RC 20331 (01/11/96)
Computer Science/Mathematics

Research Report

An Adaptive Virtual Path Allocation Policy for Broadband Networks

Ariel Orda

Department of Electrical Eng.
Technion, Haifa 32000
Israel

Giovanni Pacifici, Dimitrios Pendarakis

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598
USA

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

An Adaptive Virtual Path Allocation Policy for Broadband Networks

Ariel Orda
Department of Electrical Eng.
Technion, Haifa 32000
Israel
ariel@ee.technion.ac.il

Giovanni Pacifici and Dimitrios Pendarakis
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
USA
{giovanni,dimitris}@watson.ibm.com

ABSTRACT

We propose a new policy for virtual path bandwidth allocation in broadband networks. Based on a threshold scheme, our policy handles the inherent tradeoff between bandwidth utilization and processing costs. In each virtual path controller the thresholds are chosen so as to keep bandwidth utilization high, while obtaining a low rate of processing requests. Two novel ideas are used in our threshold scheme: *adaptivity*, which results in a better prediction of future bandwidth requirements; and *hysteresis*, which prevents excessive processing of requests due to oscillations around thresholds. We tested the performance of our new bandwidth control scheme, and compared it with previously suggested schemes. The performance measures were the expected amount of unused bandwidth, the average signaling load and the blocking probability. Our analysis is based on a *time segmentation* technique, which allows us to reduce a Markov chain with NM states into M Markov chains with N states and a one-dimensional chain with M states. Our results show that our policy significantly improves upon previously suggested approaches.

To appear in the Proceedings of IEEE INFOCOM, San Francisco, CA.
April 1995

1. Introduction

Accommodating calls in a broadband network requires two different types of resources, namely link bandwidth and processing power. The link bandwidth is necessary to guarantee the quality of service associated with each virtual circuit (VC), while the processing power is required for controlling and managing the VC, e.g., route selection, bandwidth reservation and configuring network devices. Since the amount of used processing resources increases with the frequency of requests for call set–up and tear–down, high call arrival intensities can lead to excessive call set up delays and result in a reduction in throughput [1].

In an attempt to both reduce network access delays and eliminate redundant processing at intermediate nodes, many researchers have suggested using the concept of *virtual path* (VP) [2, 3, 4, 5]. Bandwidth for a VP is reserved in chunks and used to satisfy the demands of several calls. Since VP set–up and tear–down requests are less frequent than VC connection requests, the VP approach reduces the computational load on the network. The reduction in processing load comes at the expense of a decreased bandwidth utilization, since the reserved bandwidth might not be fully utilized. In general, the larger the chunks of bandwidth are, the lower the processing load is and, conversely, the smaller the chunks are, the higher the bandwidth utilization is.

The objective of our work is to find a VP bandwidth allocation policy that takes into account both *bandwidth and processing resources*, by *dynamically* adjusting to the changes in the VP occupancy and by reducing the number of allocation requests. Intuitively, such a policy will lie between the VC policy (in which the reserved bandwidth is adjusted every time a new called arrives or leaves the system), and the *static* VP policy (in which bandwidth is reserved statically and its size is fixed over a relatively long time interval).

One of the most notable VP bandwidth allocation policies was proposed in [3]. According to that policy, the bandwidth of the VP changes in fixed amounts: each time the current bandwidth cannot accommodate a new call, the bandwidth is increased by a constant "step"; conversely, when the amount of used bandwidth drops back to the original amount (at which an increase of bandwidth was performed), the amount of bandwidth is decreased, by the same constant step. In other words, the scheme of [3] traces equally spaced thresholds in the VP state space, where the difference between two successive thresholds is determined by the fixed step. When the number of active calls crosses a threshold from below, bandwidth is increased to the value of the next threshold; when the number of calls crosses a threshold from above, bandwidth is decreased to the value of the crossed threshold.

The above scheme has a clear advantage over both the VC and the static VP policies, since it decreases the processing load without wasting a large amount of bandwidth. This scheme, however, has two considerable drawbacks. First, when the number of active calls grows, the rate at which calls leave the VP increases and therefore the probability of releasing bandwidth increases. This means that, when the number of active calls is high, only smaller amounts of extra bandwidth should be reserved for the VP. An allocation policy with a constant allocation step will waste more bandwidth as the number of active calls grows. Second, oscillations around a threshold state can occur resulting in an increase in the signaling load.

In this paper, we propose a new virtual path allocation policy that overcomes the above problems. We subdivide the VP state into a set of *allocation states,* and for each allocation state we define two thresholds, *viz.*, an upper threshold and a lower threshold. Each time the VP state crosses one of the two thresholds, bandwidth is either allocated or released and a new pair of thresholds is recomputed. Our policy is *adaptive* because the thresholds are recomputed by taking into account the expected call arrival and departure rates in each state of the VP. Furthermore, by keeping two separate thresholds, our policy introduces an *hysteresis* in the allocation state changes, which reduces the possibility of oscillations. Each virtual path controller computes its allocation policy independently from the other VPs, and, for each allocation state the pair of thresholds is computed independently of other allocation states. The thresholds are calculated so as to minimize a cost function based on the bandwidth and processing cost. In order to compute bandwidth and processing cost, we employ a *time segmentation* technique, which allows us to model with a Markov chain the time spent in each allocation state, while isolating it from other allocation states.

We compared the performance of our bandwidth allocation policy with the scheme proposed in [3]. We calculated the expected amount of unused bandwidth, the average signaling load and the blocking probability. Our results clearly show that our policy significantly improves upon previously suggested approaches.

This paper is organized as follows. In Sec. 2 we present our model and formulate the problem. The dynamic bandwidth allocation policy is described in Sec. 3. Section 4 presents performance results, which were derived by numerical analysis and supported by simulations, and compares our approach to previously suggested policies. Section 5 concludes the paper and presents some directions for future research.

2. Model and Problem Formulation

To keep our study focused, we will limit our attention to a single physical link that is being shared by a number of VPs.

As shown in Fig. 1, a *link controller*, associated with each physical link, collects requests for bandwidth allocation and release. The link controller must guarantee that the bandwidth allocated to the active VPs does not exceed the total link bandwidth, *i.e.* , $\sum_{j=1}^{N} B^j \leq C$, where C is the total link bandwidth, B^j is the bandwidth allocated to the j–th VP and N denotes the number of active VPs. We assume that all calls are identical and each call consumes a unit of bandwidth. Therefore C represents the maximum number of calls that can be accommodate in the link at any given time.

