Dynamic Allocation of Resources to Virtual Path Agents

Hanoch Levy, Tsippy Mendelson and Gilad Goren

***Abstract*— One of the major problems faced in operating large networks is the enormous amount of processing and communications overhead required for setting up and tearing down the large number of connections maintained by the network. ATM and MPLS aim at solving these problems via the Virtual Path (VP) mechanism which is used to group together the connections. When a need for setting up a connection rises, the request and its resource allocation are processed by the VP agent and not by the network, thus reducing the processing cost significantly.**

An important question in the design of these networks is the amount of network resources to be dynamically allocated to and held by the VP agents; too high allocation will result with bandwidth resource waste, while too low allocation will result with heavy connection set-up and tear-down processing load.

In this paper we deal with this problem, and at deriving simple operational rules to determine the amount of bandwidth resources to be held by the various VP agents, while balancing between bandwidth waste and connection processing overhead. We formulate the resource allocation problem by accounting both for bandwidth utilization and for connection processing constraints. Recognizing the complexity of the problem, we use a decomposition approach in which we first analyze the single link problem and then propose to use this solution as a building block in constructing algorithms for the whole network. For the single link problem we realize that the pure problem is too complex and thus formulate an approximate model and derive the *optimal allocation* for it. The optimal rule is expressed as a closed-form square-root allocation. Extensive numerical examination shows that the rule proposed yields very efficient allocations. For the full network problem, we propose to capitalize on the closed form structure of single link problem solution and use it in devising algorithms for the whole network.

1 Introduction

One of the major problems faced in operating large networks is the enormous amount of processing and communications overhead required for managing the bandwidth allocation to individual connections; this results from the huge number of connections maintained by the network. These scalability problems can be solved by the Virtual Path (virtual pipe) mechanism. This mechanism has been introduced first by ATM, (Asynchronous Transfer Mode), see e.g. [1] and [12], and was then adopted and further developed for IP networks through the MPLS (Multi-Protocol-Label-Switching) mechanism. In fact, the concern from management scalability issues over IP networks and the need to provide them with proper Quality of Service (QOS) is one of the driving forces behind current efforts of network developers and operators in developing and deploying the MPLS (as well as DiffServ) technology.

H. Levy, School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel, hanoch@cs.tau.ac.il

T. Mendelson, Israel Design Center, Intel Communication Group - Jerusalem, Intel Corp., Jerusalem, Israel, tsippy.mendelson@intel.com

G. Goren, Native Networks Technologies, Ltd., Ramat Gan, Israel, gilad@nativenetworks.com

The work was done while the authors were with Tadiran Telecom. Petach-Tikva, Israel. Partial results of this work are reported in [16].

The VP is a mechanism for grouping a set of connections (individual connections are termed Virtual Connections (VCs) in the ATM language) and for handling them and manipulating them as a group. The grouping is normally based on the characteristics of the connections, e.g., all connections originating at node x and destined to node y are grouped under one VP. When a need for setting up a connection (VC) rises, the connection request and its resource allocation are processed by a *VP agent* and not by the network. This significantly reduces the number of network entities dealing with the request and therefore the associated processing complexity and cost. To facilitate this, the network must allocate and delegate, at an earlier stage, certain amount of bandwidth resources to the VP agent. To achieve high efficiency, allocation of bandwidth resources should be done *dynamically* as a function of the network state and the resources requested.

In a practical network, e.g. an IP network, virtual paths are expected to span over a number of routers. The management of the VP, including the admission of connections into the VP is expected to be handled at the ingress node of the VP (and thus to simplify the operation of core routers and reduce their processing requirements). This is likely to be an edge router in an IP network. For example, if one is to handle IP telephony traffic between Los Angeles and London, the MPLS VP may consist of a number of routers located throughout the USA and England. The management of the calls permitted to enter the VP is excepted to be carried out by a VP agent located at or nearby the edge router at Los Angeles. The routers en-route from Los-Angeles to London, including the network core routers, are expected to see a little fraction of the overhead involved in admitting calls into the VP.

The concept of using VPs in order to reduce the overhead involved with connection processing was introduced and addressed in Addie et al. [1], Burgin [5], Ohta and Sato [12] and Orda et al. [19]. Burgin recognizes that the use of virtual paths can reduce the network overhead involved in call setup and tear down. Burgin [5] provides an analysis of the bandwidth requirements of the network when (fixed-interval) periodic VP capacity assignments are conducted. Ohta and Sato [12] and Orda et al. [19] considered the trade-offs between efficient use of the bandwidth resources and that of the processing resources. In those works the connection processing resource is considered as an entity that must be minimized, and thus the analysis can be used in a design problem (e.g. answering the question of how many processing resources are needed in the network). In [12] a simple uniform distance threshold algorithm is considered. An improvement upon that work is provided in [19] by using dynamically changing thresholds with hysteresis.

In light of VPs' role as means for reducing network processing overhead, we recognize that an important operation problem is that of how to efficiently operate the VPs under the constraint of the available network processing power. This translates to the problem of how many bandwidth resources to be allocated to the VP agents (group agents) when they are in lack of such resources; too high allocation will lead to situations where VP agents hold non-utilized bandwidth resources and thus will result with bandwidth resource waste; in contrast, too low allocation will result with frequent requests made by the VP agents to the network and with heavy processing load.

