MPLS Automatic Bandwidth Allocation via Adaptive Hysteresis☆

N.Akar*, M. A. Toksöz

Electrical and Electronics Engineering Department
Bilkent University, Ankara 06800, Turkey

Abstract

MPLS automatic bandwidth allocation (or provisioning) refers to the process of dynamically updating the bandwidth allocation of a label switched path on the basis of actual aggregate traffic demand on this path. Since bandwidth updates require signaling, it is common to limit the rate of updates to reduce signaling costs. In this article, we propose a model-free asynchronous adaptive hysteresis algorithm for MPLS automatic bandwidth allocation under bandwidth update rate constraints. We validate the effectiveness of the proposed approach by comparing it against existing schemes in (i) voice (ii) data traffic scenarios. The proposed method can also be used in more general GMPLS networks.

Key words:

1. Introduction

MPLS (Multi-Protocol Label Switching) is a forwarding paradigm for IP networks in which IP traffic is carried over an LSP (Label Switched Path) that is established between two MPLS network edge devices using a signaling protocol such as RSVP (Resource Reservation Protocol) or CR-LDP (Constraint-based Routed Label Distribution Protocol) [1]. A label in the IP

☆The work described in this article was carried out with the support of the Scientific and Technological Research Council of Turkey (TUBITAK) under the project EEEAG-106E046.

*Corresponding author

Email addresses: akar@ee.bilkent.edu.tr (N.Akar), altan@ee.bilkent.edu.tr (M. A. Toksöz)

header is used for making forwarding decisions together with label swapping in an MPLS network as opposed to the destination-based routing paradigm of pure IP networks. Such paths are called Label Switched Paths (LSP) and routers that support MPLS are called Label Switching Routers (LSR). In this architecture, ingress LSRs place IP packets belonging to a certain Forwarding Equivalence Class (FEC), for example packets in the same QoS-(Quality of Service) class destined to the same egress LSR, in the corresponding LSP. Core LSRs forward packets based on the label in the header and egress edge LSRs remove the labels and forward these packets as regular IP packets. While the forwarding decision is made on the basis of the MPLS labels, core LSRs employ mechanisms such as per-LSP queuing or per-LSP admission control for QoS management at the LSP level making it possible to provide QoS-based services including circuit emulation [2]. MPLS signaling and routing mechanisms were originally designed for IP routers that can perform packet switching. Generalized MPLS (GMPLS), on the other hand, allows to extend these mechanisms to a combination of devices that can do switching not only in the packet domain but also in time, wavelength, or fiber domains [3].

In this paper, we consider an MPLS LSP carrying aggregate traffic between an ingress LSR and an egress LSR. In this setting, Automatic Bandwidth Allocation (ABA) refers to the process of dynamically changing the bandwidth allocation of the LSP on the basis of the instantaneous aggregate traffic demand of the LSP; see [4] which presents an underlying mechanism to modify the bandwidth and possibly other parameters of an established LSP using CR-LDP (Constraint-based Label Distribution Protocol) without service interruption. One possible approach for ABA is to update the bandwidth allocation very frequently based on the measured load of the LSP every T time units where the measurement interval T is relatively short, e.g., in the order of milliseconds or seconds. This approach is bandwidth-efficient since the allocated bandwidth closely tracks the actual bandwidth requirement of the LSP due to relatively short measurement intervals. However, this method increases signaling costs associated with each bandwidth update. Another simple approach to engineer the LSP is through bandwidth allocation for the largest traffic demand over a relatively long time window, e.g., 24-hour period. This approach does not suffer from signaling costs but the LSP capacity may be vastly underutilized when the average traffic demand is far less than the peak traffic demand. In this article, the signaling costs are indirectly taken into consideration by imposing a limit on the LSP bandwidth update rate. Our goal then is to find an automatic bandwidth allocation scheme for MPLS networks under update rate constraints. We seek model-free and simple-to-implement ABA schemes for practicality purposes. We next describe in more detail the ABA problem studied in this article.

We study two versions of ABA for two different types of traffic (i) circuit-oriented traffic (such as TDM voice) (ii) packet-oriented traffic (such as IP). Circuit-oriented traffic requires a circuit to be formed for a call between two nodes for the traffic to flow. If a circuit can not be formed, the call would be dropped. When a call is admitted in the network, a certain bandwidth needs to be guaranteed for QoS. In case when circuit-oriented traffic is carried through an MPLS network, the concept of a circuit would be replaced by an LSP which employs per-LSP call admission control at the ingress and per-LSP queuing at the core nodes. The main QoS parameter for such traffic is the call blocking probability. In packet-oriented traffic, an establishment of a circuit is not necessary and admission control is typically not used. Instead, users may reduce their rates for congestion control purposes, i.e., TCP congestion control.

For circuit-oriented traffic case, we focus our attention to a single-class traffic scenario in which individual calls arrive at an MPLS ingress node according to a non-homogeneous Poisson process with rate $\lambda(t)$ and call holding times are exponentially distributed with mean $1/\mu$. We set the maximum arrival rate $\lambda_m = \max_t \lambda(t)$ over the time interval of interest. These individual calls are then aggregated into an MPLS LSP in the core network whose bandwidth needs to be dynamically adjusted on the basis of instantaneous aggregate traffic demand. If the bandwidth allocation for the LSP is not sufficient for the incoming call, then either the ingress router will signal the network for a bandwidth update which is eventually accepted by all nodes on the path and the call would be admitted, or the call is dropped. In our model, each individual call requires one unit of bandwidth. We also impose a maximum bandwidth allocation denoted by C_m. We suggest to set C_m to the bandwidth required for the LSP to achieve a desired call blocking probability P_b in the worst case scenario, i.e., $\lambda(t) = \lambda_m$. The parameter C_m can be derived using the Erlang-B formula which gives the blocking probability $B(\rho_m, C_m)$ in terms of the maximum traffic load $\rho_m = \lambda_m/\mu$ and C_m [5]:

$$P_b = B(\rho_m, C_m) = \frac{\rho_m^{C_m}/C_m!}{\sum_{k=0}^{C_m} \rho_m^k/k!}. \tag{1}$$

We introduce a desired update rate parameter β to address the trade-off between bandwidth efficiency and signaling costs. Our goal in ABA is then to select the update decision epochs to dynamically vary the allocated bandwidth $R(t)$ at time t for the LSP as a function of the number of ongoing calls in the system denoted by $N(t)$ so as to minimize the average bandwidth use over time subject to the following three constraints:

- the bandwidth constraint

$$N(t) \leq R(t) \leq C_m, \tag{2}$$

- the blocking constraint that the actual blocking probability should be less than the desired probability P_b,
- the update constraint that the frequency of bandwidth updates should be less than the desired update rate parameter β.

We envision scenarios in which there is a cost associated with each allocated unit of bandwidth and we therefore seek to minimize the total cost under blocking rate and update rate constraints.

For circuit-oriented traffic, call arrivals and departures are the natural decision epochs at which potential bandwidth updates take place. This situation is different for packet-oriented traffic. Instead of making decisions at each packet arrival and departure which would be overwhelming for the ABA scheme, we measure and monitor the traffic at every T time units for packet-oriented traffic case. Let N_k denote the average rate of traffic measured in the interval $[(k-1)T, kT], k = 1, 2, \ldots$. Also let R_k denote the bandwidth allocation in the interval $[kT, (k+1)T], k = 1, 2, \ldots$. At time $t = kT$, the ABA agent will make a decision on the allocated bandwidth R_k on the basis of $N_i, i = k, k-1, \ldots$ and R_{k-1}. The goal of packet-oriented ABA is to dynamically vary $R_k, k = 1, 2, \ldots$ so as to track N_{k+1} while minimizing the average bandwidth use over time subject to the bandwidth constraint

$$R_k \leq C_m, \tag{3}$$

where C_m is set to $\max_k N_k$ while ensuring that the frequency of bandwidth updates be less than the desired update rate parameter β. As opposed to circuit-oriented ABA, in this formulation it is possible that the allocated bandwidth can come short of the actual traffic demand in some interval $[jT, (j+1)T]$, i.e., $R_j < N_{j+1}$ for some j but these short-term problems can be overcome by using buffers and/or end-to-end congestion control especially when the measurement interval T is short.

There are two different approaches ABA for connection-oriented networks (such as MPLS) under update frequency constraints, namely the synchronous and asynchronous approaches. In the synchronous approach, the bandwidth allocation of a connection is adjusted at regularly spaced time epochs with a frequency dictated by signaling constraints. For circuit-oriented traffic, the work in [6] proposes that at a decision epoch, a new capacity is reserved for the aggregate depending on the current system occupancy so that the expected time average of the blocking probability in the forthcoming interval will be less than a predefined limit. A numerical algorithm is proposed by [7] for the efficient numerical calculation of such time-dependent blocking probabilities. It is clear that the approach in [6] in conjunction with [7] is model-based since one should have a stochastic model at hand that describes the actual traffic accurately. A practical example for synchronous bandwidth adjustment for MPLS LSPs but for packet-oriented traffic is the MPLS-TE (Traffic Engineering) Automatic Bandwidth Adjustment for TE tunnels described in [8] and [9], the so-called auto-bandwidth allocator. This automatic bandwidth adjustment mechanism adjusts the bandwidth for each such LSP according to the adjustment frequency configured for the LSP and the sampled output rate for the LSP since the last adjustment without regard for any adjustments previously made or pending for other LSPs. In particular, there are two types of intervals, a Y-type interval (default: 24 hours) and an X-type interval (default: 5 minutes). The average bandwidth requirement is sampled for each X-type interval within a Y-type interval and the highest of these X-type samples is then allocated for the aggregate for the next Y-type interval. The work in [10] also studies other sizing mechanisms that take the average, or the weighted averages of the X-type samples rather than the highest of them as in [8]. These algorithms are model-free, i.e., they do not require a traffic model to be available. Model-based synchronous bandwidth provisioning schemes that take into account packet level QoS requirements such as packet delay and packet loss are proposed in [11] and [12], the first of which also takes the signaling costs into consideration. The ref. [13] proposes a scheme for inter-domain resource management by estimating the inter-domain traffic using a Kalman filter and then forecasting the capacity requirement at a future instant by the use of transient probabilities of the system states. An ARIMA-based traffic model in conjunction with a traffic forecasting and synchronous bandwidth provisioning scheme is proposed in

[14].