Two different mechanisms regulate the activities of each VP — namely, a VP admission controller and a VP bandwidth controller. The task of the *VP admission controller* is to accept or reject calls in order to guarantee that the number of active calls n^i in the i-th VP does not exceed the bandwidth allocated to that VP, *i.e.* , $n^i \leq B^i$. The *VP bandwidth controller* is responsible for managing the allocated bandwidth B^i. In order to perform this task, it continuously monitors the VP state and the call arrival process, and depending on its reservation policy, transmits allocation request and release messages to the link controller. Upon receiving the response from the link controller, the VP bandwidth controller updates the value of the bandwidth allocated to the VP. Note that the response time of the link controller is a function of the propagation delay between the two controllers, and the queuing and processing delay in the link controller. Thus, by reducing the rate of allocation requests from the link controller, one achieves smaller response times from the controller, resulting in

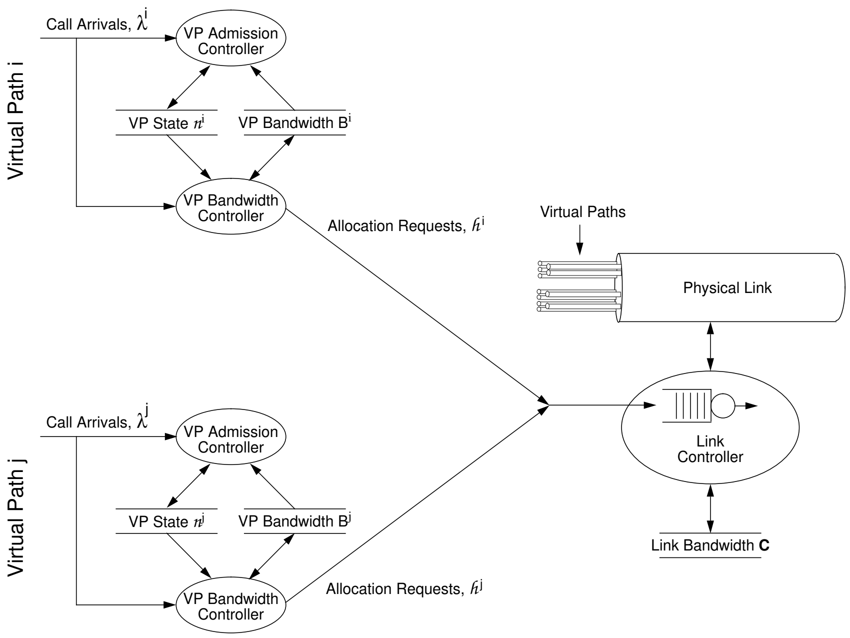

Figure 1: The virtual path allocation model

smaller set up delays and increased throughput.

The VP admission controller and the VP bandwidth controller operate either *asynchronously* or *synchronously*. In the *asynchronous* mode, a call is rejected (*i.e.* , blocked) by the VP admission controller when the link state is equal to the currently allocated bandwidth (*i.e.* , $n^i = B^i$). In the *synchronous* mode, if a call arrives while $n^i = B^i$, the VP bandwidth controller attempts to increase the allocated bandwidth, while the call request is queued in the admission controller. If the attempt fails (because either the link controller denies the request or the response is not received within a preset timeout), the call is blocked. Asynchronous operation reduces the implementation complexity of the VP controllers, while synchronous operation achieves a higher bandwidth utilization.

In the next section, we will present a new bandwidth allocation policy, derived assuming synchronous operations between VP admission and VP bandwidth controllers. The aim of our policy is to maintain a low rate of allocation requests, while utilizing the allocated bandwidth efficiently. The policy is adaptive in the sense that its decisions, for the j-th VP, depend on its state, *i.e.* , number of calls n^j. The dependency on the number of calls enables a better estimation of the future rates of call arrival and call termination.

We assume that call arrival to each VP is a Poisson process, with rate λ^j for the j–th VP. We further assume that call departure is an exponential process, with rate μ^j. The allocation policy is formulated under the assumption that allocation requests are responded immediately.

3. The Dynamic Bandwidth Allocation Policy

Our allocation policy consists of maintaining, for each VP, two time–varying values, termed *upper* and *lower thresholds*. The upper threshold represents the current amount of bandwidth reserved for the VP. Each time the current bandwidth allocation level cannot accommodate a new call, additional bandwidth is requested, and a new pair of thresholds is recomputed. The difference between upper and lower threshold represents the maximum amount of unused bandwidth (ie, the amount of reserved bandwidth that is not used by any call) that can be tolerated before releasing some of the bandwidth. When the number of active calls drops below the lower threshold, some bandwidth is released and a new pair of thresholds is recomputed.

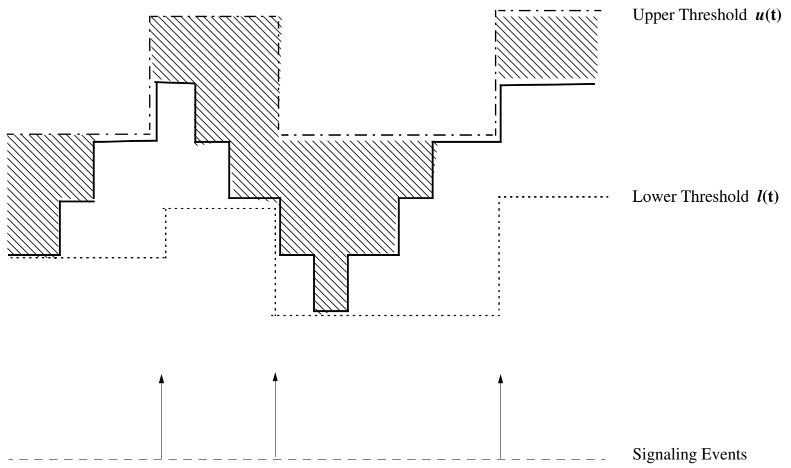

Figure 2: The dynamic bandwidth allocation policy

In Fig. 2 we show a prototypical sample path of the number of calls, $n(t)$, and the upper and lower thresholds, $u(t)$ and $l(t)$, for a single VP. The shaded area represents the amount of unused bandwidth. When either the lower or upper threshold is reached, bandwidth is released or reserved and a new set of thresholds is calculated. We define by *allocation state* the number of active calls in a VP when the upper and lower thresholds were last calculated (i.e., when bandwidth was released or reserved). If $T^j = \{0, 1, \cdots, C\}$ denotes the set of all possible allocation states for VP j, then for each $t^j \in T^j$, an upper and lower threshold are defined, $u^j(t^j) \geq t^j$ and $l^j(t^j) \leq t^j$, respectively.

The thresholds should be calculated so as to increase the bandwidth utilization of the VP while reducing the processing load. In our proposed policy, each VP j tries to minimize a cost function J^j,

$$J^j = H^j + \kappa^j \cdot U^j \tag{3.1}$$

where H^j accounts for the processing cost and U^j for the cost of wasting bandwidth. More precisely, H^j estimates the average "hitting rate", i.e., the average rate at which VP j

changes its size, U^j estimates the average amount of bandwidth wasted by VP j, and κ^j is a "normalizing" factor between the two costs.

Note that the cost function J^j does not explicitly penalize for blocking, i.e., for the cases in which a VP cannot accommodate a call since the total amount of reserved bandwidth (by all VPs) has reached the maximal value C, where C is the link capacity. However, a policy in which calls are blocked on VP j while there is still some available bandwidth on VP i, would be penalized by the cost function J^j since such a policy would be wasting bandwidth in VP i. In other words, by incurring costs on wastage of bandwidth policies that create "blocking-prone" scenarios are discarded.

Denoting by N the number of VPs sharing a link of capacity C, we observe that $2NC$ thresholds need to be specified to fully determine the policy. This number is prohibitively large to allow a global optimization of the value of all thresholds. In order to reduce the complexity of our calculation, we first assume that each VP controller computes its allocation policy (*i.e.* , thresholds) independently from other VPs; then, for each VP, we consider each allocation state separately, and calculate the optimal pair of thresholds *independently* of other allocation states.