In this work we deal with this problem, aiming at deriving simple operational rules to determine the amount of bandwidth resources to be delegated to the various VP agents. We formulate the resource allocation problem by accounting both for the bandwidth resource waste and for call processing costs. In this context we recognize that the network owner's objective is to maximize the use of the bandwidth resources, or in other words, to minimize the amount of connection request blocking. On the other hand, the call processing resources do not form an objective of their own; rather – they suppose to facilitate the management of the bandwidth resources (connection setup and tear-down). As such, their rate is simply limited by the processing rate of the network (or of the proper routers or switches) and they should be treated as a constraint. Thus the operational problem to be addressed is that of minimizing bandwidth waste under a connection processing constraint. (This is in contrast to a network design problem in which the designer may be free to choose the amount of processing resources and thus aims at *optimizing* them.) It should be noted that if one is not concerned with the connection processing constraint, then an optimal allocation of bandwidth resources can be easily achieved via a centralized agent to which each and every call set up (and release) request must be forwarded for approval; however this would result with tremendous and impractical call processing overhead. Thus accounting for the processing constraint is essential.

The problem as a whole seems to be very complex. We therefore resort to a decomposition approach, in which we recognize that a key to the optimization of the whole network is proper optimization at the single link level. Such optimization must account properly for the parameters and lead to a relatively simple solution, so that it can be plugged into over-all network algorithms. We thus concentrate on analyzing each link in isolation aiming at deriving simple operational rules that can be utilized in network algorithms. For this problem we deal with all the VPs passing through the link (each with different traffic volume) and are interested in finding efficient operational rules regarding the allocation of bandwidth resources to each of the VPs passing through it. That is, we deal with how many bandwidth resources should be granted to the various VPs (namely to their agents) and at what situation. Investigating this *single link* problem we realize that the pure problem is too complex to yield exact analytic solution and thus formulate an approximate model for it.

We analyze this model and derive the *optimal allocation* for it. The allocation is expressed in terms of the expected amount of processing the system should delegate to the various VP agents. The importance of the allocation rule derived is that it is expressed in a closed form expression. This can be used as a heuristics for more complex formulations, in particular for the full network resource allocation problem. The rule derived is a "square-root" rule, and it suggests that the *amount of processing* occupied by a *VP agent* should be *proportional to the square root of its request load.*

Once the expected amount of processing is allocated to the various VPs, we next specify the exact bandwidth allocation policy for each VP, where fixed threshold policy can be used. Having applied our processing allocation rule to fixed thresholds policy we then demonstrate how our allocation rule can be combined with an adaptive hysteresis-type threshold rule as proposed in [19]. Numerical examination of a variety of cases shows that the approximate rules derived are very efficient for the exact problem.

The operational rules derived for the single link model can be used as a building block in devising a general network algorithm. In the context of this paper we demonstrate how this rule can be applied by proposing a simple network-wide algorithm which can be performed either in a distributed manner on in a centralized manner.

The structure of this paper is as follows: In Section 2 we describe the single link model and formulate the cost functions. In Section 3 we analyze this model and derive an efficient closed form allocation rule based on fixed thresholds. In Section 4 we demonstrate how our resource allocation rule can be combined with a hysteresis-type adaptive threshold policy. In Section 5 we examine the quality of the allocation rule by checking a variety of numerical cases. In Section 6 we propose network-wide algorithms based on the operational rules devised for the single link problem.

1.1 Related work

In addition to the references mentioned above, the literature contains many references dealing with VP design issues. Most of those studies deal with the *static* allocation of resources to VPs (VP capacity) and their relation to routing and Quality of Service. Evans [8] treats the problem of optimizing the network capacity granted to various services, where various VPs carry different services and have different value to the network. Medhi[18] considers the problem of efficient network design and management and examines how the network performance can improve by reassigning bandwidth to the VPs several times a day. Herzberg [10] and Herzberg and Pitsillides [11] use linear programming to deal with the problem of bandwidth assignment to VPs to maximize the total network throughput. Pitsillides et al. [20] use genetic algorithms to solve the optimization problem of allocating bandwidth to VPs as to maximize the multi-objective function of throughputs in the network. The use of flexible pipes (such as the VPs) for optimizing network operation and bandwidth consumption is very common, see e.g., Gerla et al. [9]. The advantages of virtual paths for bandwidth management are discussed in Burgin [6]. The use of VPs for traffic engineering purposes was discussed in several articles, see e.g., Awduche [3], Awduche et al. [4], Elwalid et al. [7], and Swallow [22] who deal with this problem in the MPLS context. A good reference for various VP design issues is Anerousis and Lazar [2] and the references appearing there. Partial results of our work were reported in [16].

2 The single link problem: A model of resource request and allocation

We start our analysis with a simplifying model dealing with the allocation of resources in a single link (or node) in isolation. The solution to this problem should be used as a building block in solving full network problems. A system consists of V *virtual path agents* which compete for R bandwidth resources. This represents a single link which can contain up to R connections and which are to be shared by the V VPs. For each VP, a VP agent, say v, is in charge of all the connections belonging to the VP which are the connections of certain type (in the global network formulation of the problem this will translate, for example, to all the traffic that originates in the network at node x and destined to node y). We assume that the resource requests made by a connection are identical to all connections; A single resource, therefore, can model the amount of bandwidth required by a single connection. For ease of notation we call these bandwidth resources. The link agent (*central agent*) is in charge of allocating the link resources to the connections. The link agent does it by delegating bandwidth resources to the VP agents. As such, the link agent grants, from time to time, a pool of resources to v, based on v's needs. Agent v, in turn, grants the resources to the connections upon demand. v also releases pools of resources back to the link agent once they are not utilized (due to connection release).