Restricting the bandwidth update decisions to regularly spaced time epochs as in the synchronous approach may lead to poor bandwidth usage. In asynchronous ABA, bandwidth adjustments take place asynchronously and corresponding bandwidth update decision instants depend on the current system state. An early work on this approach is by [17] for circuit-oriented traffic which proposes the increase of bandwidth by a constant predetermined step each time the current bandwidth can not accommodate a new call and the bandwidth is decreased by the same constant step when the bandwidth requirement drops back to the original value. Two drawbacks of this proposal are the potential oscillations around a threshold which might substantially increase the signaling load and the wastage of bandwidth as the number of active calls grows due to the use of the constant step. In [18], a bandwidth allocation policy is proposed that eliminates the above problems by applying adaptive upper and lower thresholds and hysteresis. Since the computation of the thresholds require construction of an auxiliary Markov chain with known parameters, the work presented [18] provides a model-based policy. In [19], simple operational rules are derived to determine the amount of bandwidth resources to different connections while balancing between bandwidth waste and connection processing overhead. A heuristic is proposed for a similar problem for a channel sharing application by [20] which however falls short of ensuring a desired update rate. The ref. [21] proposes a scheme for MPLS networks that uses continuous-time Markov decision processes. The proposed scheme decides on when an LSP should be created and how often it should be re-dimensioned while taking into consideration the trade-off between utilization of network resources and signaling/processing load incurred on the network. In [16], the authors present an ARCH-based traffic forecasting and dynamic bandwidth provisioning mechanism. The reference [15] proposes a novel dynamic bandwidth provisioning scheme for traffic engineered tunnels. The mechanism in [15] uses information from the traffic trend to make resizing decisions and is designed to lower signaling and computational overhead while meeting QoS constraints. Although a vast amount of literature exists on dynamic bandwidth provisioning while taking into consideration signaling costs, none of the existing asynchronous algorithms ensure a desired update rate, which is the main goal of this article.

In this paper, we propose a model-free asynchronous adaptive hysteresis algorithm for ABA for connection-oriented networks like MPLS. The proposed algorithm does not assume a traffic model to be available and therefore it is applicable to a wide range of scenarios with unpredictable and non-stationary traffic patterns. The approach uses hysteresis to control the number of updates but the hysteresis operation regime and the band of the hysteresis vary adaptively over time based on system state and the occupancy of a leaky bucket that we incorporate for the aim of update frequency control. To the best of our knowledge, such model-free adaptive hysteresis methods have not been proposed for dynamic bandwidth allocation in the existing literature. Our preliminary results have been reported in [22] but a more extensive study with guidelines on algorithm parameter selection and results concerning packet-oriented data is presented in the current manuscript.

The rest of this article is organized as follows. In Section 2, we describe the proposed model-free algorithm in the circuit-oriented traffic case as well as two model-based conventional approaches. Section 3 describes the proposed algorithm for packet-oriented traffic. In Section 4, we provide numerical examples to validate the effectiveness of the proposed approach for both circuit-oriented and packet-oriented traffic scenarios. Finally, we conclude.

2. Automatic Bandwidth Allocation for Circuit-Oriented Traffic

The ABA problem for circuit-oriented traffic in a single-class scenario is described in Section 1. In this section, we describe two model-based methods that address this particular problem as well as our proposed model-free method. The role of the model-based methods described here is in their use as benchmarks when we compare them against the model-free approach which is the main theme of this article.

- Synchronous model-based ABA: This method is based on [6] where the LSP bandwidth is updated periodically and an optimum bandwidth update policy can be found using transient solutions of Markov chains. However, we present our own implementation for synchronous model-based ABA in this section.

- Asynchronous model-based ABA: This method allows LSP bandwidth updates to occur asynchronously and is therefore more flexible than the synchronous approach. We provide a novel semi-analytic iterative procedure to find the optimum policy where at each iteration, we solve an auxiliary Markov decision problem and carry out simulations to obtain the average bandwidth update rate with the currently found policy. We continue iterations until the actual update rate is equal to the desired update rate.

- The proposed asynchronous model-free adaptive hysteresis-based ABA.

2.1. Synchronous Model-based ABA

In synchronous model-based ABA denoted by *syn-aba*, the system is sampled at regularly spaced epochs at $t = kT, k = 0, 1, 2, \ldots$, where T is the update period and is set to $T = 1/\beta$ so as to ensure the desired update rate β. The minimum bandwidth reservation $R_k = R(kT)$ that guarantees a desired blocking probability P_b throughout the time interval $[kT, (k+1)T]$ is then chosen on the basis of $N_k = N(kT)$ and λ_k which is the average call arrival rate in the interval $[kT, (k+1)T]$. This approach assumes that a-priori information on λ_k is available to the ABA agent.

For calculating R_k, we need to study the following problem. Given the number of calls N_k in progress at time zero with a bandwidth allocation of $R \geq N_k$, the task is to calculate the probability of finding the system in state R at time t denoted by $P(t, R, N_k)$. The average blocking probability in an interval of length T is then given by

$$P_b^{(T)} = 1/T \int_0^T P(t, R, N_k) dt \tag{4}$$

which approaches $B(\rho_k, R)$ as $T \to \infty$ where $\rho_k = \lambda_k/\mu$. The bandwidth allocation R_k is then chosen as the minimum R for which the probability given in (4) stays below the desired blocking probability P_b. This idea is based on [6] that uses the numerical algorithm given in [23] for finding a solution to (4). The particular procedure in [23] requires a spectral expansion of the underlying Markov chain. An alternative algorithm is given in [7] for the same problem by numerically solving a single integral equation. In what follows, we propose a novel numerical procedure for finding the quantity $P_b^{(T)}$ which is not only simple to implement but also it does not have to find the spectral expansion as in [23].

Recall that the number of calls in progress at time 0 is N_k with bandwidth allocation R in the interval $[0, T]$. Let $\pi_j(t), j = 0, 1, \ldots, R$ denote the probability that there are j calls in progress at time $t, 0 \leq t \leq T$ and let $\pi(t) = [\pi_0(t), \pi_1(t), \ldots, \pi_R(t)]$. Also define $z(t), 0 \leq t \leq T$ such that $\frac{d}{dt} z(t) = \pi_R(t)$. It is not difficult to show that

$$\frac{d}{dt}[\pi(t), z(t)] = [\pi(t), z(t)]Q, \tag{5}$$

where

$$Q = \begin{bmatrix} -\lambda_k & \lambda_k & & & & 0 \\ \mu & -(\lambda_k+\mu) & \lambda_k & & & 0 \\ & 2\mu & -(\lambda_k+2\mu) & \lambda_k & & 0 \\ & & \ddots & \ddots & \ddots & \vdots \\ & & (R-1)\mu & -(\lambda_k+(R-1)\mu) & \lambda_k & 0 \\ & & & R\mu & -R\mu & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix},$$

and the vector v defined by $v = [\pi(0), z(0)]$ is a $1 \times (R+2)$ vector of zeros except for a unity entry in the $(N_k+1)^{th}$ position. Also let s be a $(R+2)\times 1$ vector of zeros except for its last entry which is unity. Note that $z(t) = \int_0^t \pi_R(\tau)d\tau$ and therefore also equals $TP_b(t)$. By solving the linear differential equation in (5), we write

$$P_b^{(T)} = \frac{1}{T}z(T) = \frac{1}{T}ve^{Q\,T}s. \tag{6}$$

2.2. Asynchronous Model-based ABA

In this section, we introduce an asynchronous model-based ABA algorithm (denoted by *asyn-aba*) based on Markov decision processes in which LSP bandwidth updates occur asynchronously as opposed to *syn-aba*. Since the theory of Markov decision processes typically assumes a stationary model, we assume that the arrival rate is fixed to λ.

We first define the following auxiliary problem denoted by *aux-aba*. For this problem, we assign a fixed cost of K for each LSP bandwidth update and a cost for allocated unit bandwidth per unit time (denoted by b). Our goal is to minimize the average cost per unit time while maintaining a desired call blocking probability of P_b. We denote the set of possible states in our model by S:

$$S = \{s | s = (s_a, s_r), 0 \leq s_a \leq C_m, \max(0, s_a - 1) \leq s_r \leq C_m\},$$

where s_a refers to the number of active calls using the LSP just after an event which is defined either as a call arrival or a call departure and s_r denotes the bandwidth allocation before this particular event. For each $s = (s_a, s_r) \in S$, one has a possible action of reserving s_r', $s_a \leq s_r' \leq C_m$ units of bandwidth until the next event. The time until the next decision epoch (state transition time) is a random variable denoted by τ_s that depends only on s_a and its average value is given by $E[\tau_s] = \frac{1}{\lambda + s_a\mu}$.

Two types of incremental costs are incurred when at state $s = (s_a, s_r)$ and action s'_r is chosen; the first one is the cost of allocated bandwidth which is expressed as $b\tau_s s'_r$ where b is the cost parameter of allocated unit bandwidth per unit time. Secondly, since each reservation update requires message processing in the network elements, we also assume that a change in bandwidth allocation yields a fixed cost K. As described, at a decision epoch, the action s'_r (whether to update or not and if an update decision is made, how much allocation/deallocation will be performed) is chosen at state (s_a, s_r), then the time until, and the state at, the next decision epoch depend only on the present state (s_a, s_r) and the subsequently chosen action s'_r, and are thus independent of the past history of the system. Upon the chosen action s'_r, the state will evolve to the next state $s' = (s'_a, s'_r)$ and s'_a will equal to either $(s_a + 1)$ or $(s_a - 1)$ according to whether the next event is a call arrival or departure. The probability of the next event being a call arrival or call departure is given as

$$p(s'_a \mid s_a) = \begin{cases} \dfrac{\lambda}{\lambda + s_a\mu}, & \text{for } s'_a = s_a + 1, \\ \dfrac{s_a\mu}{\lambda + s_a\mu} & \text{for } s'_a = s_a - 1. \end{cases}$$

The problem formulation above reduces to the semi-Markov decision model described in depth in [24] where the long-run average cost is taken as the optimality criterion. Relative Value Iteration (RVI)-based algorithms can efficiently be used for solving *aux-aba* as in [24]. Now, we propose a semi-analytic binary search-based procedure to produce a policy for the original ABA problem for circuit-oriented traffic case under update rate constraints. For this purpose, we set the maximum value for the update cost parameter K to K_m. This proposed procedure comprises the following steps.

1) First fix $K = K_m/2, K_u = K_m, K_l = 0$ and fix the bandwidth cost per unit time to b.
2) Solve the *aux-aba* problem to obtain the optimal bandwidth update rate policy.
3) Simulate the overall system using the policy obtained above and monitor the actual bandwidth update rate denoted by β_a.