In other words, the values of H^j, U^j are computed ignoring the interaction among various VPs, an interaction that inherently results from the finiteness of available bandwidth, and J^i is minimized by taking into account only one set of thresholds at the time, *i.e.* , we employ a local optimization process.

Although the threshold values of our policy are derived through a local optimization approach, and thus potentially inferior to those that could have been obtained by a (hypothetical) global optimization, the numerical computations presented in Sec. 4.2 demonstrate that such sets of thresholds already achieve a significant improvement over former approaches.

3.1 Modeling the VP Dynamics

In order to compute our VP allocation policy we need to express the bandwidth cost U^j and the hitting rate H^j as a function of the allocation states t^j and the upper and lower thresholds $u^j(t^j)$ and $l^j(t^j)$. To achieve this, we use a Markov chain that describes the evolution of the VP state and its transition rates. Given the equilibrium probabilities of this Markov chain, we can calculate H^j as the rate at which the Markov chain crosses thresholds and U^j as the expected value of the difference between the upper threshold and the number of calls in the VP.

In order to completely describe the state of an individual VP, three parameters have to be specified at any time: the number of calls in the VP, and the upper and lower thresholds. Since both thresholds are a function of the allocation state, the state of a VP j is given by the pair (t^j, n^j), where n^j is the number of calls in VP j and t^j is the current allocation state. At state (t^j, n^j), the bandwidth reserved in VP j is $u^j(t^j)$. The aggregate state of the Markov chain is given by the combination of states of the N VPs, *i.e.* it is the 2N-tuple

$$(t^1, n^1, t^2, n^2, \ldots, t^N, n^N).$$

The transition rates $q(\cdot, \cdot)$ of the Markov chain are then as follows:

$$q\{(t^1, n^1, \ldots, t^j, n^j, \ldots, t^N, n^N), (t^1, n^1, \ldots, t^j, n^j + 1, \ldots, t^N, n^N)\} = \lambda^j, \ \forall\, n^j < u^j(t^j), \quad (3.2)$$

i.e. , when a new call arrival can be accommodated by the current amount of bandwidth reserved;

$$q\{(t^1, n^1, \ldots, t^j, n^j, \ldots, t^N, n^N), (t^1, n^1, \ldots, t^j, n^j - 1, \ldots, t^N, n^N)\} = n^j \mu^j, \ \forall\ n^j > l^j(t^j) \quad (3.3)$$

i.e. , when a call departure does not violate the lower threshold;

$$\begin{aligned} &q\{(t^1, n^1, \ldots, t^j, u^j(t^j), \ldots, t^N, n^N), (t^1, n^1, \ldots, u^j(t^j) + 1, u^j(t^j) + 1, \ldots, t^N, n^N)\} \\ &\qquad = \begin{cases} \lambda^j & \text{if } \sum_{m=1,\ m \neq j}^{N} u^j(t^m) + u^j(u^j(t^j) + 1) \leq C \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (3.4)$$

i.e. , when a call arrival results in the upper threshold being exceeded; in this case, if there is available bandwidth the call is accepted and the new allocation state equals the previous upper threshold plus 1;

$$\begin{aligned} &q\{(t^1, n^1, \ldots, t^j, l^j(t^j), \ldots, t^N, n^N), (t^1, n^1, \ldots, l^j(t^j) - 1, l^j(t^j) - 1, \ldots, t^N, n^N)\} \\ &\qquad = l^j(t^j)\mu^j, \text{ if } l^j(t^j) > 0, \end{aligned} \quad (3.5)$$

i.e. , when a call departure results in a drop below the lower threshold; the new allocation state equals the previous lower threshold minus 1.

All other transition rates are equal to 0. By using the above expressions for the transition rates in the corresponding equilibrium equations, one can compute the steady-state probabilities of the above Markov chain.

A building block of this Markov chain for $N = 1$ is shown in Fig. 3. Each "row" of this chain corresponds to one allocation state. Observe that the rows shown are not "consecutive" and that *transitions into a different row occur only into the allocation state of the new row.* This is a key property[1] that will allow us to employ a *time segmentation* technique to simplify the two dimensional Markov chain.

3.2 Calculation of Thresholds — The Time Segmentation Principle

Since the structure of the Markov chain described in the previous section exhibits dependencies among different allocation states, it cannot be directly used to compute H^j and U^j. In fact, as explained in Sec. 3, minimizing the objective function in eq. 3.1 under this condition is prohibitively complex.

In this section we introduce a novel technique, viz. the *time segmentation technique*, that allows us to model directly the evolution of the VP state for each allocation state, when viewed in isolation from other allocation states. We use the time segmentation technique to construct an *auxiliary* Markov chain that describes the evolution of the VP state for each allocation state. The values of the H^j and U^j, corresponding to a particular allocation state and a set of thresholds, are computed through this auxiliary chain. We repeat the calculation for all possible pairs of lower and upper thresholds, and then we select the pair of thresholds that minimize the objective function in eq. 3.1 for each given allocation state.

In the following we will focus on a single VP and, for notation simplicity, we will drop the subscript denoting the VP index from the allocation state and the upper and lower thresholds.

[1]Note that this property only holds for the single VP case.