We assume that VP agent v receives a Poisson [1] stream of requests at rate λ_v. Each request demands a single resource (single connection). The duration of a connection is assumed to be exponentially distributed with mean $1/\mu$, and thus when i resources are occupied by v their total release rate is $i\mu$.

Let $D_v(t)$ denote the number of resources (connections) demanded from agent v at time t. $D_v(t)$ can be represented by a Markov chain which is depicted in Figure 1.

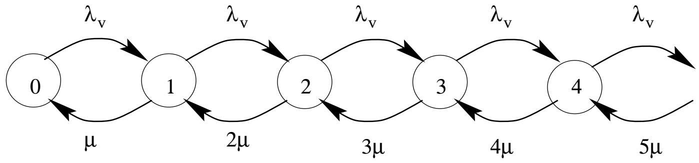

Figure 1: The Markov chain of resources required by a single VP

The allocation and release of resources to v is done in "bundles". Formally, let $G_v(t)$ denote the number of resources granted to v by the central agent. When v receives a connection set-up request at time t it examines whether the number of resources allocated (granted) to it, $G_v(t)$, allows the accommodation of the new request; in other words, whether $D_v(t)$ exceeds $G_v(t)$. If $D_v(t) \leq G_v(t)$ v grants the request without making more demands to the link agent. If, on the other hand, $D_v(t) > G_v(t^-)$, v requests a "bundle" of resources B_v whose size is a function of G_v, namely $B_v(G_v(t^-))$, from the link agent and updates G_v accordingly: $G_v(t) \leftarrow G_v(t^-) + B_v(G_v(t^-))$.

Upon the release of a connection, v updates $D_v(t)$ (reduce by 1). Further, if it finds out that it possesses more than one bundle of idle resources, it releases the bundle and returns it to the link agent.

A more formal description of the actions taken by v is as follows: v employs a resource request policy π, denoted **fixed threshold policy**, which is characterized by a set of thresholds $T^\pi = \{T_0, T_1, T_2, ...\}$, $0 \leq T_0 < T_1 < T_2 < ... <$. The number of resources granted at epoch t to the VP entity v, $G_v(t)$ obeys

$$G_v(t) = G_v(D_v(t)) = min\{T_i : T_i \geq D_v(t)\}. \qquad (1)$$

Updates of G_v are made either when this equation is violated due to a new request (increment of $D_v(t)$) or when it is violated due a resource release (decrement of $D_v(t)$).

The selection of these thresholds influences the operation of the system in two ways: 1) The bandwidth resource utilization increases with the number of thresholds, and 2) the costs of processing a request increase with the number of thresholds. The trade-off between these factors as to optimize the system performance is the main subject of this

[1] The modeling of call arrivals as a Poisson stream and call duration as an exponential process are very common in telephony. Furthermore, in the new IP networks, it is reasonable to expect that Voice over IP (VoIP) sessions (which resemble telephony in their characteristics) will require their own virtual pipes, which will be granted top priority at the network (due to their stringent quality of service requirements). Thus, we expect that the traditional "exponential modeling" of telephony should be well suitable for modeling VoIP in the new generation IP networks. This modeling might be in contrast to the modeling of *data* for which studies have shown higher burstiness and long-tail properties. Nonetheless, one should note that the modeling (as Poisson) serves only as the basis for the optimization problem used to derive an efficient (approximate) operation rule. The operation rule itself might potentially prove useful for other types of traffic.

work and the modeling of these factors is done in Section 2.2 below. Special case policies (of selecting the thresholds) are enlisted next.

2.1 Special case policies

Several special cases of the above policy are:

No dedication (fully dynamic allocation): In this policy $T_i = i$, i=0, 1, 2, ... In other words, the VP agent actually does not exist, since all requests go to the central agent. This policy maximizes bandwidth utilization as well as the link processing cost.

One-time dedication (fully static allocation): In this policy all resources are statically allocated and dedicated to the VP agents and the central agent keeps no resources. Agent v sets T_0 to be equal to the number of resources allocated to v, and whenever more than T_0 resources are required from v the request results in blocking.

