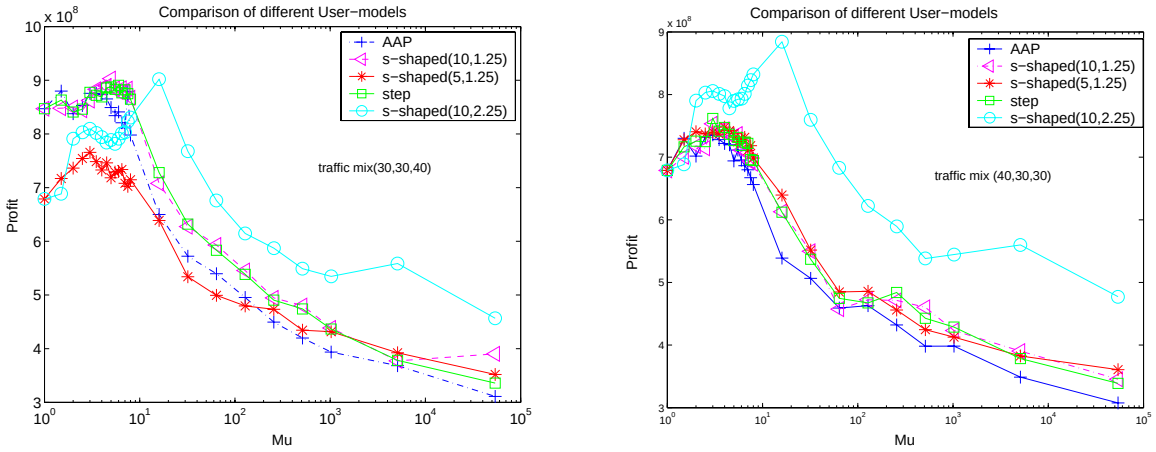


Figure 15: Comparison of different user model functions. The step function used is the function $f_{step}^{[1,1.25,2,\infty],[1,0.8,0.2,0]}$.

As it was expected, in the network with resources for only about 50% of the requests, increasing the probability of acceptance of higher price in the user model has increased the total profit obtained by the tuned AAP algorithm. Results for different parameters of two user model functions are given in Figure 15.

# 6 Conclusion

We have presented a feasible architecture for realizing call admission control and Quality of Service guarantees in DiffServ Networks based on expected profit. The advantages and disadvantages of the proposed architecture are:

**Advantages:** Core routers are not involved in admission control and signaling. Admission control for each connection request is handled by just two edge routers: the router where the connection enters the network, and the router where the connection leaves the network. For each edge router $R$, it is enough to store the four values $R^+$, $R^-$, $T_R^+$, and $T_R^-$, so the storage requirement is minimal. Besides this, every router needs to store the value $b_{u,v}$ for every outgoing link $(u, v)$, and packet scheduling inside the router must be implemented accordingly (to ensure that ACT can always get $b_{u,v}$ bandwidth on the link). Every edge router might also want to keep a list of all admitted connections that go through it (but this is not strictly necessary if only admission control is to be implemented).

Furthermore, the approach combines two generally accepted existing models: the DiffServ model with stateless packet forwarding in the core, and the hose model for bandwidth provisioning in virtual private networks. Thus, it should be feasible to implement the approach.

**Disadvantages:** The values $R^+$, $R^-$ and $b_{u,v}$ must be chosen according to a good estimate of the ACT. If the traffic is highly dynamic, in the sense that at one time one edge router produces most of the traffic and at some other time a second edge router produces most of the traffic, bandwidth reservation according to the hose model might not be the best choice.

From our studies we concluded that the VPN hose model for admission control is a feasible and convenient solution for PCAC in DiffServ networks achieving the said goals. Relatively simple algorithms, Greedy and (Tuned) AAP, can be implemented locally (involving just the ingress and egress routers when a call request is processed) for star networks and thus for general networks with hose-model bandwidth reservation for ACT. A Bandwidth Broker or other entity is only involved in initialization and can resize the VPN hoses if required.

The alternative use of the Greedy or Tuned AAP algorithms with an adequate parameter $\mu$ will depend on the average utilization of the network and the particular traffic mix in the given Autonomous System domain or DiffServ domain. In this way the obtained profit can be increased with price-based call admission control.

Our results suggest a number of issues for future research. The analysis and implementation of restoration algorithms for the hose model would be interesting. Such restoration algorithms could help to achieve deterministic bandwidth guarantees through the use of backup links even in the case of link failures. See [16] for some recent work in this direction.

It would also be interesting to compare the achievable profit in our CAC scheme based on the hose model with that of a scheme in which per-flow reservations are carried out on the individual links of the path used by the flow (as in IntServ). We have demonstrated that the hose model simplifies CAC and enables a distributed and scalable implementation, but it may also lead to unnecessary rejection of requests if the hose capacity is exceeded while the physical links still have sufficient capacity. This trade-off should be studied in more detail.