4) If $\beta_a > \beta + \varepsilon$ for some tolerance parameter ε then set $K_u = K, K = \frac{1}{2}(K_u + K_l)$ and go to step 2. If $\beta_a < \beta - \varepsilon$ then set $K_l = K, K = \frac{1}{2}(K_u + K_l)$ and go to step 2. Otherwise, stop.

The above four-step semi-analytic procedure is denoted by *asyn-aba* due to its asynchronous nature and it can be shown to produce the optimal policy for the original ABA problem as $\varepsilon \to 0$ and when simulations are run long enough to estimate the bandwidth update rate accurately. To the best of our knowledge, the semi-analytic procedure we describe above is novel. The reason to present this algorithm in this article is that it provides the optimum policy for ABA amongst asynchronous model-based algorithms and therefore it can be used as a benchmark for model-free algorithms that will be introduced later.

2.3. Adaptive Hysteresis-based Model-Free ABA

Conventional static hysteresis-based control systems possess two actions, say 0 and 1, a controlled variable x, a threshold parameter T_x on the controlled variable, and a hysteresis (band) parameter d. The actions 0 and 1 help the controlled variable x move lower and higher, respectively. If the value of x drops below the lower threshold $T_x - d$ then the action is 1; if this value is larger than the upper threshold T_x+d then the action is 0. Otherwise, if x is within the hysteresis band $(T_x - d, T_x + d)$ then the previous action does not change; see Fig. 1 depicting the hysteresis relationship between the controlled variable x and the control action. It is clear that the hysteresis control mechanism keeps the controlled variable close to the threshold value T_x whereas by suitably choosing the hysteresis band parameter d, the frequency of action changes can be controlled.

For the ABA problem of interest, we propose an adaptive hysteresis whose threshold and hysteresis parameters are made to vary appropriately in time. For this purpose, we first introduce a leaky bucket of size B_m that is drained at a rate of β unless the bucket is empty. This bucket is incremented by one unit every time a bandwidth update occurs. Let $B(t)$ denote the bucket occupancy at time t. Obviously, the bucket occupancy $B(t)$ staying around the bucket size B_m is indicative of too many recent bandwidth updates that would jeopardize the bandwidth update frequency constraint. In this case, the hysteresis band needs to be widest possible, i.e., $d(t) = C_m$, so that new bandwidth updates would not happen. On the other hand when $B(t) = 0$, the hysteresis band needs to be narrowest possible, i.e., $d(t) = 0$, since otherwise

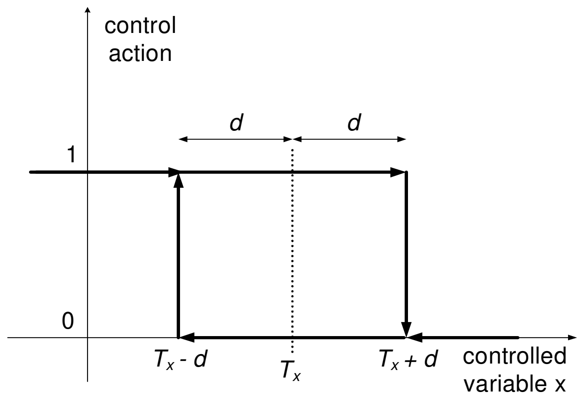


Figure 1: A binary control system using static hysteresis.

bandwidth update credits would be wasted. We therefore allow the hysteresis band $d(t)$ to be proportional with $B(t)$. In particular, we propose to use linear control

$$d(t) = \frac{C_m}{B_m} B(t), \tag{7}$$

which clearly meets the requirements at the boundaries $B(t) = 0$ and $B(t) = B_m$. Other types of control including those in which the relationship between $d(t)$ and $B(t)$ is nonlinear are also studied in the numerical examples. We'll now describe how the hysteresis threshold varies in time and how the bandwidth reservation updates are to be made. For this purpose, let t_i denote the ith bandwidth update epoch and let the bucket occupancy be $B({t_i}^+)$ and the hysteresis parameter be $d({t_i}^+)$ at time ${t_i}^+$. Here, t_i^+ denotes the epoch just after the bandwidth update decision is made upon the event occurring at t_i. We also assume that the current allocation be just changed at time t_i to $R({t_i}^+)$. Let $N(t)$ denote the number of calls in progress in the system. For $t > t_i$, we define two hysteresis thresholds, namely the lower threshold $N(t_i^+) - d(t)$ and the upper threshold $N(t_i^+) + d(t)$. Note that these two thresholds depend on the number of ongoing calls at the instant of the latest update. In time, $N(t)$ will vary randomly, the bucket occupancy will drop linearly, and the hysteresis band $(N(t_i^+) - d(t), N(t_i^+) + d(t))$ will shrink due to the drainage of the leaky bucket. Following t_i, a new arrival at time t will be admitted in the connection if $N(t) < C_m$ but otherwise would be dropped. In the former case, $N(t^+)$ will be set to $N(t) + 1$. If the current reservation can not accommodate the new call, i.e., $R(t) < N(t^+)$, then a bandwidth update needs to take place. On the other hand, when an existing call departs, we write $N(t^+) = N(t) - 1$. We now define an event as the union of an arrival or a departure. After an event takes place at time t, we need to decide on making a bandwidth update if one of two conditions below are met:

$$\begin{aligned} &(i) \quad N(t^+) > R(t) & (8) \\ &(ii) \quad N(t^+) \notin \big(N({t_i}^+) - d(t), N({t_i}^+) + d(t)\big) & (9) \end{aligned}$$

Note that when the second condition is met, the system occupancy does not lie in the hysteresis band making it possible for us to make a bandwidth reservation update. Upon an update decision, say at time t_{i+1}, the new bandwidth reservation and the new bucket values are expressed as:

$$\begin{aligned} R({t_{i+1}}^+) &= \min(C_m, N({t_{i+1}}^+) + \lceil d(t_{i+1}) \rceil), & (10) \\ B({t_{i+1}}^+) &= \min(B_m, B(t_{i+1}) + 1), \text{if} R({t_{i+1}}^+) \neq R(t_{i+1}) & (11) \\ d({t_{i+1}}^+) &= \frac{C_m}{B_m} B(t_{i+1}^+), & (12) \end{aligned}$$

where $\lceil x \rceil$ denotes the smallest integer $\geq x$. Note that for $t > t_{i+1}$, we rewrite the lower and upper thresholds of the hysteresis as $N(t_{i+1}^+) - d(t)$ and $N(t_{i+1}^+) + d(t)$, respectively, and the hysteresis band immediately starts to shrink in size in time after $t = t_{i+1}$. This procedure is repeated afterwards.

In order to describe how the proposed algorithm works, we construct an example system that starts at $t = 0$ and for which $C_m = B_m = 10$, $N(0^+) = 5$, $R(0^+) = 6$, $B(0^+) = 2$ and $\beta = 1/4$ updates/min. We assume at $t = 0^+$, a bandwidth update has just occurred. Note that with this choice of β, we have 15 update opportunities per hour. Instead of a stochastic model, we introduce arrivals and departures at pre-specified instances for this system. The evolution of $N(t)$, $R(t)$, and the lower and upper hysteresis thresholds are illustrated in Fig. 2. Let us first focus our attention to the update epochs. At $t = 3$ and $t = 7$, we have departures from the system and condition (ii) in (9) is satisfied or in other words $N(t^+), t = 3, 7$, lies outside the hysteresis band. Therefore, these two time instances are used for bandwidth updates as described in (10). At the time epochs $t = 14$ and $t = 15$, we have arrivals and we have corresponding bandwidth updates since the conditions (ii) and (i) in (9) and (8) are met for the first and second of these time epochs, respectively. We have two more updates at the time epochs $t = 16.5$ and $t = 19.5$ stemming from condition (ii).

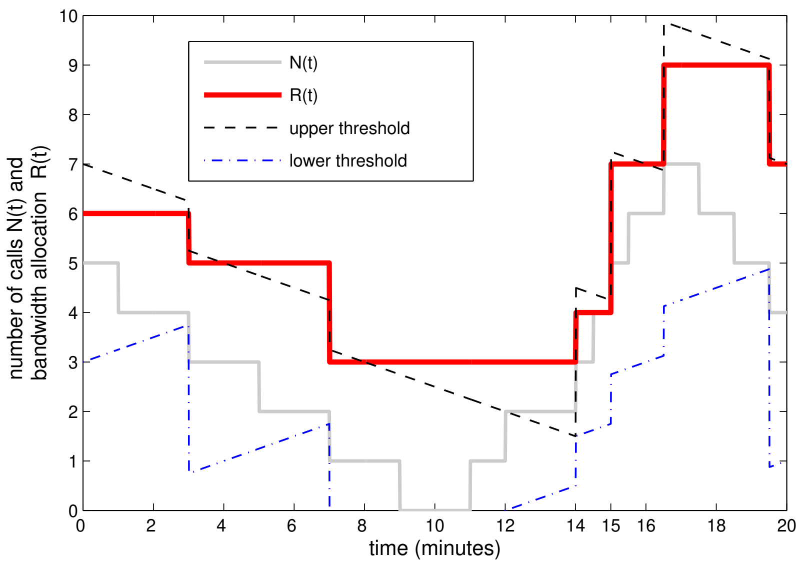


Figure 2: The evolution of number of ongoing calls $N(t)$ and the bandwidth allocation $R(t)$ as a function of t for a sample scenario for which $C_m = B_m = 10$, $N(0) = 5$, $R(0) = 6$, $b(0) = 2$ and $\beta = 1/4$ updates/min.

We study the same scenario but with the desired update rate increased to $\beta = 1$ updates/min in Fig. 3. It is clear that when the desired update rate increases, $R(t)$ starts to track $N(t)$ closely as indicated in Fig. 3 stemming from loosened signaling constraints. To see this, the desired update rate is large relative to the arrival rate and therefore the width of the hysteresis band more rapidly drops toward zero and hence the occurrence of an arrival or departure triggers a bandwidth update in most cases. If we increase β further and practically remove the signaling constraint, the optimal policy would change the bandwidth allocation upon each event as expected in which case $R(t)$ is to track $N(t)$ exactly. These two examples are not meant to quantify the effectiveness of the approach but rather to help the reader visualise the basic features of the proposed algorithm. This algorithm is referred to as *hys-aba* due to its reliance on adaptive hysteresis.