One-time dedication with a central pool: In this policy each VP agent receives one allocation of dedicated resources, namely $T_0 \geq 1$. When the number of resources required from v exceeds T_0, v makes a request for *a single resource* whenever it is needed, and releases it once it is not utilized. This policy can be described formally by $T_0 \geq 1$ and $T_{i+1} = T_i + 1$, $i = 1,, 2....$

2.2 Performance and cost

As stated above and presented in Figure 1 the behavior of the demands from agent v can be described by a simple Markov chain. In particular, let d_k^v, $k = 0, 1, 2, ...$, be the probability that at steady state there are k connections demanded from v, namely, $d_k^v = lim_{t\to\infty} Pr[D_v(t) = k]$. Then d_k^v obeys (see, e.g., Kleinrock [13]):

$$d_k^v = \frac{(\lambda_v/\mu)^k/(k!)}{\sum_{i=0}^{\infty}(\lambda_v/\mu)^i/(i!)}; k = 0, 1, 2, ... \qquad (2)$$

Assuming that there is *no blocking* in the system, let g_k be the probability that at steady state there are k resources granted to v (namely $g_k = lim_{t\to\infty} Pr[G_v(t) = k]$, $k = 0, 1, 2, ...$). Then we have:

$$g_{T_i} = \begin{cases} \sum_{j=T_{i-1}+1}^{T_i} d_j & i = 1, 2, ... \\ 0 & otherwise \end{cases} \qquad (3)$$

Two major costs are incurred in the operation of this system:

1. *Bandwidth resource cost* - modeled by measuring the fraction of connections that get denied due to lack of bandwidth resources; this is the connection blocking rate.
2. *Processing cost* - measuring the amount of efforts spent on processing a request. To this end the reader should note that these costs break into two costs: a) *Agent processing* and, b) *network processing*. Agent processing occurs for each connection both at connection set-up and tear-down. This cost is unavoidable and thus is not subject to optimization. However, it is relatively cheap since it requires only a simple operation by the agent. Network processing, in contrast, is expensive since it may require involvement of many nodes in the network (at least the nodes on the path of the VP). However, this processing occurs only at reservation epochs, namely at threshold points. Therefore, one would like to control this high cost by controlling the set of the threshold points. Thus for the sake of our analysis we will focus on the network processing cost, which we will denote, in short, processing cost.

3 The multiple-VP single-link problem: Analyzing The allocation of resources to non-uniform VP entities

Having described the behavior of a VP on a link and discussing the costs associated with it, we are now ready to analyze the operation of multiple VP's while focusing on a single link. We assume that the VP's passing through the link under consideration differ from each other in their arrival rates and recall that λ_v is the arrival rate to VP v. The total arrival rate to the link is given by $\Lambda = \sum_{v=1}^{V} \lambda_v$. The connection service rate (call completion rate) is identical to all VP's and is given by μ.

Our objective is to derive an efficient, yet simple, policy for the operation of these agents. More specifically, we aim at designing a threshold policy for each of the VP agents.

The system efficiency will be measured by its blocking rate. The processing rate, in contrast, is treated as a constraint, modeling the fact that the amount of processing power that can be used by the system is fixed and preassigned. Thus the problem addressed will be that of minimizing the blocking rate under a processing rate constraint.

Let D_v be a random variable denoting the number of resources demanded (requested) from agent v when the system is under steady state. Let d_k^v denote the probability that at steady state k resources are requested from v. Then the value of d_k^v is given in Eq. 2.

Let $T_0^v < T_1^v, ...$ denote the thresholds applied to v. Let θ_v denote the probability that under steady state v is in a threshold state, namely:

$$\theta_v = \sum_{i=0}^{\infty} d_{T_i^v}^v. \qquad (4)$$

Let R_v denote the rate of requests made by agent v to the link agent. R_v is given by:

$$R_v = 2\lambda_v\theta_v, \qquad (5)$$

where the 2 results from accounting both for resource requests and for resource release (whose rates are off-course identical to each other).

We assume that the link agent has certain processing capacity designed to handle request processing. Let R_{max}

be the total request rate that can be handled by this agent. Thus we must design our policy subject to the constraint:

$$\sum_{v=1}^{V} R_v \leq R_{max}. \tag{6}$$

Let $G_v(D_v)$ be a random variable denoting the number of resources granted (provided) to agent v by the link agent (as dictated by the policy) when the system is in steady state. Note that if the value of D_v is d then the value of $G_v(D_v)$ is equal to the smallest threshold value g such that $d \leq g$. Let G be a random variable denoting the total number of resources that are requested from the link (central) agent. This obeys:

$$G = \sum_{v=1}^{V} G_v. \tag{7}$$

Let W_v be a random variable denoting the number of resources "wasted" at agent v; This is the number of resources allocated to v but not used by v, which obeys:

$$W_v(D_v) = G_v(D_v) - D_v \geq 0. \tag{8}$$

Let G_{max} be the total number of resources available to the (central) link agent. Subject to the constraint given in Eq. 6 our objective is to minimize the probability of blocking, given by:

$$\begin{aligned} Pr[Blocking] &= Pr[G \geq G_{max}] \\ &= Pr[\sum_{v=1}^{V} D_v + \sum_{v=1}^{V} W_v(D_v) \geq G_{max}]. \end{aligned}$$

Thus our objective is the minimization:

$$\begin{aligned} min \quad & Pr[\sum_{v=1}^{V} W_v(D_v) \geq G_{max} - \sum_{v=1}^{V} D_v] \\ s.t. \quad & \sum_{v=1}^{V} R_v \leq R_{max}. \end{aligned} \tag{9}$$

3.1 Allocation algorithm: Optimal allocation of processing rates to VP agents – approximate optimization

The minimization given in Eq. 9 is hard to conduct analytically. We therefore resort to an approximation. To this end we note that the value of the objective function is affected by the policy chosen via the variables W_v (wasted resources), and that the major statistics of W_v affecting Eq. 9 are the expected value and variance of $\sum_{v=1}^{V} W_v$. Thus we will transform this minimization to the minimization of the expected value of $W = \sum_{v=1}^{V} W_v$.