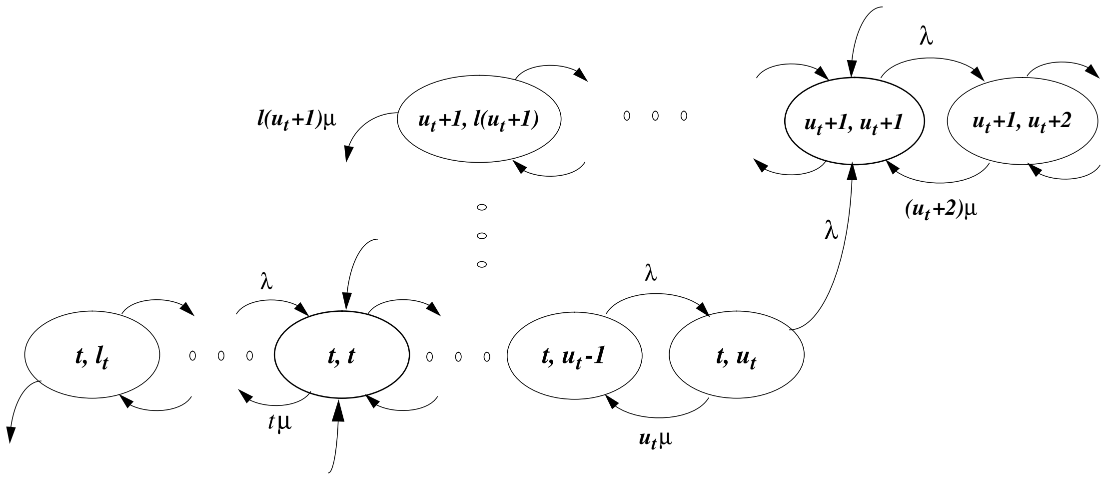

Figure 3: Building block of 2-dimensional Markov chain

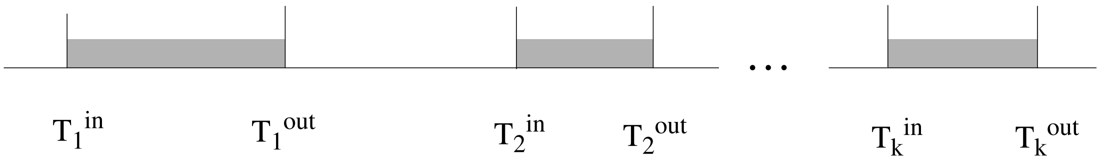

Figure 4: Time segmentation

Consider some allocation state t_i. Since the Markov chain is irreducible, the allocation state t_i will be visited infinitely often. Denote by T_n^{in} and T_n^{out}, $n = 1, 2, \ldots$, the time that the VP enters and exits, respectively, allocation state t_i for the n-th time. In order to calculate the average unused bandwidth for allocation state t_i, we observe the state of the VP during the interval $\bigcup_{n=1}^{\infty}[T_n^{\text{in}}, T_n^{\text{out}}]$ (see Fig. 4). At any exit time T_n^{out}, a change in the allocation state occurs, because the number of calls either exceeds the upper threshold or becomes less than the lower threshold of the current allocation state t_i. In both cases, at the next entry time, T_n^{in}, the number of calls will be t_i, since as mentioned Sec. 3.1 transitions from one "row" of the 2-dimensional Markov chain to another occur only into the allocation state of the new row. Thus, by observing the state of the VP during the time interval $\bigcup_{n=1}^{\infty}[T_n^{\text{in}}, T_n^{\text{out}}]$, we obtain the auxiliary Markov chain depicted in Fig. 5.

Using balance equations, in the Appendix it is shown that the steady-state probabilities corresponding to the auxiliary Markov chain of Fig. 5 obey the following recursive expressions:

$$P_{l+j} = \frac{\sum_{i=0}^{j}(\Pi_{n=0}^{j-1-i}(l+n))\lambda^i\mu^{j-i}}{(\Pi_{n=1}^{j}(l+n))\mu^j}P_l, \quad j = 1, 2, \cdots, k-l \tag{3.6}$$

$$P_{u-j} = \frac{\sum_{i=0}^{j}\Pi_{n=1}^{j-i}(u-j+n)\lambda^i\mu^{j-i}}{\lambda^j}P_u, \quad j = 0, 1, u-k\ . \tag{3.7}$$

(3.6) and (3.7) express all equilibrium probabilities in terms of P_l and P_u. These terms can

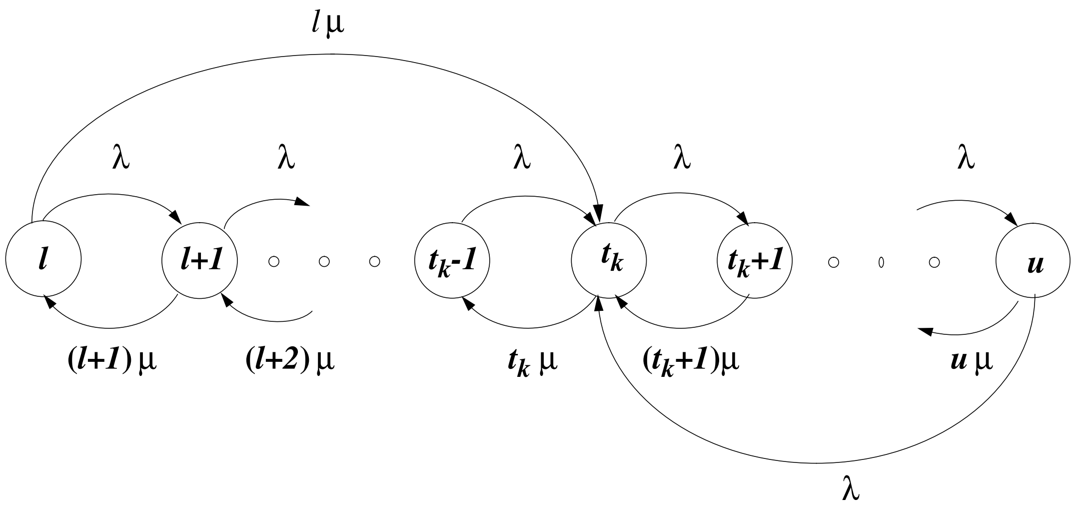

Figure 5: Equivalent representation of auxiliary Markov chain

be obtained through the following balance equation:

$$P_k \cdot (\lambda + k\mu) = P_l \cdot l\mu + P_{k-1} \cdot \lambda + P_{k+1} \cdot (k+1)\mu + P_u \cdot \lambda \ , \tag{3.8}$$

together with:

$$\sum_{i=0}^{u-l} \sum_{i=0}^{u-l} P_{l+i} = 1 \ . \tag{3.9}$$

The hitting rate $H(t_i, l, u)$, corresponding to allocation state t_i and to the thresholds l and u, is, then:

$$H(t_i, l, u) = P_u \cdot \lambda + P_l \cdot l \cdot \mu \tag{3.10}$$

Similarly, the average unused bandwidth $U(t_i, l, u)$ is:

$$U(t_i, l, u) = \sum_{k=l}^{u} P_k \cdot (u - k) \tag{3.11}$$

The cost $J(t_i, l, u)$, corresponding to allocation state t_i and to the thresholds l and u is

$$J(t_i, l, u) = H(t_i, l, u) + \kappa \cdot U(t_i, l, u) \ . \tag{3.12}$$

Thus, the pair of thresholds $l(t_i)$ and $u(t_i)$, assigned to allocation state t_i, is such that:

$$J(t_i, l(t_i), u(t_i)) \leq J(t_i, l, u) \ \ \forall l, u \ . \tag{3.13}$$

To summarize, the computation of thresholds for a single VP is performed as follows:

1. $n := 0$.

2. Considering state n as the allocation state, apply the following for all possible pairs of thresholds l and u:

(a) Construct the auxiliary Markov chain (Fig. 5) corresponding to $t_k = n$, l and u.

(b) Compute the equilibrium probabilities P_i, using (3.6-3.9).

(c) Compute $H(t_i, l, u)$, $U(t_i, l, u)$ and $J(t_i, l, u)$, using (3.10-3.12).

3. Choose $l(t_i)$ and $u(t_i)$, through (3.13). If $n = C$ then *stop*. Else, $n := n + 1$ and *go to Step 2*.

We note that the above computation is performed off-line, thus its number of iterations, of order $O(C^3)$, is manageable. In the Conclusions we discuss several schemes for further reducing the computation complexity.

4. Performance Evaluation

We tested the performance of our new bandwidth control scheme, and compared it with previously suggested schemes. The performance measures were the expected amount of unused bandwidth, the average hitting rate and the blocking probability. Performance has been evaluated through numerical computations as well as by simulations.

4.1 Numerical Computations

Calculation of the performance measures, namely unused bandwidth, hitting rate and blocking probability, amounts to calculating the steady-state probabilities

$$P(t^1_{i_1}, n^1, t^2_{i_2}, n^2, \cdots, t^j_{i_j}, n^j, \cdots t^N_{i_N}, n^N),$$

for all VPs $1 \leq j \leq N$, transition states $0 \leq t^j_{i_j} \leq C$ and number of calls $0 \leq n^j \leq C$.