3. Automatic Bandwidth Allocation for Packet-Oriented Traffic

In this section, we will describe the automatic bandwidth allocation mechanism we propose for packet-oriented traffic based on adaptive hysteresis

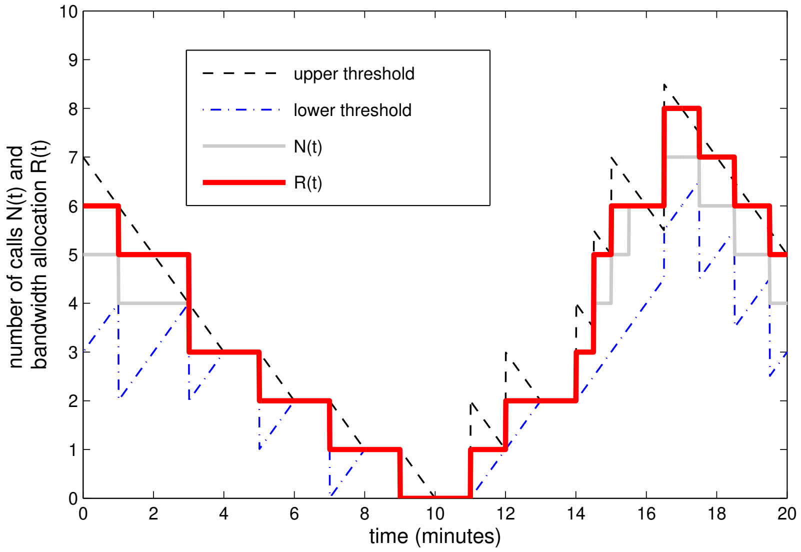


Figure 3: The same example as in Fig. 2 with the desired update rate set to $\beta = 1$ updates/min.

which is along the same lines of the algorithm already described in the previous section. For this purpose, let T denote the measurement window length in units of hours. Recall that N_k denotes the average rate of packet-oriented traffic measured in the interval $[(k-1)T, kT]$, $k = 1, 2, \ldots$ and R_k denotes the bandwidth allocation in the interval $[kT, (k+1)T]$, $k = 1, 2, \ldots$. Also let us assume that $C_m = \max_k N_k$ is already known or we have a fairly accurate estimate of C_m. Let B_k denote the occupancy of the leaky bucket at $t = kT$. Let β denote the update rate in units of updates/hr and let $\kappa = C_m/\eta$ denote the learning parameter where η represents a resolution parameter. At the end of each measurement epoch $t = kT$, the bucket occupancy is always decremented by $\kappa\beta T$ units until the bucket occupancy hits zero, i.e., $B_k = \max(0, B_{k-1} - \kappa\beta T)$. Moreover, let k_i denote the ith bandwidth update epoch and let the bucket occupancy be $B^+_{k_i}$. We also assume that the current allocation be just changed at time k_i to $R_{k_i^+}$. Similar to the previous section, we define a lower threshold $N_{k_i^+} - B_j$ and an upper threshold $N_{k_i^+} + B_j$ for $t_i \leq j \leq t_{i+1}$ until the next bandwidth update but recall that the corresponding hysteresis band $(N_{k_i^+} - B_j, N_{k_i^+} + B_j)$ shrinks unless a bandwidth update takes place. At time $t = kT > k_iT$, we need to decide on making a bandwidth update if the following condition is met:

$$N_k \notin \left(N_{k_i^+} - B_k, N_{k_i^+} + B_k\right) \tag{13}$$

Note that the situation of N_k lying outside the hysteresis band as in (13) gives us an opportunity to make a bandwidth update. Upon a potential update decision, say at time k_{i+1}, the new bandwidth allocation, and the new bucket values are written as:

$$\begin{aligned} B_{k_{i+1}} &= B_{k_{i+1}-1} - \kappa\beta T, && (14) \\ R_{k_{i+1}} &= \min(C_m, N_{k_{i+1}} + B_{k_{i+1}}), && (15) \\ B_{k_{i+1}} &= \min(C_m, B_{k_{i+1}} + \kappa), \text{if} R_{k_{i+1}} \neq R_{k_{i+1}-1} && (16) \end{aligned}$$

The above expressions completely describe the proposed algorithm for the packet-oriented traffic scenario which is also referred to as the *hys-aba* as before. As the algorithm suggests, the bucket occupancy B_k ranges between 0 and C_m and a bucket occupancy value away from these two boundaries is indicative of the update rate compliance to β. The learning parameter κ determines the rate of adapting to changes in the traffic pattern. When κ is large (η is small), changes in traffic are quickly detected at the expense of relatively poor steady-state behavior and possible update rate violations since the bucket can more likely hit the two boundaries in this scenario. On the other hand, when κ is small (η is large) then the algorithm learns about the environment very slowly but with more robust steady-state behavior. In Section 4, we will provide guidelines for selecting η (or equivalently the κ parameter) of the algorithm.

4. Numerical Examples

4.1. Circuit-Oriented Traffic - Single LSP Case

In this example, we study the automatic bandwidth allocation problem for a single LSP carrying circuit-oriented traffic such as voice. We assume stationary Poisson call traffic arriving at the LSP and exponentially distributed call holding times. We are interested in the LSP's average bandwidth allocation as a function of the desired update rate β. In this setting, we compare the performances of the proposed method *hys-aba* with the alternative *syn-aba* and *asyn-aba* approaches that are described in Section 2. For benchmarking purposes, we also present results for the PVP (Permanent Virtual

Path) approach in which the bandwidth to the aggregate is fixed to C_m according to the Erlang-B formula given in (1) as well as the SVC (Switched Virtual Circuit) approach for which the bandwidth allocation is written as $R(t) = min(C_m, N(t))$ and the allocation is updated every time a new call arrives or leaves. Clearly, both approaches guarantee a desired blocking probability of P_b due to the way C_m is set according to (1). However, the PVP approach suffers from poor bandwidth usage whereas the SVC approach suffers from high bandwidth update rates. In this example, we set $C_m = 16$ and the desired call blocking probability to $P_b = 0.01$. The mean service time for calls $(1/\mu)$ is set to 180 seconds. The Erlang-B formula in (1) leads us to set the call arrival rate to $\lambda = 0.0493055$ calls/sec, i.e, $B(\lambda/\mu, C_m) = P_b = 0.01$. We also set the bucket size B_m to 16 in this example unless otherwise stated. The average bandwidth use of each algorithm is given in Fig. 4. Note that in the PVP approach, the allocated bandwidth always equals $C_m = 16$. For the SVC case, the average reserved bandwidth equals $\lambda(1 - P_b)/\mu = 8.785$ since each accepted call will occupy one unit of bandwidth for $1/\mu$ sec. for the current example. We observe that the *hys-aba* approach outperforms the *syn-aba* approach for all values of the desired update rate β by taking advantage of asynchronous updates. While doing so, we note that *hys-aba* is model-free and does not assume a traffic model to be available only except the value of C_m that ensures a blocking probability P_b. Moreover, as expected, for low values of β, *hys-aba* approaches the PVP policy whereas for very large β it approaches the SVC policy. In the *syn-aba* algorithm, even for very large values of β, the bandwidth use would be slightly larger than that of the SVC policy. On the other hand, the model-based optimal algorithm *asyn-aba* produced the best results for all values of β as expected. However, the relatively short gap between the *asyn-aba* and *hys-aba* is the price we pay for not using an a-priori traffic model. We note that the model-free feature of the proposed algorithm allows us to use this approach in a wider variety of scenarios. It is also worthwhile to note that the blocking probabilities we obtained as a result of the *hys-aba* algorithm were within the neighborhood 0.01 ± 0.0001 for all values of β.

4.1.1. Update Rate Compliance

In this example, we monitor the quantity $U(L)$ which refers to the maximum of the update rates each of which is measured over a monitoring window of length L hours over the entire simulation run of 64 hours. For example, when $L = 1$, we set the window size to 1 hour and we count the number

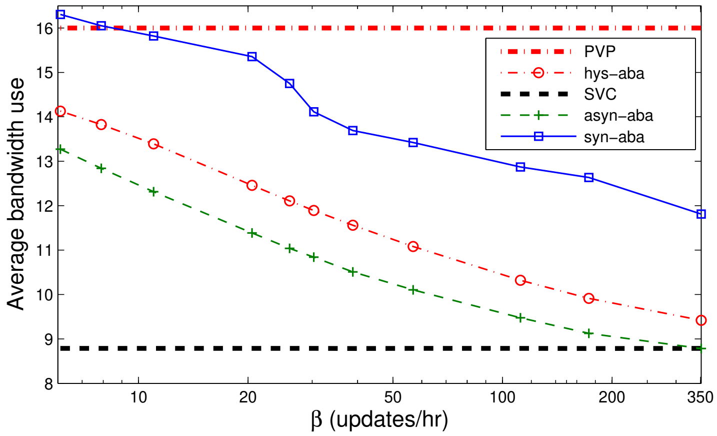


Figure 4: Average bandwidth use of the three methods *syn-aba*, *asyn-aba*, and *hys-aba* as a function of the desired update rate β for the case $\lambda = 0.0493055$ calls/s., $C_m = 16$, $\mu = 1/180$ calls/s., and $P_b = 0.01$.

of updates in each window over a span of 64 hours and $U(1)$ then refers to the maximum of the 64 counts. $U(L)$ is plotted in Fig. 5 as a function of the measurement window size L for the two algorithms *asyn-aba* and *hys-aba* when $\beta = 11$ updates/hr. Note that $U(L)$ approaches β as the window length L increases for both algorithms. However, $U(L)$ converging to β relatively more quickly for the *hys-aba* algorithm compared to the *asyn-aba* algorithm is indicative of update rate compliance of the *hys-aba* algorithm even for shorter time scales. Although the bandwidth utilization performance for *hys-aba* is slightly worse than that of *asyn-aba*, it presents a more strict update rate compliance.

4.1.2. Effect of Hysteresis Band Control Policy

We have proposed to use proportional control given in (7) for hysteresis band control which we refer to as *linear-control*. It is also worthwhile to study whether other variations of hysteresis band control may lead to improved performance. For this purpose, we study two alternative non-linear controls: in the first one denoted by *square-control*, we have $d(t) = \frac{C_m}{B_m^2} B^2(t)$ whereas the second one suggests $d(t) = \frac{C_m}{\sqrt{B_m}} \sqrt{B(t)}$ and is referred to as *sqrt-control*. Note that both controls ensure the desired boundary behavior at $B(t) = 0$ and $B(t) = B_m$. In Fig. 6, we plot the percent gain (with respect to the PVP approach) in average bandwidth use of the linear and non-linear control

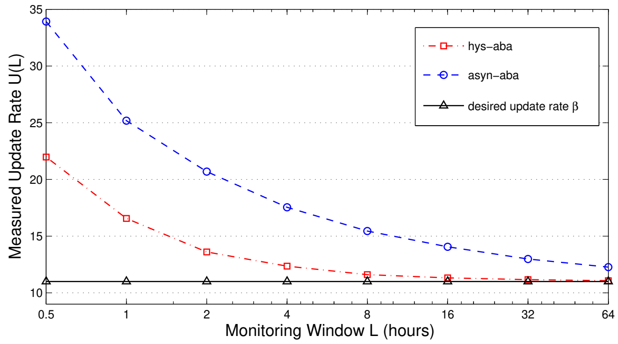


Figure 5: The quantity $U(L)$ as a function of the monitoring window length L for the two algorithms *asyn-aba* and *hys-aba* when β is set to 11 updates/hr.

versions of the proposed algorithms as a function of the desired parameter β. We observe that *linear-control* and *square-control* present similar performance while they outperform *sqrt-control* for all values of β. Throughout the current article, we will employ *linear-control* in all remaining examples and we leave a more detailed study of other control polices for future research.