Further, to allow expressing W_v conveniently as a function of the policy chosen we next resort to randomization of the threshold policy. To this end we will consider a family of policies in which each of the states in the Markov chain of v is *probabilistically* (as opposed to deterministically) chosen to be a threshold state. Specifically, let f_v denote the probability that a given state in the Markov chain of agent v is a threshold state. Under this assumption the probability that the agent is in threshold state, θ_v, is simply given by:

$$\theta_v = f_v. \tag{10}$$

Under this assumption it is easy to see that $W_v(D_v)$ is independent of D_v and that it is distributed geometrically with parameter f_v whose mean and variance are:

$$E[W_v] = \frac{1-f_v}{f_v}; \quad Var[W_v] = \frac{1-f_v}{f_v^2}. \tag{11}$$

Further, the processing rate at agent v is simply given by:

$$R_v = 2\lambda_v \sum_{k=0}^{\infty} d_k^v f_v = 2\lambda_v f_v. \tag{12}$$

Thus, the optimization problem now translates to:

$$\begin{aligned} Min_{\{f_v\}} \quad & E[W] = \sum_{v=1}^{V} \frac{1-f_v}{f_v} \\ s.t. \quad & \sum_{v=1}^{V} 2\lambda_v f_v \leq R_{max} \\ & 0 \leq f_v \leq 1; \quad v = 1, ..., V. \end{aligned} \tag{13}$$

3.1.1 The optimal solution

It is easy to see that the structure of the objective function in Eq. 13 dictates that the constraint $f_v \geq 0$ is redundant. In contrast, the constraint $f_v \leq 1$ is meaningful; thus special consideration must be given to the V constraints involved.

To address this problem let us assume (without loss of generality) that $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_V$.

Let f_v^* be the optimal value of f_v $(1 \leq v \leq V)$. Then the next lemma establishes a major property of the optimal solution:

Lemma 1 *For $2 \leq i \leq V$, if $f_i^* = 1$ then $f_{i-1}^* = 1$.*

The proof is given in [16].

In a similar manner one can prove the following property:

Theorem 1 *The optimal values obey : $f_1^* \geq f_2^* \geq ... \geq f_V^*$.*

Lemma 1 leads to the following property regarding the structure of the optimal solution:

Corollary 1 *The optimal solution $f_1^*, ..., f_V^*$ follows the following structure: There exists $0 \leq \tilde{v} \leq V$ such that: 1) $f_v^* = 1$ for $v \leq \tilde{v}$, and 2) $f_v < 1$ for $v > \tilde{v}$.*

The derivation of f_v^* for $v > \tilde{v}$ can be done using the minimization problem given in Eq. 13 where 1 is substituted for $f_1, ..., f_{\tilde{v}}$ and the constraints $f_v \leq 1$ are disregarded. This yields the final solution as follows :

$$f_v^* = \begin{cases} 1 & v \leq \tilde{v} \\ \frac{1}{\sqrt{\lambda_v}} \frac{R_{max} - 2\sum_{i=1}^{\tilde{v}} \lambda_i}{2\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}} & v > \tilde{v} \end{cases} \tag{14}$$

The optimal processing rate R_v^* is given by:

$$\begin{aligned} R_v^* &= 2\lambda_v f_v^* \qquad (15) \\ &= \begin{cases} 2\lambda_v & v \leq \tilde{v} \\ \frac{\sqrt{\lambda_v}}{\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}} \left(R_{max} - 2\sum_{i=1}^{\tilde{v}} \lambda_i \right) & v > \tilde{v} \end{cases} . \end{aligned}$$

Finally, the value of $\tilde{v}$ can be found using a linear search on v and utilizing the following observation:

Observation 2 *$f_v^* = 1$ if and only if a) $f_{v-1}^* = 1$ and, b) $\frac{1}{\sqrt{\lambda_v}} \frac{R_{max} - 2\sum_{u=1}^{v-1} \lambda_u}{2\sum_{u=v}^{V} \sqrt{\lambda_u}} \geq 1$.*

The expected wasted resources at v and at the whole system, under this optimal solution are given by:

$$E[W_{v^*}] = \begin{cases} 0 & v \leq \tilde{v} \\ \frac{2\sqrt{\lambda_v}\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}}{R_{max} - 2\sum_{u=1}^{\tilde{v}} \lambda_u} - 1 & v > \tilde{v} \end{cases} \tag{16}$$

$$\sum_{v=1}^{V} E[W_{v^*}] = \frac{2[\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}]^2}{R_{max} - 2\sum_{u=1}^{\tilde{v}} \lambda_u} - (V - \tilde{v}). \tag{17}$$

3.1.2 A functional form of the optimal solution

In the sequel, the structure of the optimal solution (Eq. 14) will be used in the derivation of network resource allocation strategies. To that end we next derive one more property of this solution. Let us denote by $F^*(x)$ the *link allocation function* and define:

$$F^*(x) = min \left\{ 1, \frac{1}{\sqrt{x}} \frac{R_{max} - 2\sum_{i=1}^{\tilde{v}} \lambda_i}{2\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}} \right\} \quad x > 0. \tag{18}$$

The following theorem states that f_v^* is identical to the value of the link allocation function $F^*(x)$ at $x = \lambda_v$.