Each VP can be in any of $O(SC)$ states, where S is the average number of states corresponding to each transition state (*i.e.* the average distance between upper and lower bounds) and the number of transition states is C. For N VPs, the size of the resulting Markov chain is $O(S^N \cdot C^N)$, which is definitely prohibitive for large N. Therefore, we apply an approximation, based on an aggregation principle. Specifically, for each VP j, $1 \leq j \leq N$, all the states corresponding to the same transition state are aggregated. Thus, the state of the new Markov chain is $(t^1_{i_1}, \cdots t^j_{i_j}, \cdots t^N_{i_N})$, i.e. all VPs are represented by their transition state only. The transition rates between states of this chain are derived from the auxiliary Markov chain of each transition state. Hence, the transition rate from state $(t^1_{i_1}, \cdots t^j_{i_j}, \cdots t^N_{i_N})$ to state $(t^1_{i_1}, \cdots, u^j(u^j(t^j_{i_j}) + 1), \cdots t^N_{i_N})$ (correspondingly, to state $(t^1_{i_1}, \cdots, l^j(t^j_{i_j}) - 1, \cdots t^N_{i_N})$) is taken to be the flow between states $t^j_{i_j}$ and $u^j(u^j(t^j_{i_j}) + 1)$ (correspondingly, between $t^j_{i_j}$ and $l^j(t^j_{i_j}) - 1$) for a single VP. However, in this case we differentiate among states of this Markov chain which are blocking for some VP and those which are not. Specifically, if $\sum_{k \neq j} u^k(t^k_{i_k}) + u^j(u^j(t^j_{i_j}) + 1) > C$, then state $(t^1_{i_1}, \cdots t^j_{i_j}, \cdots t^N_{i_N})$ is blocking for VP j; therefore, the transition upwards is not allowed in this state for VP j. In this case, the auxiliary chain of Fig. 5 is modified so that the transition rate between u and t_k is set to 0 instead of λ. Finally, we numerically compute the steady-state probabilities of the global chain, from which the performance measures can be derived.

We note that in the case of a single VP, this Markov chain models exactly the behavior of the system, due to the separation principle described in section 3.2. Furthermore, in the multiple VP case the approximation error is incurred only at boundary states, where blocking applies. Numerical tests as well as simulation support the efficiency of this approximation.

Overall, we make computations on $O(N \cdot C)$ auxiliary chains, each of size $O(S)$, and on a global chain of size $O(C^N)$. Although the problem size is still exponential, the above approximation results with a considerable reduction over the initial $O(S^N \cdot C^N)$ problem size. In practice, the approximation enabled us to perform numerical computations for a few VPs.

4.2 Results

In our first experiment, we studied the tradeoff between bandwidth reservation, i.e., average unused bandwidth, and signaling load, i.e., hitting rate. In our cost function, the relative importance of each of the two is determined by the bandwidth cost coefficient κ. Thus, by changing the value of κ we can produce a spectrum of possible operating points for the above tradeoff. In this experiment, we performed the numerical calculations for a link with bandwidth $C = 155$ with a single active VP. The VP call arrival and departure rates were set to $\lambda = 100$ and $\mu = 1$. For this scenario, we tested a whole range of values for the κ coefficient. For each value of κ, the thresholds were computed as described in Sec. 3.2, and then the unused bandwidth and hitting rate were computed.

The results of this experiment are reported in Fig. 6. Figures 6(a) and (b) show the impact of the parameter κ on the signaling load (hitting rates) and on the unused bandwidth. As expected, when the values of κ is high the amount of unused bandwidth is low, since, higher values of κ produce more conservative policies in terms of bandwidth consumption, i.e., the upper threshold is closer to the current number of active calls. However, under this condition, the thresholds will be reached more frequently and therefore the signaling load will increase.

We note that when $\kappa \to \infty$, the adaptive policy approaches the traditional "pure" VC policy, and, therefore, the amount of bandwidth reserved and not used (bandwidth wastage) is zero. This comes at the expense of a steep increase in the signaling load. Figures 6(a) and (b) clearly show the advantage of our adaptive bandwidth allocation scheme. Indeed, large ($\sim 95\%$) reductions in the signaling load can be achieved with just a small increase ($\sim 3\%$) in the unused bandwidth.

We compared our adaptive policy with the *fixed step* policy proposed in [3]. Recall that in the fixed step policy, the difference between two successive thresholds is kept constant, therefore the bandwidth is reserved and released in a fixed step of size S. Different values of the step size S correspond to different amounts of unused bandwidth and signaling load pairs. In Fig. 6(c) we compare the signaling load and the unused bandwidth for the fixed step and the adaptive policies. Each point in the plot of Fig 6(c) was obtained varying the values of S for the fixed step policy and κ for the adaptive policy. As shown in Fig. 6(c), the adaptive policy performs better, since it incurs less signaling load for the same amount of unused bandwidth (or, conversely, wastes less bandwidth for the same amount of signaling load). In particular, the difference in performance is more pronounced for high values of unused bandwidth. This performance increase can be intuitively explained by the fact that

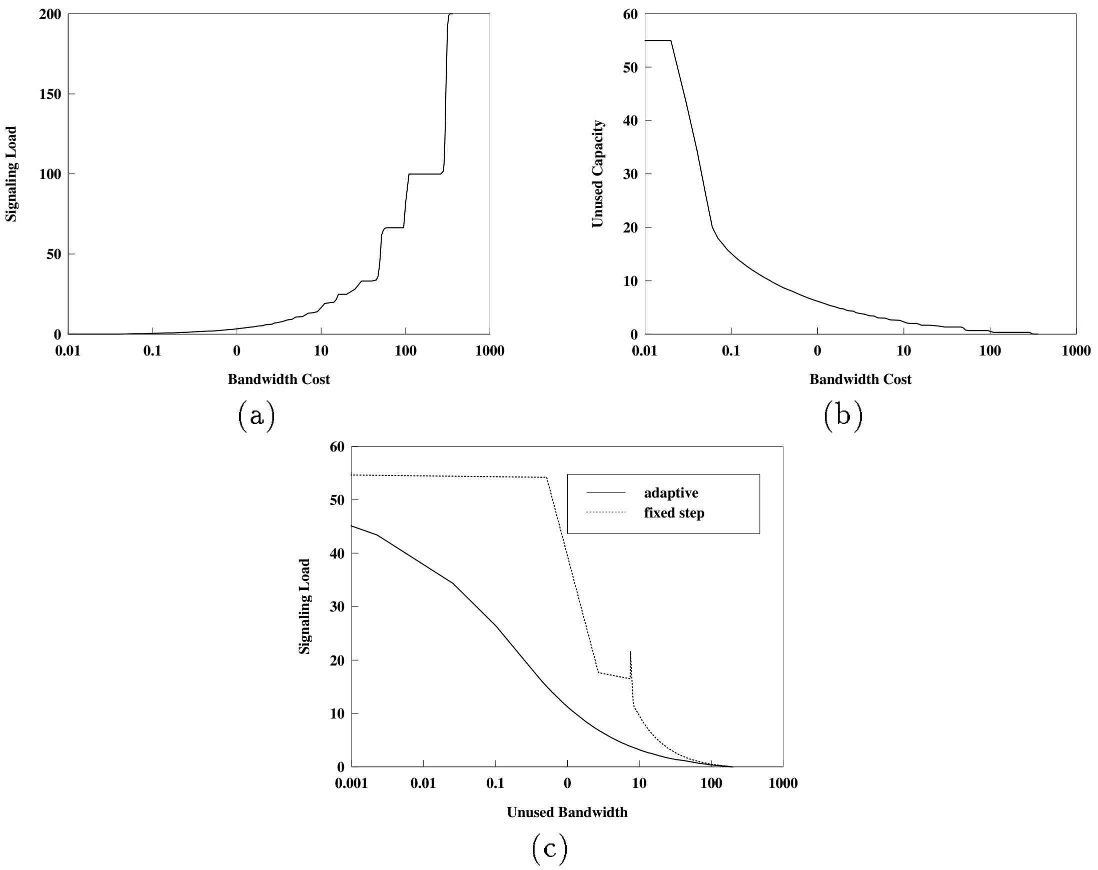

Figure 6: Tradeoff between bandwidth reservation and signaling load with $C = 155$, $\lambda = 100$ and $\mu = 1$: (a) signaling load for the adaptive scheme; (b) unused bandwidth for the adaptive scheme; (c) signaling load of adaptive and fixed step schemes