We have performed some initial experiments with user models where the user may accept a price proposed by the network with a certain probability. Further analysis of the user behavior

would help to validate such user models and to tune parameters in order to further improve the financial gain.

The simulations we carried out were done for a high-level view of the network. Issues such as packet scheduling disciplines in the individual routers were ignored by these simulations. It would be interesting to perform detailed packet-level simulations in order to study the mutual influence of different flow classes in the same network. With such simulations one could also determine whether it is beneficial to allocate separate hoses for different classes or whether the simpler approach of using only one hose for all ACT is sufficient.

# References

[1] R. Adler and Y. Azar. Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. In *Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms SODA'99*, pages 1–10, 1999.

[2] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science FOCS'93*, pages 32–40, 1993.

[3] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive non-preemptive call control. In *Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms SODA'94*, pages 312–320, 1994.

[4] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science FOCS'94*, pages 412–423, 1994.

[5] F. Baker, C. Iturralde, F. Le Faucheur, and B. Davie. Aggregation of RSVP for IPv4 and IPv6 reservations, March 2000. Internet Draft, draft-ietf-issll-rsvp-aggr-02.txt.

[6] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing STOC'00*, pages 735–744, 2000.

[7] A. Bar-Noy, J. Naor, and B. Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *Proceedings of the 31st Annual ACMSymposium on Theory of Computing STOC'99*, pages 622–631, 1999.

[8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[9] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview, June 1994. RFC 1633, Internet Engineering Task Force.

[10] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *Proceedings of ACM Sigcomm 2000*, Aug. 2000.

[11] N. Duffield, P. Goyal, and A. Greenberg. A flexible model for resource management in virtual private networks. In *Proceedings of ACM Sigcomm'99*, 1999.

[12] M. Falkner, M. Devetsikiotis, and I. Lambadaris. An Overview of Pricing Concepts for Broadband IP Networks. *IEEE Communications Surveys*, pages 2–13, 2nd Quarter 2000.

[13] J. A. Garay, I. S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23:180–194, 1997.

[14] M. Garey and D. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York-San Francisco, 1979.

[15] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACMSymposium on Theory of Computing STOC'01*, pages 389–398, 2001.

[16] G. Italiano, R. Rastogi, and B. Yener. Restoration algorithms for virtual private networks in the hose model. In *Proceedings of IEEE Infocom*, 2002.

[17] J. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[18] J. Kleinberg and É. Tardos. Disjoint paths in densely embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science FOCS'95*, pages 52–61, 1995.

[19] A. Kumar, R. Rastogi, B. Yener, and A. Silberschatz. Algorithms for provisioning virtual private networks in the hose model. In *Proceedings of ACM Sigcomm'01*, pages 135–145, 2001.

[20] S. Leonardi. On-line network routing. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, LNCS 1442. Springer-Verlag, Berlin, 1998.

[21] C. A. Phillips, R. Uma, and J. Wein. Off-line admission control for general scheduling problems. In *Proceedings of the 11th Annual ACM–SIAM Symposium on Discrete Algorithms SODA'00*, pages 879–888, 2000.

[22] S. Sargento, R. Valadas, and E. Knightly. Resource stealing in endpoint controlled multi-class networks. In *Proceedings of IWDC'01*, 2001.

[23] S.Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W.Wiess. An architecture for differentiated services, December 1998. RFC 2475, Internet Engineering Task Force.

[24] P. Schaffhauser and F. Hauser. Bandbreitenreservierung in QoS-Netzwerken, March 2002. Student thesis, Computer Engineering and Networks Laboratory, ETH Zürich.

[25] E. Scheinerman and D. Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Wiley, New York, 1997.

[26] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Chichester, 1986.

[27] B. Stiller, J. Gerke, P. Reichl, and P. Flury. A Generic and Modular Internet Charging System for the Cumulus Pricing Scheme. *Journal of Network and Systems Management*, 3(9):293–325, September 2001.

[28] B. Stiller, P. Reichl, and S. Leinen. Pricing and Cost Recovery for Internet Services: Practical Review, Classification, and Application of Relevant Models. *Netnomics – Economic Research and Electronic Networking*, 3(2):149–171, September 2001.

[29] I. Stoica and H. Zhang. Providing Guaranteed Services without Per Flow Management. In *Proceedings of ACM Sigcomm'99*, pages 81–94, 1999.

[30] Z.-L. Zhang, Z. Duan, L. Gao, and Y. T. Hou. Decoupling QoS control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. In *Proceedings of ACM Sigcomm 2000*, Aug. 2000.