Theorem 3 *For $v = 1, ..., V$, the optimal values f_v^* obey $f_v^* = F^*(\lambda_v)$.*

For lack of space the proof is omitted and can be found in [15].

3.1.3 Discussion

The optimal allocation given in Eq. 15 implies that the allocation of processing rate R_v is monotonic increasing with the arrival rate λ and follows a sqaure-root proportional allocation. This sqaure-root rule resembles prior network optimization solutions, such as the *capacity assignment* of link capacities (see [14]).

3.1.4 Model generalization to multi-resource VCs

The model described above assumes that each VC demands exactly one resource. The model can be extended to the case where the VCs of VP v demand c_v resources where c_v is an integer. This can be done by keeping the form of the Markov chain (in Figure 1) and modifying the states to represent multiples of c_v. This will modify the optimization equation 13 by modifying the sum $\sum_{v=1}^{V} \frac{1-f_v}{f_v}$ to sum $\sum_{v=1}^{V} c_v \frac{1-f_v}{f_v}$ (and not affecting the constraint). The resulting optimal solution for f_v^* should be now proportional to $\sqrt{c_v}/\sqrt{\lambda_v}$ instead of being proportional to $1/\sqrt{\lambda_v}$. The optimal processing rate R_v^* is proportional to $\sqrt{c_v \lambda_v}$. The practical implication is that the processing rate, in an optimal system, monotonically increases both in the arrival rate and in the number of resources required by the VC and is proportional to the square root of their product. This result can be used as a design guideline for systems with different characteristics VP's.

3.2 Fixed threshold algorithm: Selection of thresholds under processing cost constraint

Given the selection of the processing rate to agent v, $\theta_v^* = f_v^*$, it is required to propose an efficient threshold policy for this agent. Following the formulation of the previous subsection we may adopt the objective function of minimizing the expected "wasted" resources subject to the constraint that the processing rate does not exceed R_v.

In a deterministic threshold policy a state of the Markov chain is deterministically either a threshold or not. In that case the processing cost constraint is given by:

$$2\lambda_v \sum_{i=1}^{\infty} d_{T_i^v} \leq R_v. \tag{19}$$

Alternatively one may generalize the deterministic threshold policy to a stochastic threshold policy in which state i is selected to be a threshold with probability f_i^v; note that the deterministic policy is a special case of the stochastic policy in which f_i^v receives the values of 0 or 1 only. Further, the probabilistic policy introduced in the previous section is also a special case in which $f_i^v = f_v$. In the following analysis (Section 3.2.1) we show that an optimal policy must be (almost) deterministic. We then (Section 3.2.2) propose a simple heuristic deterministic threshold algorithm.

3.2.1 An optimal threshold policy must be (almost) deterministic

Considering the objective function, let W_i^v be the number of resources wasted when agent v is at state i ($i = 0, 1, 2...$). Thus we have

$$E[W_v] = \sum_{k=0}^{\infty} d_k^v E[W_k^v]. \tag{20}$$

Further, one may recursively compute $E[W_k^v]$ as follows:

$$E[W_k^v] = f_k^v \cdot 0 + (1 - f_k^v)(1 + E[W_{k+1}^v]); \;\; k = 0, 1, 2, ... \tag{21}$$

Theoretically, this recursion can bear computation difficulties due to being a downward infinite recursion; however, practically, there exists some k_{max} such that $f_k^v = 1$ for every $k \geq k_{max}$ and thus $E[W_k] = 0$ and the recursion can go downward starting at k_{max}.

The optimization problem therefore boils down to the following:

$$\begin{aligned} Min_{\{f_k^v\}} \quad & E[W_v] \\ s.t \quad & \sum_{k=0}^{\infty} 2\lambda_v d_k^v f_k^v \leq R_v^* \\ & 0 \leq f_k^v \leq 1. \end{aligned} \tag{22}$$

Observation 4 *It is easy to see from Eqs. 20 and 21 that the objective function $E[W_v]$ is monotonically decreasing with f_k^v. Thus the optimal solution must be on the constraint.*

Let ${f^*}_k^v$ denote the optimal solution of Eq. 22. Below we claim that this solution must consist mainly of integers, namely that the values of all variables ${f^*}_k^v$, $k = 0, 1, 2, ...$, except perhaps for a single variable, must obey ${f^*}_k^v = 0$ or ${f^*}_k^v = 1$.

For the sake of brevity, we omit the index v in this analysis.

Lemma 2 *Consider a solution $\{f_k^v\}, k = 0, 1, ...$ of the minimization problem given in Eq. 22. If there exist $0 \leq i < j$ such that $0 < f_i^v < 1$ and $0 < f_j^v < 1$ then the set $\{f_k^v\}, k = 0, 1, ...$ is not an optimal solution of Eq. 22.*

The proof of this lemma can be found in [16].

This leads to the following corollary:

Corollary 2 *An optimal threshold policy must be "almost deterministic", in the sense that except perhaps for one state, each state must be, deterministically, either a threshold or not.*

3.2.2 A deterministic "equal spacing" heuristic threshold algorithm

The Equal-Spacing policy determines the thresholds of v at a link using the following heuristics: It attempts to equalize the expected waste of resources between every pair of thresholds. This is done under the obvious constraint that the total processing rate of v is bounded as in Eq. 19.