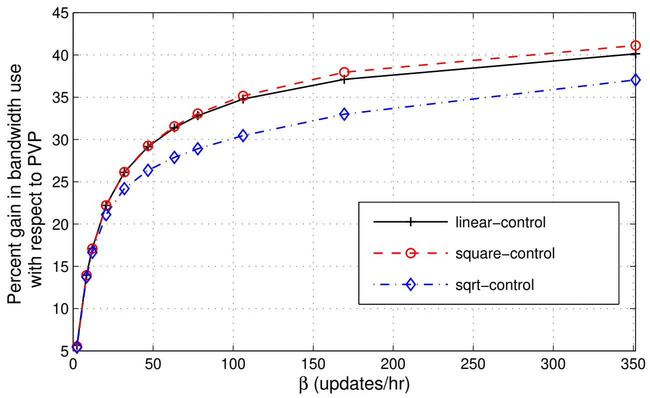


Figure 6: Percent gain in average bandwidth use as a function of β for various hysteresis band control policies.

4.1.3. Effect of B_m

For the same example, we also study the effect of the bucket size B_m on system performance. For this purpose, we vary the bucket size B_m for several values of β and for three values of C_m. We then plot the percent gain in average bandwidth use with respect to the PVP approach in Fig. 7. When varying C_m, we also change λ as a function of C_m according to (1) so that $P_b = B(\lambda/\mu, C_m) = 0.01$. For each simulation run except for the scenario $\beta = 2$, the maximum gains have been obtained when $B_m = C_m$. However, we note that the performance of the proposed system is quite robust with respect to this particular choice of B_m. In the remainder of this study concerning circuit-oriented traffic, B_m will be set to C_m unless otherwise stated.

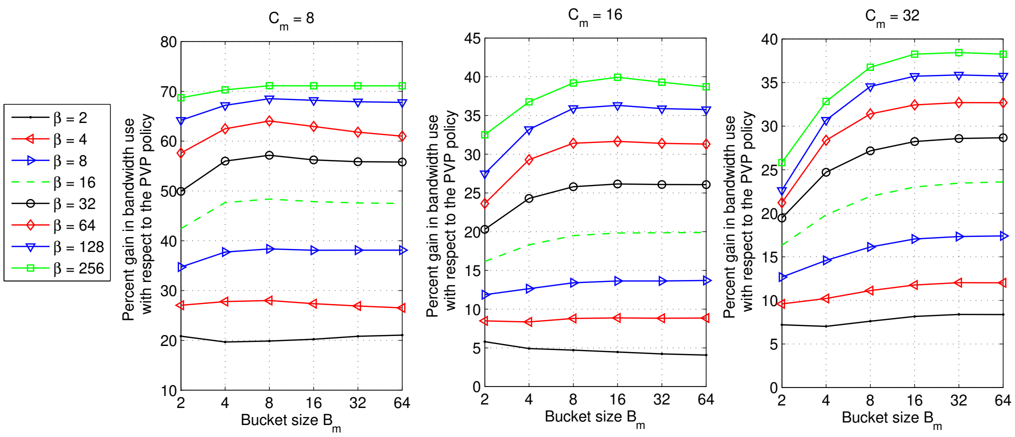


Figure 7: Percent gain in average bandwidth use as a function of B_m for various values of β.

4.1.4. Effect of C_m

In this part of the experiment, we vary C_m as in the previous example. Our goal is to study if the gain in using bandwidth updates in *hys-aba* changes with respect to system capacity C_m. We plot the average bandwidth gains of *hys-aba* with respect to the PVP approach for three different values of C_m as a function of β in Fig. 8. It is clear that the systems with lower capacities benefit more from bandwidth updates using *hys-aba*. This example shows that bandwidth updates are effective even with stationary traffic input with the effectiveness increasing for lower capacity systems. The added benefits of dynamic bandwidth updates stemming from traffic non-stationarity will be explored next.

4.1.5. Non-Stationary Input Traffic

In this part, we employ a non-stationary Poisson call arrival model with call intensity function $\lambda(t) = K + \lambda_0(1 + \alpha \sin(2\pi t/T_p))$. In this model, the

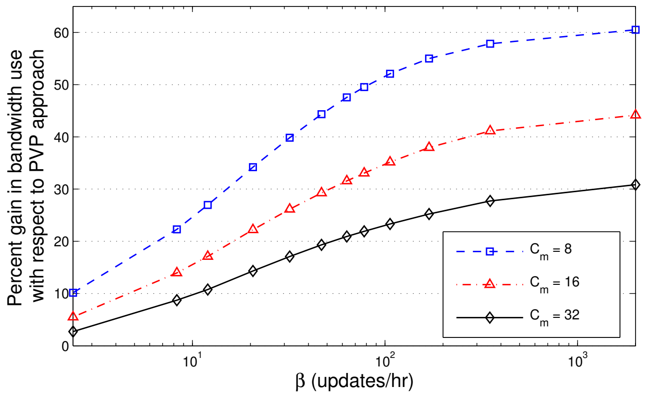


Figure 8: Percent bandwidth gains with respect to the PVP approach as a function of β for three different values of C_m.

average call arrival rate is $K + \lambda_0$ which is independent of the parameter α which is more indicative of the peak-to-peak variability of the incoming traffic. In this example, we fix $K = 0.010$, $\lambda_0 = 0.055$, and $T_p = 1$ day. We then study three scenarios corresponding to the choice of the parameter $\alpha = 1.0, 0.5, 0.0$, in which we use $C_m = 32, 26$, and 20, respectively, based on the Erlang-B formula (1). The intensity of the incoming traffic is depicted in Fig. 9a whereas percent gains in bandwidth use with respect to the PVP approach as a function of the update rate β are depicted in Fig. 9b. As expected, it is clear that benefits of asynchronous bandwidth updates increase as the peak-to-peak variability of the call intensity function increases. When peak-to-peak variability is zero as in the case $\alpha = 0$, there is still a gain stemming from the Poisson nature of arrivals which increases as β increases but these gains become more significant when the mean intensity of call arrivals also changes over time.

4.2. Circuit-Oriented Traffic - Multiple LSPs

Up to now, we have investigated the effects and properties of the proposed algorithm for the single LSP case. However, typically a physical link consists of various LSPs sharing that link. In link sharing, bandwidth that is not used for an LSP can be used by other LSPs. In this model, LSPs signal the network with their bandwidth update requests. Bandwidth release requests are immediately approved but bandwidth increase requests can be accepted as long as the free capacity on the link is large enough to accom-

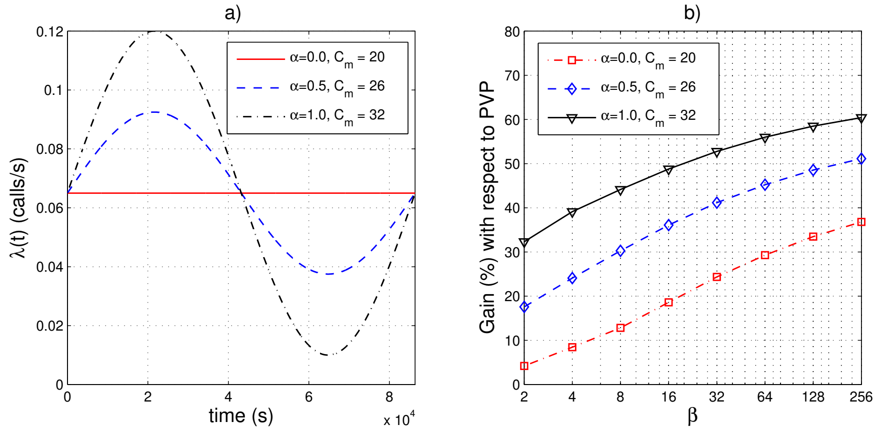


Figure 9: a) Call intensity function for three different values of the parameter α when $K = 0.01, \lambda_m = 0.055$, and $T_p = 1$ b) Percent bandwidth gains with respect to the PVP approach as a function of β for three different values of α

modate this request. In our implementation, each LSP employs a separate instance of the *hys-aba* algorithm for a desired update rate β but also with the *min* function removed in (10) since we occasionally want to allow to allocate a capacity exceeding C_m so as to reduce loss rates. Moreover, when a capacity increase request is received by the network, the network checks whether resources are available. If such a capacity is available, the capacity increase request is accepted. Otherwise, part of the request is accepted to the extent resources are available. In this experiment, we compare a static allocation policy with a dynamic allocation policy which is driven by adaptive hysteresis implemented for each LSP in a distributed manner. In the example we construct, we consider a link carrying M LSPs for voice traffic. Each LSP is fed with stationary Poisson traffic with intensity $\lambda = 0.0493055$ calls/s. and $1/\mu = 180$ sec. with a desired blocking probability of $P_b = 0.01$ leading to a choice of $C_m = 16$ for the individual *hys-aba* algorithms run for each LSP. A static allocation policy is to allocate a fixed C_m amount of capacity to each LSP which ensures a blocking probability of 0.01. A dynamic allocation policy is one in which *hys-aba* is run for each LSP as described before. For comparison purposes, we fix the link capacity to MC_m so that the link capacity will be used only for voice traffic. In Fig. 10, we plot the blocking probability for various values of M as a function of the desired update rate β. We observe that significant blocking probability reductions are attainable with distributed dynamic allocation policies. Such performance

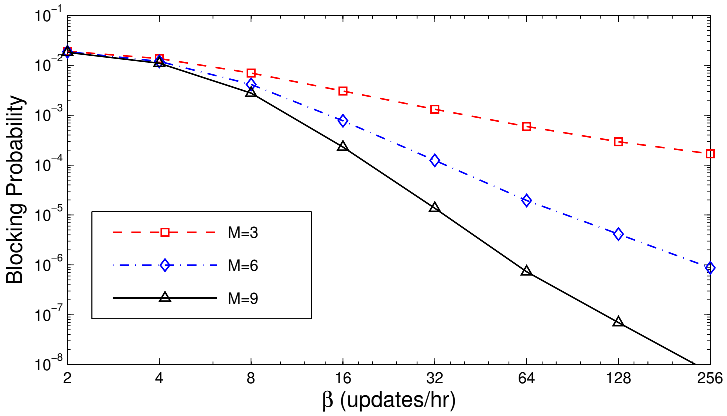


Figure 10: Overall blocking probability as a function of β for various values of M when dynamic bandwidth allocation is used.

improvements become more evident when the number of LSPs increases due to statistical multiplexing effects. However, when β is very low, it is also possible for a dynamic allocation policy to produce a blocking probability slightly larger than that of the static allocation policy.