the adaptive policy reserves different amounts of bandwidth when the VP is in different states, thus, taking into account the higher departure rate of calls when the number of active calls in the VP is high, while the fixed step policy always reserves the same amounts of bandwidth. Moreover, in the fixed step policy, even very large values of S cannot decrease to zero the hitting rate, due to the possible oscillations near the thresholds [3].

In our second set of experiments, we studied the performance of the bandwidth allocation policies when two VPs are multiplexed over the same link. We considered the following scenario: link bandwidth $C = 50$, call arrival rates $\lambda_1 = \alpha\lambda$, $\lambda_2 = (1-\alpha)\lambda$ and call departure rates $\mu_1 = \mu_2 = 1$. The parameter α took the values 0.5, which represents an equal loading of the two VPs and 0.1, which represents an unbalanced loading. We measured the blocking probabilities and the signaling loads for different values of λ as well as for each value of α. For the fixed step policy, we considered three different values of the step size, $S = \{2, 4, 8\}$, corresponding to high, medium and low signaling load, respectively, while, for the adaptive policy, we selected the coefficient κ that was producing the same signaling load of the fixed step approach. Performance values were then calculated by varying the aggregate offered load $\rho = \frac{\lambda}{\mu C}$.

The results for $\alpha = 0.5$ and $S = 2, 4, 8$ are shown in Fig. 7. Figure 7(d) shows that the

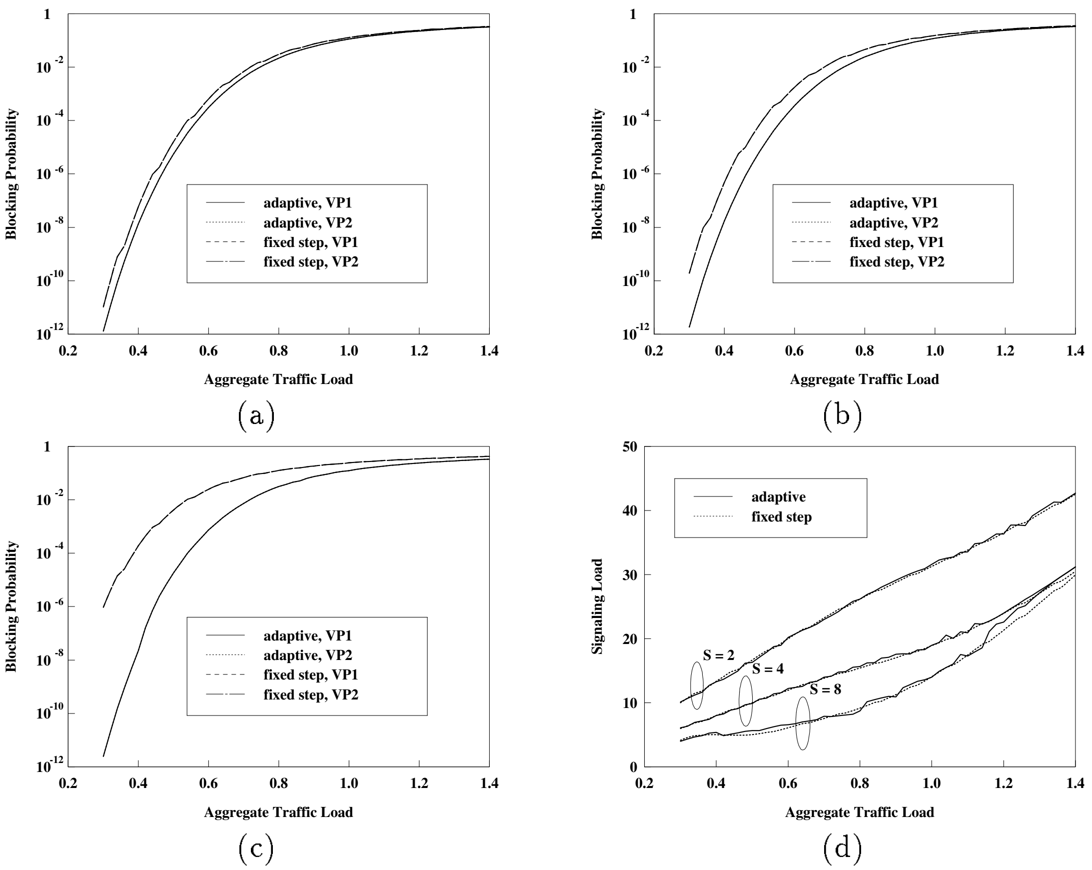

Figure 7: Blocking probabilities and signaling load for the fixed step and adaptive allocation policies with $C = 50$, $\mu = 1$ and $\alpha = 0.5$: (a) blocking probabilities for $S = 2$; (b) blocking probabilities for $S = 4$; (c) blocking probabilities for $S = 8$; (d) signaling load.

signaling load for the adaptive and fixed step policies were indeed matched for every value of the offered load ρ. In this case, both VPs have the same call arrival rates and policies, and, therefore, the call blocking probabilities are equal. As can be seen in Fig. 7(a)–(c), when the signaling load is the same, the adaptive policy has always a smaller blocking probability than the fixed step policy. The difference in blocking probability is lower for smaller values of load and lower signaling loads. Intuitively, for small values of the fixed step S, both policies approach the same pure VC scheme and therefore the difference in blocking probabilities is small. Furthermore, when the offered load is high, the capacity of the system is exceeded and no policy can prevent call loss.

The results for $\alpha = 0.1$, shown in Fig. 8, describe the differences in blocking probabilities between the two VPs under unbalanced load conditions. This difference stems from the fact that both VPs share the same value of the bandwidth cost κ, although they have different traffic loads. In order to equalize the blocking probabilities among different VPs, the value of κ can be made dependent on the traffic load of each VP. For example, a VP with a lower traffic load could use a smaller value of κ. Since κ is a real number while S is an integer, the adaptive policy allows a finer control of blocking probabilities than the fixed step approach. The impact of these can be quite substantial, as some preliminary results show.