Recall that d_k^v denotes the probability that k resources are demanded by v and that T_i is the ith threshold, then the expected waste W_i between the thresholds T_{i-1} and T_i is :

$$W_i = \sum_{k=T_{i-1}+1}^{T_i - 1} d_k^v \cdot (T_i - k); \;\; i = 1, 2, ... \tag{23}$$

Thus, formally, the objective of the heuristics is to choose the threshold such that the values of W_i, $i = 1, 2, ...$ will be as close as possible to each other.

For a given Markov chain there are possibly several different optional sets that fit the heuristics and obey the constraint. Among all these we select the set that minimizes the expected waste.

Following this allocation, it may turn out that not all of the processing rate is used by the allocation. In that case we utilize the extra available processing rate by allocating it over the Markov chain from right to left (namely, by setting up the high index states of the Markov chain to be thresholds, until consuming all the processing rate available).

3.3 Algorithm: Single link resource allocation

The full allocation algorithm for a single link, denoted **single link resource allocation algorithm**, is a combination of the two algorithms developed above. First, optimal processing rates, R_v^*, are allocated to the VP's using the allocation algorithm (Eq. 14 and 15, and Observation 2). Then the thresholds for each of the VP's are determined using the heuristic algorithm proposed in Section 3.2.2.

4 The use of an adaptive hysteresis-type threshold policy

The algorithm provided in Section 3.2 focuses on the use of *fixed thresholds* for each VP. Under fixed thresholds an allocation point serves also as a release point. Thus if a VP makes a request for the allocation of more connections when it is at state j (namely when j connections are occupied by the VP), it will release the newly requested connections once reaching state j again. This may lead to vibration situations under which a VP may release a set of connections soon after it acquired it and vice versa.

To avoid these situations a hysteresis type policy can be employed. A hysteresis type policy will operate as follows: Suppose that state k is a point of connection request (or release). At this state the VP chooses two values $l \leq k$ and $k \leq u$. These two values are set as the new release and request points respectively. The number of resources to be allocated or released at k is determined by these values.

A hysteresis type policy as described above was proposed at [19]. In that paper the authors consider a VP in isolation and analyze the hysteresis type policy as to optimize the selection of the points l and u as function of k.

Below we propose to combine the resource allocation strategies devised in the previous sections with the hysteresis type policy. Specifically, we propose to use the single link allocation expressions (Eq. 14 and 15) and the network allocation algorithms to determine the amount of processing rate to be allocated to each VP. Once this allocation is set, we then use the hysteresis type algorithm to determine the actual allocation and release within each VP. As such, the hysteresis type algorithm can be used to replace the heuristic threshold policy described in Section 3.2.

4.1 Analysis

The policy presented in [19] and described above sets lower (release) and upper (request) points for every state k, namely

$l(k)$ and $u(k)$. This in turn implies representation of the system state by a two dimension Markov chain where a state consists of the couple (k, x) where k is the current allocation/release state and x is the number of resources demanded from the VP. The analysis provided in [19] shows that this two-dimensional Markov chain can be partitioned into auxiliary one-dimensional Markov chains, each corresponds to different value of k, $k = 0, 1, 2, \ldots$. The auxiliary Markov Chain is given in Figure 2, and consists of the segment of the Markov chain provided in Figure 1 containing the states $l, l+1, \ldots, k, k+1, \ldots, u$ with additional two transitions from l and u to k.

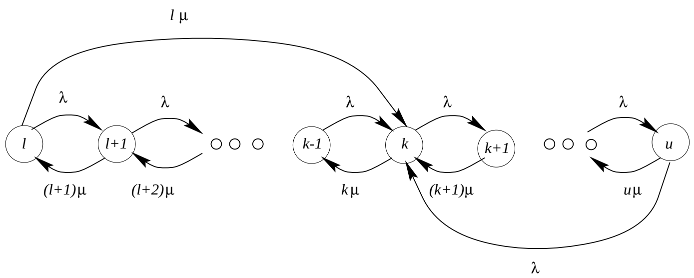

Figure 2: The Markov Chain for the Hysteresis Policy

The analysis in [19] shows that the steady state probabilities of the auxiliary Markov chain is given by:

$$P_{l+j} = \frac{\sum_{i=0}^{j}(\prod_{n=0}^{j-1-i}(l+n))\lambda^i\mu^{j-i}}{(\prod_{n=1}^{j}(l+n))\mu^j}P_l, \quad j = 1, 2, \ldots, k-l \tag{24}$$

$$P_{u-j} = \frac{\sum_{i=0}^{j}(\prod_{n=1}^{j-i}(u-j+n))\lambda^i\mu^{j-i}}{\lambda^j}P_u, \quad j = 0, 1, 2, \ldots, u-k, \tag{25}$$

and

$$\sum_{i=0}^{u-l} P_{l+i} = 1. \tag{26}$$

The processing rate ('hitting rate' in [19]) attributed to this auxiliary chain and characterized by l, k and u is: is given by:

$$R(k, l, u) = \lambda P_u + l\mu P_l. \tag{27}$$

Given the allocation of processing rate given to VP v, namely f_v^* and R_v^*, as given in Eq. 14 and 15, we are now interested in deriving a hysteresis type strategy for v whose total processing rate is given by R_v^*. The following theorem states that if the limits of k are $l(k)$ and $u(k)$ and if we set $R(k, l(k), u(k)) = R_v^*$ for every k, then the total processing rate of the hysteresis type allocation is given by R_v^*.