4.3. Packet-Oriented Traffic

In this example, we study the automatic bandwidth allocation problem for a single LSP carrying packet-oriented data traffic. We will study two different scenarios for packet-oriented traffic; the first one is synthetic data traffic and the second one is obtained using a one-day traffic trace taken from a traffic data repository maintained by the MAWI (Measurement and Analysis on the WIDE Internet) Working Group of the WIDE Project [25]. In both scenarios, we compare our results obtained by *hys-aba* with CISCO's auto-bandwidth allocator (referred to as *cisco-aba* in short) which is a measurement-based technique that is commonly used for dynamic bandwidth allocation for data traffic [8],[9]. In Cisco's auto-bandwidth allocator, there are two types of intervals, a Y-type interval (default: 24 hours) and an X-type interval (default: 5 minutes). The average bandwidth requirement is sampled for each X-type interval within a Y-type interval and the highest of these X-type samples is allocated for the aggregate for the next Y-type interval. The minimum and maximum allowed allocations are also configurable [9]. Although there are also other optional configuration parameters given in [9], we use the basic auto-bandwidth allocator as given in [8] for comparison purposes. The frequency of bandwidth adjustments is then simply the reciprocal of the configured Y-type interval length. Therefore, the Y-type interval length is set to the reciprocal of the desired update rate β of our proposed algorithm when these two algorithms are compared against each other. Consequently, these two algorithms will then have the same average bandwidth adjustment rate.

4.3.1. Synthetic Data Traffic

For synthetic traffic, we use the particular $M/G/\infty$ flow-based traffic model, the so-called Poisson Pareto Burst Process (PPBP) presented in [26] with the further assumptions:

- Measurement window length T is set to 1/12 hrs (5 mins.) and simulation time is set to thirty days corresponding to $K = 30 \cdot 24 \cdot 12 = 8640$ overall measurement epochs.
- The flow arrival process is a non-homogeneous periodic Poisson process with intensity $\lambda(t)$ with a period of one day whose daily behavior is sketched in Fig. 11. This particular shape for the intensity function is obtained by aggregating traffic belonging to six pairs of cities in different time zones and by using the model proposed in [27] that determines the link activity on the basis of the population of the cities involved and the relative time zones of the cities that constitute the pairs. The city pairs we used are New York-Athens, New York-New Delhi, Paris-Athens, Paris-New Delhi, Berlin-Athens, and Berlin-New Delhi.
- All packets belonging to the same flow are assumed to have the same packet size. On the other hand, the packet size distribution is obtained from the traffic traces from [25] as given in Table 1. For example, packet size for a new flow will be uniformly distributed in the interval [33,64] bytes with probability 0.2955, or will be uniformly distributed between 65 and 128 with probability 0.2621, and so on.
- Considering each flow to be associated with a file transfer, the file size distribution is assumed to be Pareto distributed with mean 562.5 Kbytes and shape parameter $\gamma = 1.4$ which corresponds to an asymptotically self-similar model with Hurst parameter $H = 0.8$; we refer the reader to [26] for details on the PPBP and its parameterization.

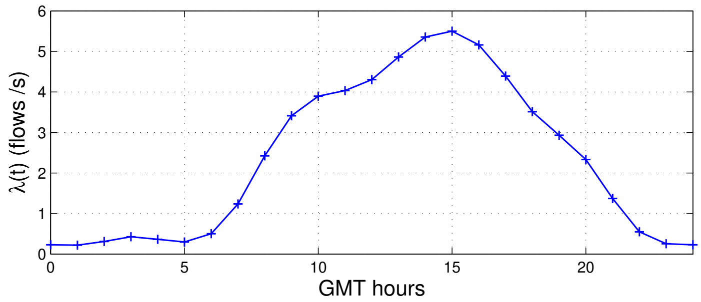


Figure 11: The intensity $\lambda(t)$ of the flow arrival process used in the $M/G/\infty$ traffic model.

Table 1: Packet Size Distribution from [25]

Size Range (bytes)	# Packets	Probability
33-64	2171017	0.2955
65-128	2519797	0.2621
129-256	574504	0.0598
257-512	297002	0.0309
513-1024	251686	0.0262
1025-2048	3800020	0.3953

- When the file size S (in bytes) is determined together with the packet size P (in bytes), the inter-arrival times denoted by I between two successively arriving packets of the same flow will be deterministically chosen so that the rate of traffic generated by this flow will have a mean of 300 Kbps. In particular, I is set to $I = 8P/300$ in milliseconds and an overall number of $\lceil \frac{S}{P} \rceil$ packets will be generated for this particular flow.

- Maximum physical link capacity C_m is set to 28 Mbps which is slightly larger than the average bit rate obtained when $\lambda(t) = \lambda_m \approx 5.5$. The parameter η is set to 32.

Under these assumptions, we have compared the performances of the two bandwidth allocation algorithms *cisco-aba* and *hys-aba*. For visualization purposes, we provide one-day snapshots of the bandwidth allocation results of *cisco-aba* and *hys-aba* for two different values of β in Fig. 12. For this particular scenario, it is clear that *hys-aba* does a better job in tracking the actual traffic in both cases of average intensity increasing or decreasing espe-

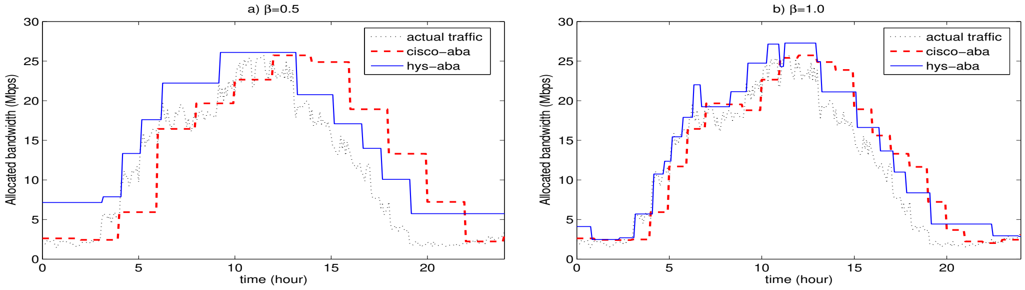


Figure 12: The bandwidth allocation dictated by *cisco-aba* and *hys-aba* algorithms over a one-day period for the $M/G/\infty$ traffic model for two different values of β: a) $\beta = 0.5$ b) $\beta = 1$.

cially for low β. On the other hand, when the average intensity is increasing, the *cisco-aba* bandwidth allocation algorithm appears to lag the actual traffic whereas an overbooking is evident when the average intensity is decreasing within a day for $\beta = 0.5$. Similar observations are obtained when $\beta = 1$ but the tracking performances of both algorithms get much better as would be expected but still favoring the *hys-aba* algorithm. The reason behind the outperformance of *hys-aba* is that *cisco-aba* makes periodic decisions and after a decision is made, one needs to wait until the next decision epoch irrespective of potential significant changes in between two successive decision epochs. On the other hand, such significant changes are captured in the *hys-aba* algorithm while still maintaining a desired update rate β.

We now study the impact of the choice of the parameter η on algorithm performance. Recall that N_k denotes the average rate of packet-oriented traffic measured in the interval $[(k-1)T, kT]\,, k = 1, 2, \ldots$ and R_k denotes the bandwidth allocation in the interval $[kT, (k+1)T]\,, k = 1, 2, \ldots$. We define bandwidth gain as

$$\text{gain} = \frac{\sum_{k=1}^{K-1} (C_m - R_k)}{(K-1)C_m}, \tag{17}$$

where K is the total number of measurement epochs used in the simulation. To study the impact of η, we vary the parameter η and β and obtain by simulations the percent gain in average bandwidth use with respect to C_m. Our results are depicted in Fig. 13a. We observe that gains are generally robust with respect to the choice of η. However, note that a too aggressive choice of η can also lead to occasional lagging of the actual traffic. To quantify this effect, we introduce the concept of *under-provisioning* which arises when the bandwidth allocation lags the actual traffic requirement. We also introduce a parameter called R_U to denote the under-provisioning rate which is defined as the ratio of the area between the bandwidth allocation and the actual traffic when the former lags the latter to the overall number of transmitted bits over a measurement window. Mathematically,

$$R_U = \frac{\sum_{k=1}^{K-1} max(0, N_{k+1} - R_k)}{\sum_{k=1}^{K-1} N_{k+1}}. \tag{18}$$

We attempt to quantify the QoS (Quality of Service) received by an LSP through the parameter R_U. A high R_U is indicative of long time epochs during which the allocated bandwidth to the LSP lags the actual traffic. Depending on the traffic mix using this LSP, a high R_U means relatively higher loss rates for UDP-type traffic and reduced throughput for TCP-type flows. However, the focus of this paper is on the study of the under-provisioning rate R_U and the detailed study of this parameter on packet-level QoS for UDP and TCP traffic for different traffic mix scenarios is left outside the scope of this paper. Fig. 13b depicts the under-provisioning rate R_U as a function of η which tends to first drop by increasing η but then starts to slightly increase once a certain limit is reached. Based on these examples, we tend to believe that a choice of η in the vicinity of 16 and 32 provides acceptable performance both in terms of gain and R_U.

In Table 2, we compare the gain in bandwidth use and R_U for the two algorithms *cisco-aba* and *hys-aba* for various values of β and for two separate choices of $\eta = 16$ and $\eta = 32$. We have the following observations:

- The under-provisioning rate R_U characterizing the lagging of the bandwidth allocation with respect to the actual traffic is relatively high for *cisco-aba*. On the other hand, the parameter R_U for the *hys-aba* algorithm is quite acceptable for both choices of η. There is definitely a larger gain in employing *cisco-aba* which comes at the expense of significant performance deterioration in terms of R_U.