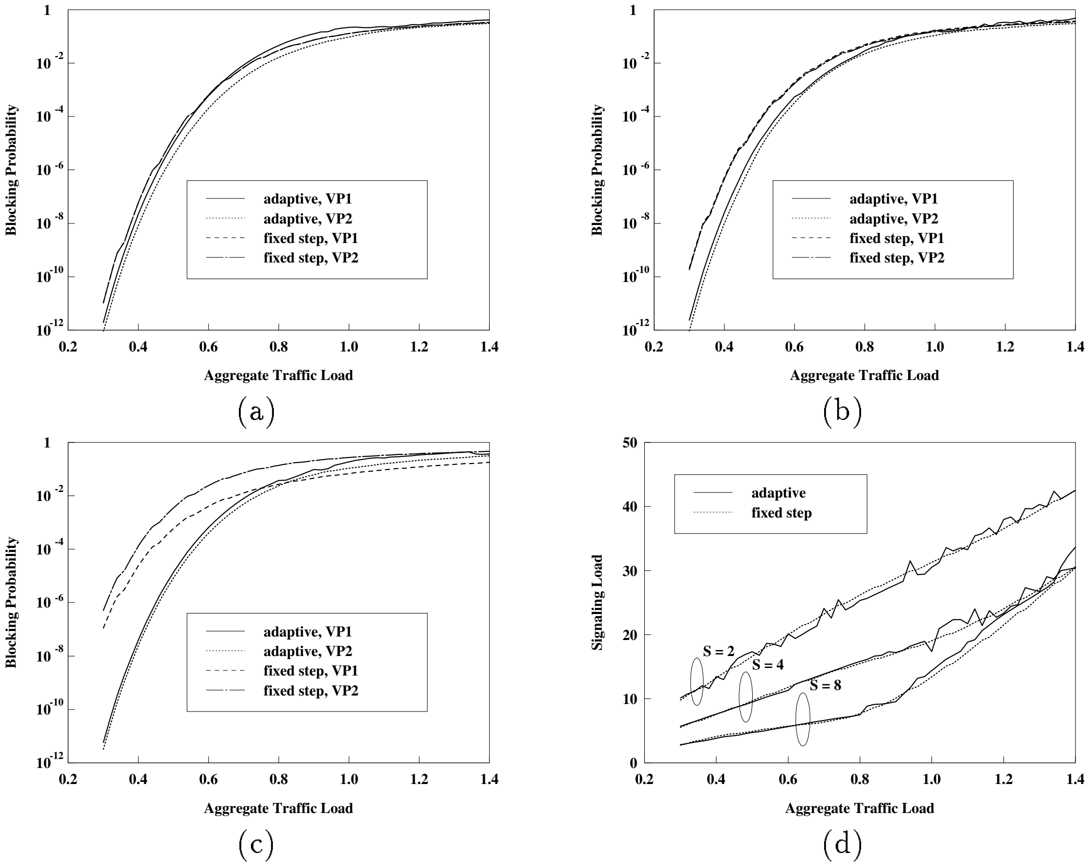

Figure 8: Blocking probabilities and signaling load for the fixed step and adaptive allocation policies with $C = 50$, $\mu = 1$ and $\alpha = 0.1$: (a) blocking probabilities for $S = 2$; (b) blocking probabilities for $S = 4$; (c) blocking probabilities for $S = 8$; (d) signaling load.

The analytical results have been verified by way of simulations. The simulator corresponds to the comprehensive model of Fig. 1. Figure 9 depicts the analytical and the simulation results, for the setting of the previous experiment. We conclude that the numerical analysis, performed according to the approximation described in Sec. 4.1, accurately evaluates the performance of our policy for the range of loads considered.

5. Conclusions

In this paper we presented a VP allocation policy that improves upon previous schemes, by employing adaptivity and hysteresis. In order to contain the computational complexity of our policy, we constructed an approximated local–search procedure which requires at most $O(C^3)$ calculations. Our policy could also be implemented using considerably less calculations. In fact, rather than computing thresholds for all possible allocation states (i.e., for all $0 \leq t_i \leq C$), one can consider only the relevant states, which are precisely the threshold states produced by the procedure (indeed, transitions can occur only at threshold states). In order to reduce computations even further, one might consider restricting *a–priori* the set of allowable allocation states. Some preliminary experimentation suggests that such

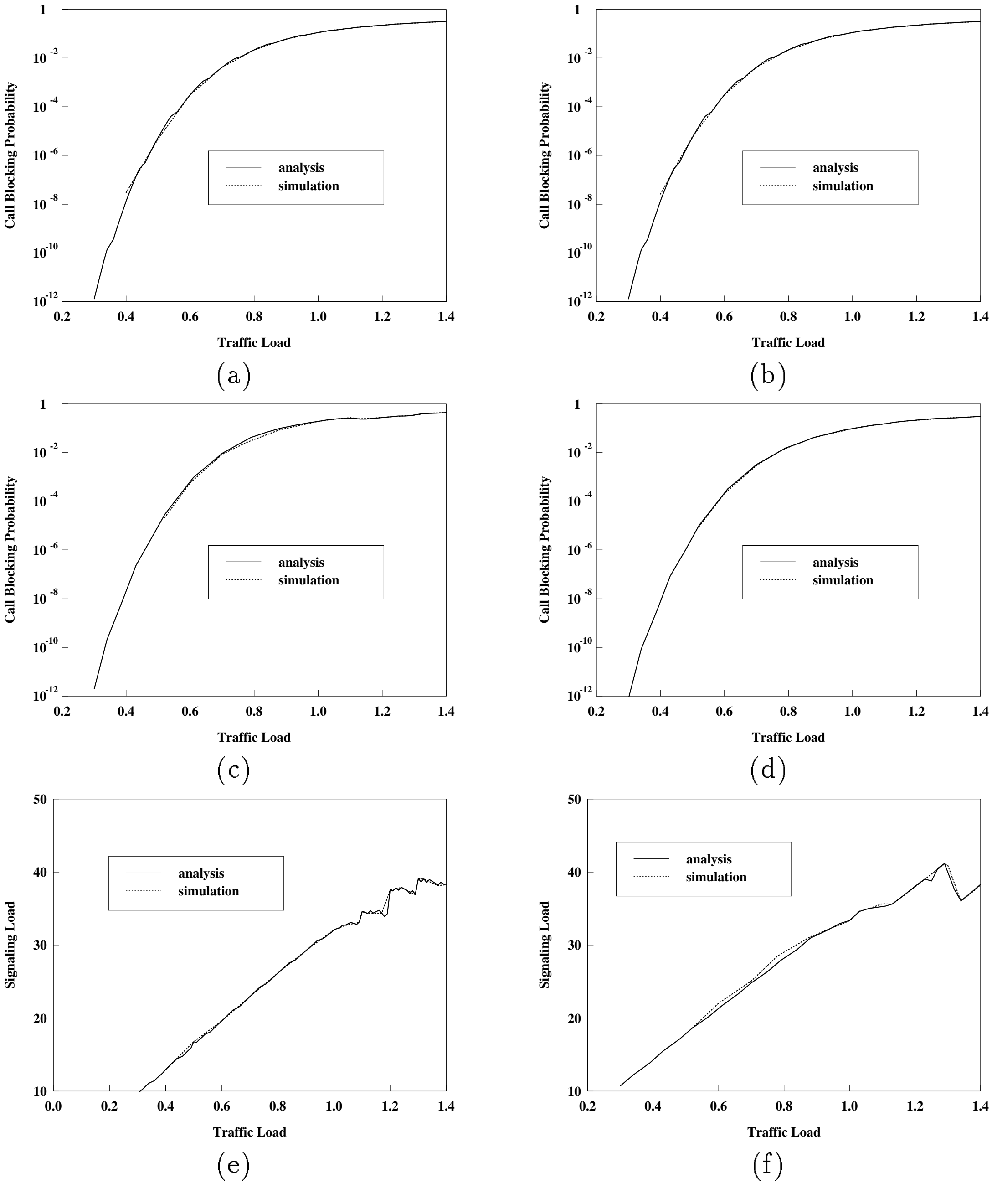

Figure 9: Blocking probabilities and signaling loads obtained with the analytical and simulation methods with $C = 50$, $\mu = 1$: (a) blocking probabilities for VP 1 for $\alpha = 0.5$; (b) blocking probabilities for VP 2 for $\alpha = 0.5$; (c) blocking probabilities for VP 1 for $\alpha = 0.1$; (d) blocking probabilities for VP 2 for $\alpha = 0.1$; (e) signaling load for $\alpha = 0.5$; (f) signaling load for $\alpha = 0.1$.

an approach typically results in a considerable decrease in the computation complexity, while incurring just a slight reduction in performance. Notwithstanding, the performance of our VP allocation policy can be further improved by using a more complex global–search procedure.