- Increasing η slightly decreases R_U but slightly decreases the bandwidth gain as well. The parameter η should be chosen so as to control gain or R_U depending on whichever is more critical for the network. However, we also note that in the vicinity of $\eta = 16$ and $\eta = 32$, the performance of the *hys-aba* algorithm is quite robust.

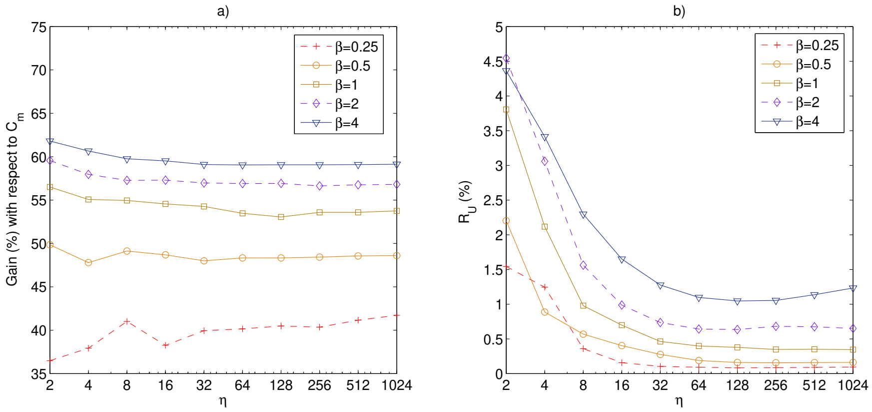


Figure 13: Bandwidth gain and the under-provisioning rate R_U for varying η when C_m is fixed at 28 Mbps.

Table 2: Bandwidth gain and the under-provisioning rate R_U when $C_m = 28$ Mbps and $\eta = 16, 32$

β	*cisco-aba* $\%R_U$	*hys-aba* $\%R_U$ $\eta = 16$	*hys-aba* $\%R_U$ $\eta = 32$	*cisco-aba* % gain	*hys-aba* % gain $\eta = 16$	*hys-aba* % gain $\eta = 32$
0.25	18.36	0.16	0.10	45.83	38.25	39.94
0.50	8.36	0.40	0.28	52.56	48.69	48.00
1.00	5.01	0.70	0.46	56.35	54.55	54.27
2.00	2.87	0.99	0.74	58.27	57.29	56.97
4.00	2.27	1.65	1.28	59.88	59.51	59.09

- The performance in terms of R_U improves for *cisco-aba* for increasing β as expected. However, this situation is reversed for *hys-aba* whose performance in terms of R_U slightly drops for increasing β. To explain, when β increases, then the average bucket occupancy drops and so is the average hysteresis band. Consequently, the bandwidth allocation becomes less conservative from (15) leading to slight increase in R_U with increasing β.

- When β is very high, both algorithms appear to present similar performance in terms of both gain and R_U.

The under-provisioning rate definition we proposed is not associated with actual packet loss rate which is a more relevant measure in packet-switched

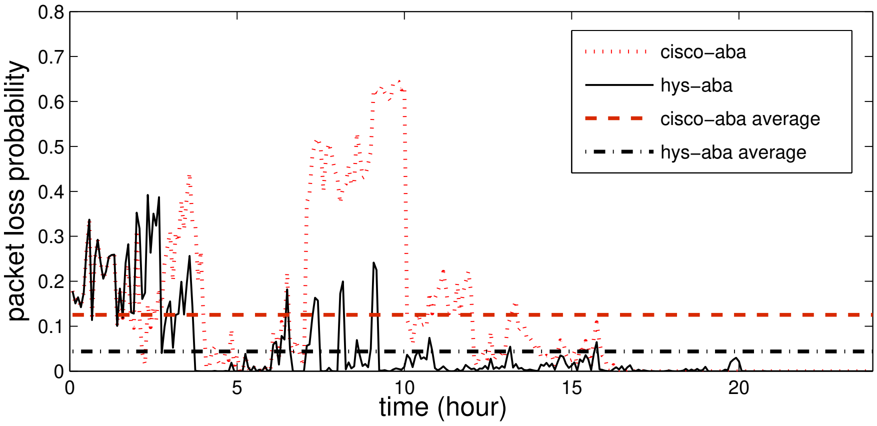


Figure 14: Instantaneous and average packet loss rates for the *cisco-aba* or *hys-aba* algorithms for synthetic traffic when $\beta = 0.5$ and $\eta = 32$.

networks. Therefore, we also carry out packet-level simulations for the particular case of $\beta = 0.5$ and $\eta = 32$ to find the actual packet loss probability using these two algorithms as a function of time. In this case, we assume that all traffic is UDP-type. While doing so, we feed the incoming traffic to a queue with a capacity of 100 packets which has a service rate dictated by the bandwidth allocation algorithm (*cisco-aba* or *hys-aba*). The instantaneous loss rate as a function of time as well as the long-term average packet loss rate for (*cisco-aba* or *hys-aba*) algorithms are given in Fig. 14. The results demonstrate that the maximum loss rate over a day as well as the average packet loss rates are much higher for the *cisco-aba* algorithm than *hys-aba*.

4.3.2. Real Traffic Trace

We also compare the bandwidth allocation results using a real traffic trace (for one single day) obtained from the WIDE backbone at Sample Point B on May 14, 1999 for US-Japan link with 10 Mbps link speed [25]. The measurement window T is again set to 5 minutes. In this study, we set $C_m = 8.7$ Mbps and $\eta = 32$. We note that this traffic record is relatively bursty compared with the synthetic traffic we used in the previous subsection. For visualization purposes, we have one-day snapshots of the bandwidth allocations made by *cisco-aba* and *hys-aba* algorithms for two different values of β in Fig. 15. Fig. 16a (Fig. 16b) depicts the gain (R_U) as a function of η for various values of β from which we conclude that the gain and R_U perfor-

Table 3: Bandwidth gain and under-provisioning ratio R_U for the traffic trace when C_m = 8.7 Mbps and $\eta = 16, 32$

β	*cisco-aba* % loss	*hys-aba* % loss $\eta = 16$	*hys-aba* % loss $\eta = 32$	*cisco-aba* % gain	*hys-aba* % gain $\eta = 16$	*hys-aba* % gain $\eta = 32$
0.25	1.59	0.09	0.09	8.71	8.96	7.75
0.50	1.90	0.20	0.19	14.06	10.87	10.65
1.00	0.77	0.39	0.30	16.26	14.68	14.98
2.00	1.86	0.58	0.46	19.88	17.12	17.27
4.00	1.65	1.00	0.82	22.01	20.44	20.34

mance of the algorithm is similar to the behavior we observed for synthetic traffic and that it is robust in the vicinity of $\eta = 16$ and $\eta = 32$. We also provide the bandwidth gain and under-provisioning rates of the two algorithms *cisco-aba* and *hys-aba* for various values of β in Table 3 for this traffic trace. Our observations are as follows:

- Again, the *hys-aba* algorithm provides better performance in terms of R_U at the expense of slightly larger average bandwidth use. However, the two algorithms presented relatively similar performances for this specific trace when compared with the previous synthetic traffic scenario.

- The performance of *cisco-aba* drops when the traffic is non-stationary and in particular when the traffic intensity rapidly increases (relative to the update rate). In this particular traffic trace, such behavior is observed only once for a short duration which caused losses but the eventual loss rate over a day is quite low for *cisco-aba* when compared with the previous synthetic traffic scenario.

- The algorithm *hys-aba* can be preferred over *cisco-aba* if the network is to provide a service to its customers where R_U is critical for the underlying service.

For the particular scenario of $\beta = 0.5$ and $\eta = 32$ for the real traffic trace, we also carry out packet-level simulations for UDP-type traffic. Again, the service rate of the queue is governed by the bandwidth allocation algorithm (*cisco-aba* or *hys-aba*) and the queue storage capacity is fixed to 100 packets.

The instantaneous loss rate as a function of time as well as the long-term average packet loss rate for (*cisco-aba* or *hys-aba*) algorithms are given in Fig. 17. As in the synthetic traffic case, the results of Fig. 17 demonstrate that the maximum loss rate as well as the average packet loss rate are much higher for the (*cisco-aba* algorithm than *hys-aba*).

5. Conclusions

In this article, we introduce a simple model-free adaptive hysteresis-based bandwidth allocation algorithm for automatic provisioning of LSPs in an MPLS network. For circuit-oriented traffic, we show that our proposed model-free mechanism provides very close to optimal results for which the latter would require an a-priori stochastic model which typically is not available due to non-stationary and unpredictable traffic patterns. We also show the benefits of adaptive hysteresis-based dynamic allocation in non-stationary traffic scenarios and link sharing problems related to effective use of bandwidth in MPLS networks. We also extend the idea to packet-oriented data traffic and compare the proposed algorithms via existing methods in various scenarios. Although more work is needed, our preliminary results show promise in demonstrating the effectiveness of model-free dynamic bandwidth allocation algorithms for MPLS networks in which LSPs carry aggregates of flows.

References

[1] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol Label Switching Architecture, RFC 3031 (Proposed Standard) (Jan. 2001).
URL `http://www.ietf.org/rfc/rfc3031.txt`

[2] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, J. Heinanen, Multi-Protocol Label Switching (MPLS) Support of Differentiated Services, RFC 3270 (Proposed Standard), updated by RFC 5462 (May 2002).
URL `http://www.ietf.org/rfc/rfc3270.txt`

[3] E. Mannie, Generalized Multi-Protocol Label Switching (GMPLS) Architecture, RFC 3945 (Proposed Standard) (Oct. 2004).
URL `http://www.ietf.org/rfc/rfc3945.txt`

[4] J. Ash, Y. Lee, P. Ashwood-Smith, B. Jamoussi, D. Fedyk, D. Skalecki, L. Li, LSP Modification Using CR-LDP, RFC 3214 (Proposed Standard) (Jan. 2002).
URL `http://www.ietf.org/rfc/rfc3214.txt`

[5] L. Kleinrock, Queuing Systems, Vol. 1, Theory, John Wiley, New York, 1975.

[6] U. Mocci, P. Pannunzi, C. Scoglio, Adaptive capacity management of virtual path network, in: IEEE Globecom, London, 1996, pp. 750–754.

[7] J. T. Virtamo, S. Aalto, Calculation of time-dependent blocking probabilities, Proceedings of the ITC Sponsored St. Petersburg Regional International Teletraffic Seminar: Teletraffic Theory as a Base for QoS: Monitoring, Evaluation, Decisions (1998) pp. 365–375.