Our policy was derived for the single class case, where each allocation state t_i is associated with a single pair of thresholds (l_i, u_i). The extension to multiple classes requires to define a pair of thresholds for each class, *i.e.* characterize the VP reservation policy with an hypercube. Whenever one of the thresholds is crossed, the boundaries of the hypercube are recalculated.

Using the simulator, we are currently evaluating our policy for cases where the analysis is prohibitively complex. In particular, we are studying the performance of our policy when many VPs are multiplexed on the same link, and when the processing speed of the link controller is finite. We expect that, due to its novel threshold structure, in these cases too our policy will continue to exhibit efficient performance and offer a significant improvement upon previous approaches.

Finally, our policy could be also applied to the class of *inventory control*[6, 7] problems, in which there is a penalty for accumulating unsold goods on the one hand, and for not being able to (immediately) supply a demand on the other. However, the “goods” in our case, i.e., bandwidth, are not “sold” but are rather “hired”, since, upon termination, calls release their bandwidth. Seen in that way, our problem could resemble a problem of controlling the inventory of a rental company (e.g., car rentals). Indeed, while maintaining a stock of unhired items incurs a cost, not being able to supply a local demand requires either having to refuse it or to ship an item from another site, *i.e.* , it incurs a “processing cost”.

Appendix

In this appendix we establish the validity of (3.6) and (3.7), corresponding to Fig. 5.
Applying the balance equations we get:

$$P_l \cdot (l\mu + \lambda) = P_{l+1} \cdot (l+1)\mu \tag{.1}$$

$$P_i \cdot (i\mu + \lambda) = P_{i-1} \cdot \lambda + P_{i+1} \cdot (i+1)\mu \quad i = l+1, \cdots k-1, k+1, \cdots u-1 \tag{.2}$$

$$P_u \cdot (\lambda + u\mu) = P_{u-1} \cdot \lambda \tag{.3}$$

together with (3.8), repeated below:

$$P_k \cdot (\lambda + k\mu) = P_l \cdot l\mu + P_{k-1} \cdot \lambda + P_{k+1} \cdot (k+1)\mu + P_u \cdot \lambda$$

From (.1) we get:

$$P_{l+1} = \frac{l\mu + \lambda}{(l+1)\mu} P_l,$$

which, after substitution into (.2) for $i = l+1$, yields:

$$P_{l+2} = \frac{l(l+1)\mu^2 + l\lambda\mu + \lambda^2}{(l+1)(l+2)\mu^2} P_l$$

Continuing this way for P_{l+j}, $j = 2, 3, \cdots, k-l$, we get the following result.
Lemma: The equilibrium probabilities satisfy (3.6), repeated below, for $j = 1, 2, \cdots,, k-l$:

$$P_{l+j} = \frac{\sum_{i=0}^{j} (\Pi_{n=0}^{j-1-i}(l+n))\lambda^i \mu^{j-i}}{(\Pi_{n=1}^{j}(l+n))\mu^j} P_l$$

Proof: By induction on j. We have already shown 3.6 for $j = 1, 2$. Assuming it holds for $j-1$ and j, we show it holds for $j+1$. By (.2):

$$P_{l+j} \cdot ((l+j)\mu + \lambda) = P_{l+j-1} \cdot \lambda + P_{l+j+1} \cdot (l+j+1)\mu$$

Applying the inductive assumption on $j-1$ and j and our hypothesis on $j+1$, we get:

$$\frac{l \cdots (l+j-1)\mu^j + l \cdots (l+j-2)\mu^{j-1}\lambda + \cdots + l\lambda^{j-1}\mu + \lambda^j}{(l+1)(l+2)\cdots(l+j)\mu^j} P_l \cdot ((l+j)\mu + \lambda) =$$

$$\frac{l \cdots (l+j-2)\mu^{j-1} + l \cdots (l+j-3)\mu^{j-2}\lambda + \cdots + l\lambda^{j-2}\mu + \lambda^{j-1}}{(l+1)(l+2)\cdots(l+j-1)\mu^{j-1}} \cdot P_l \cdot \lambda +$$

$$\frac{l \cdots (l+j)\mu^{j+1} + l \cdots (l+j-1)\mu^j\lambda + \cdots + l\lambda^j\mu + \lambda^{j+1}}{(l+1)(l+2)\cdots(l+j+1)\mu^{j+1}} \cdot P_l \cdot (l+j+1)\mu$$

which holds iff:

$$(l \cdots (l+j-1)\mu^j + l \cdots (l+j-2)\mu^{j-1}\lambda + \cdots + l\lambda^{j-1}\mu + \lambda^j)((l+j)\mu + \lambda) =$$

$$(l \cdots (l+j-2)\mu^{j-1} + l \cdots (l+j-3)\mu^{j-2}\lambda + \cdots + l\lambda^{j-2}\mu + \lambda^{j-1})((l+j)\mu\lambda +$$

$$l \cdots (l+j)\mu^{j+1} + l \cdots (l+j-1)\mu^j\lambda + \cdots + l\lambda^j\mu + \lambda^{j+1}$$

which can be verified to hold, hence the inductive claim. □

Applying a similar analysis for the upper states $k, k+1, \cdots u$ yields (3.7).

References

[1] P. J. Kühn, C. D. Pack, and R. A. Skoog, "Common channel signaling networks: Past, present, future," *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 383–394, April 1994.

[2] ITU, *State Management Function — ITU–T Rec.X.731.* ITU.

[3] S. Ohta and K.-I. Sato, "Dynamic bandwidth control of the virtual path in an asynchronous transfer node network," *IEEE Transactions on Communications*, vol. 40, July 1992.

[4] S.-B. Kim, "An optimal establishment of virtual path connections for ATM networks," in *Proceedings of the IEEE INFOCOM '95*, (Boston, MA), pp. 72–79, April 1995.

[5] K.-I. Sato, S. Ohta, and I. Tokizawa, "Broad-band ATM network architecture based on virtual paths," *IEEE Transactions on Communications*, vol. 38, pp. 1212–1222, August 1990.

[6] G. Hadley and T. M. Whitin, *Analysis of Inventory Systems.* Englewood Cliffs: Prentice Hall, 1963.

[7] S. Nahmias, "Managing repairable item inventory systems: a review," in *Studies in Management Sciences*, vol. 16, pp. 253–277, 1981.