[8] Cisco, Cisco MPLS autobandwidth allocator for MPLS traffic engineering: A unique new feature of Cisco IOS software, available at `http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/mpatb_wp.htm` (2001).

[9] Cisco, MPLS Traffic Engineering (TE)-Automatic Bandwidth Adjustment for TE Tunnels, available at `www.cisco.com/en/US/docs/ios/mpls/configuration/mp_te_auto_bandwdth.html` (2009).

[10] S. Dasgupta, J. C. de Oliveira, J.-P. Vasseur, Dynamic traffic engineering for mixed traffic on international networks: Simulation and analysis on real network and traffic scenarios, Computer Networks 52 (11) (2008) 2237 – 2258.

[11] H. T. Tran, T. Ziegler, Adaptive bandwidth provisioning with explicit respect to QoS requirements, Computer Communications 28 (16) (2005) 1862 – 1876.

[12] H. van den Berg, M. Mandjes, R. van de Meent, A. Pras, F. Roijers, P. Venemans, QoS-aware bandwidth provisioning for IP network links, Computer Networks 50 (5) (2006) 631 – 647.

[13] T. Anjali, C. Scoglio, G. Uhl, A new scheme for traffic estimation and resource allocation for bandwidth brokers, Computer Networks 41 (6) (2003) 761–777.

[14] B. Krithikaivasan, K. Deka, D. Medhi, Adaptive bandwidth provisioning envelope based on discrete temporal network measurements, in: IEEE INFOCOM, Hong Kong, 2004, pp. 1786–1796.

[15] S. Dasgupta, J. C. de Oliveira, J. -P. Vasseur, Trend based bandwidth provisioning: an online approach for traffic engineered tunnels, in: Proceedings of the Next Generation Internet Networks (Euro-NGI 2008), Krakow, Poland, April 28-30, 2008

[16] B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic, IEEE/ACM Trans. on Networking 15 (2007) pp. 683-696.

[17] S. Ohta, K. Sato, Dynamic bandwidth control of the virtual path in an Asynchronous Transfer Mode network, IEEE Trans. Commun. 40 (7) (1992) 1239–1247.

[18] A. Orda, G. Pacifici, D. E. Pendarakis, An adaptive virtual path allocation policy for broadband networks, in: INFOCOM (1), 1996, pp. 329–336.

[19] H. Levy, T. Mendelson, G. Goren, Dynamic allocation of resources to virtual path agents, IEEE/ACM Trans. Networking 12 (4) (2004) 746–758.

[20] N. Argiriou, L. Georgiadis, Channel sharing by rate-adaptive streaming applications, Perform. Eval. 55 (3-4) (2004) 211–229.

[21] T. Anjali, C. Scoglio, J. C. de Oliveira, I. F. Akyildiz, G. Uhl, Optimal policy for label switched path setup in MPLS networks, Computer Networks 39 (2) (2002) 165–183.

[22] N. Akar, Model-free adaptive hysteresis for dynamic bandwidth reservation, in: Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07. 15th International Symposium on, 2007, pp. 331–336.

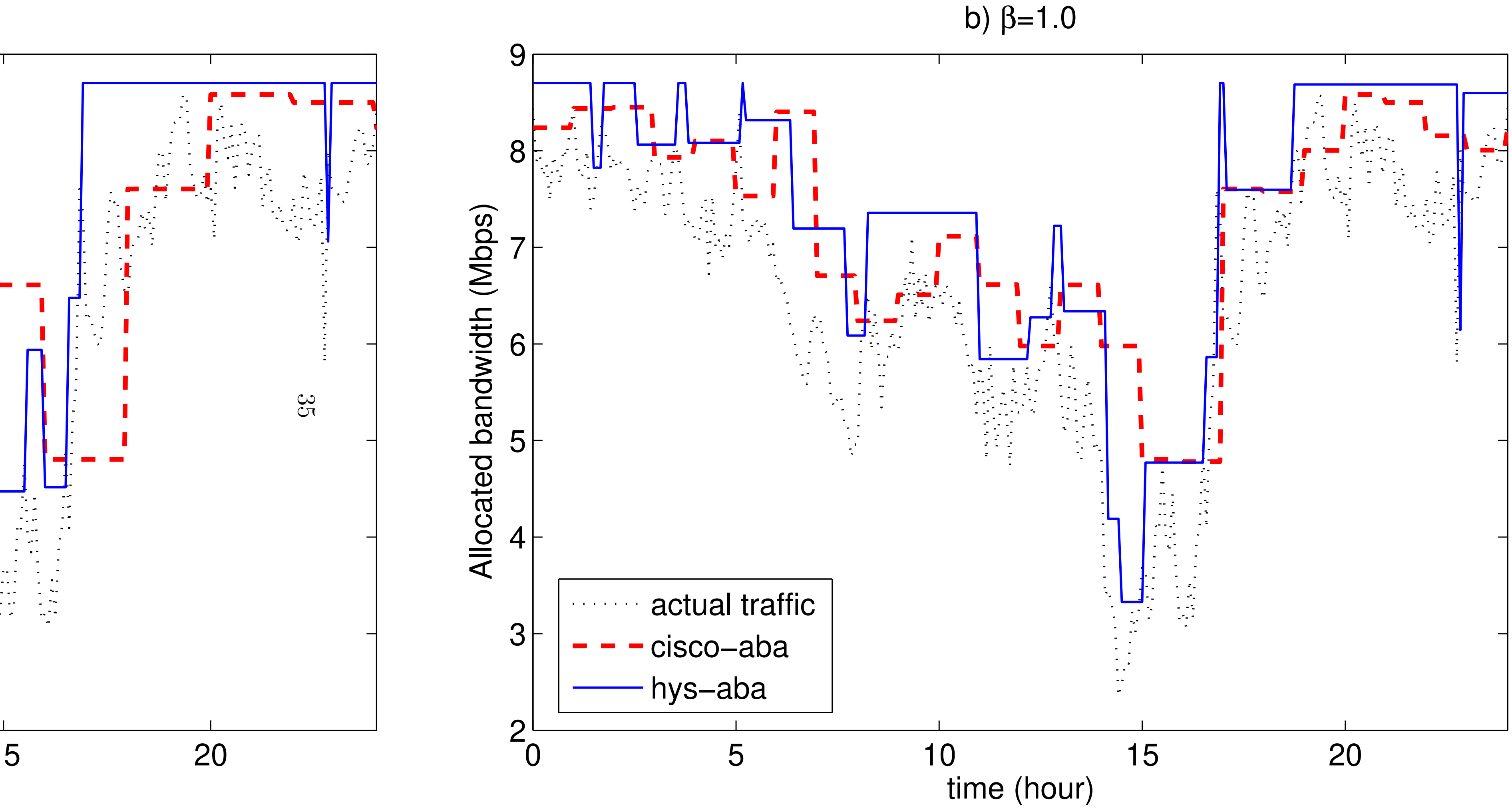
[23] C. Bruni, P. D'Andrea, U. Mocci, C. Scoglio, Optimal capacity management of virtual paths in ATM networks, in: IEEE Globecom, vol. 1, San Francisco, 1994, pp. 207–211.

[24] H. Tijms, A first course in stochastic models, John Wiley and Sons, 2003.

[25] K. Cho, K. Mitsuya, A. Kato, Traffic data repository maintained by the MAWI Working Group of the WIDE Project.
URL `http://mawi.wide.ad.jp/mawi`

[26] T. Neame, Characterisation and modelling of Internet traffic streams, Ph.D. thesis, University of Melbourne (February 2003).

[27] M. Menth, Efficient admission control and routing in resilient communication networks, Ph.D. thesis, University of Wurzburg (July 2004).


b) β=1.0
Allocated bandwidth (Mbps)
time (hour)
actual traffic
cisco−aba
hys−aba

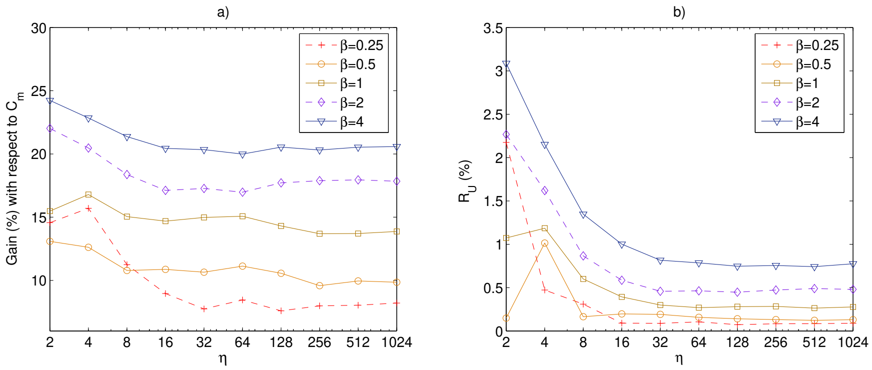


Figure 16: Bandwidth gain and under-provisioning ratio R_U for the traffic trace with varying η when C_m is fixed at 8.7 Mbps.

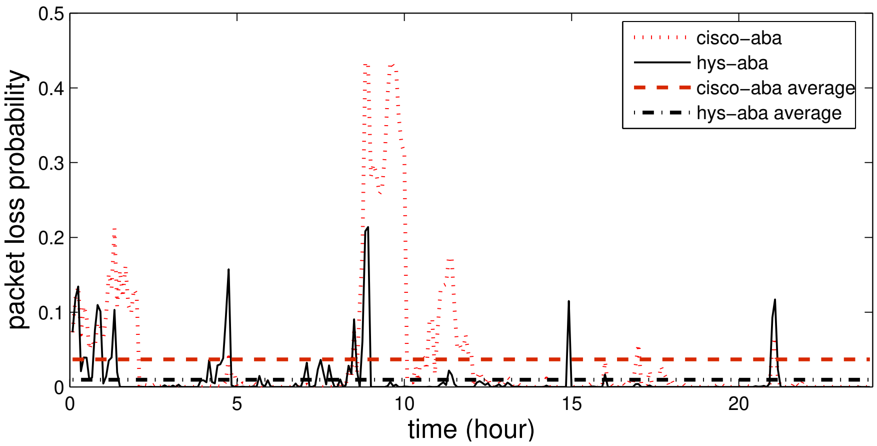


Figure 17: Instantaneous and average packet loss rates for the *cisco-aba* or *hys-aba* algorithms for the real traffic trace when $\beta = 0.5$ and $\eta = 32$.