Theorem 5 *For state k, $k = 0, 1, 2..$ let $l(k)$ and $u(k)$ be the limits set by the allocation algorithm. If $R(k, l(k), u(k)) = r$ for every value of $k = 0, 1, 2, ..$ then the overall processing rate ('hitting rate') of the two-dimensional Markov Chain is given by r.*

Proof: $R(k, l(k), u(k))$ is the conditional processing rate ('hitting rate'), conditioned on the allocation state being k. Thus, if $R(k, l(k), u(k)) = r_k$ then the overall processing rate of the two-dimensional Markov chain is given by

$$\sum_{k=0}^{\infty} Q_k r_k, \tag{28}$$

where Q_k is the probability that at steady state the allocation state is k (the probability that the two dimensional Markov chain is at the kth row). Note that in general this sum is hard to compute since the evaluation of Q_k requires the solution of the two-dimensional Markov chain. However, since Q_k is the probability of the two-dimensional Markov chain to be in state k, we have $\sum_{k=0}^{\infty} Q_k = 1$ and thus, if $r_k = r$ for every k, the above sum trivially collapses to give the value r. $\square$

This analysis, therefore, suggests the following heuristic approach for employing a hysteresis type resource allocation in a single link environment:

1. Use Eq. 14 and 15 to determine the value of R_v^* for all VP's, $v = 1, \ldots, V$.
2. For each VP v do: For each $k = 0, 1, \ldots$ select $l(k)$ and $u(k)$ such that $R(k, l(k), u(k)$, as computed from Eq. 24, 25, 26 and 27 obeys $R(k, l(k), u(k)) = R_v^*$ (or, such that its value is the closest to R_v^*).

5 Numerical examination

5.1 The added value of dynamic allocation

The following examination is geared towards demonstrating the value in conducting dynamic scheduling (as opposed to static scheduling). We demonstrate this value by comparing the **single link resource allocation algorithm** (as described in Section 3.3) to a fully static allocation; in the static policy resources are allocated to the various VP's statically (independent of the VP state) and proportionally to their arrival rate (λ_v).

We evaluate the load that can be sustained by these policies, as a function of the amount of processing resources available in the system, where the objective is to reach blocking rate of 10^{-3}; the results of this comparison are depicted in Figure 3. Note that two cases are depicted in the figure, one for a single link system consisting of 6 high capacity VP's and the other for a single link system consisting of 30 low capacity links. In the first case there are 2 VP's whose arrival rate is 150, 2 whose rate is 300 and 2 whose rate is 450. In the second case there are 10 VP's whose rate is 30, 10 whose rate is 60 and 10 whose rate is 90. Note that in both cases the overall arrival rate is the same ($\Lambda = 1800$) and so is the overall number of resources (1890 connections)

The results (Figure 3) show that the benefit in conducting dynamic allocation is very significant. The sustainable load grows by 50 percent for the low capacity VP system and by 12 percent for the high capacity VP system.

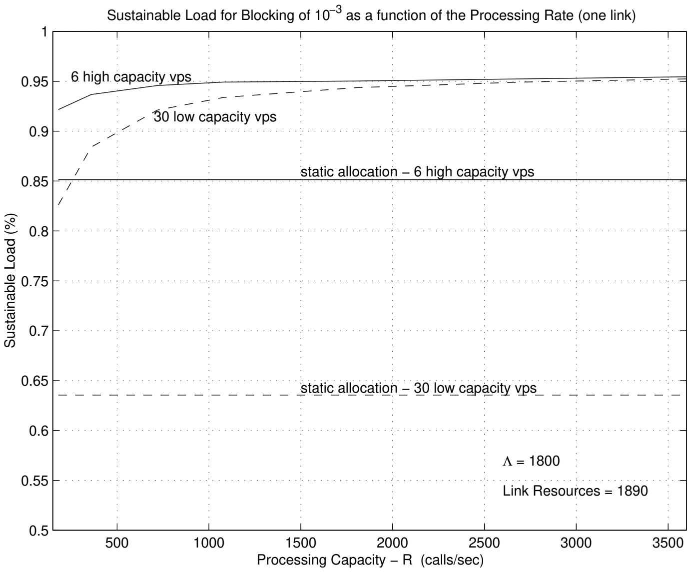

Figure 3: Comparison of Static and Dynamic Allocation Policies

5.2 Analysis of different dynamical allocation policies

In this analysis we evaluate the merits of the allocation algorithm proposed by comparing its performance to that of three alternative algorithms. The 4 strategies evaluated are:

1. **Uniform Single-Threshold (K, 1, 1, ...):** In this algorithm the resources granted to VP v are based on a single threshold, K. That is, the number of resources granted to v is K as long as the number of resources demanded is less than or equal to K; once the number of resources demanded exceeds K, they are granted one by one. This policy is a generalization of the "Erlang-based policy" in which the value of the single threshold (K) is selected to be equal to λ_v/μ. The latter is intuitively based on the fact that the expected number of resources (as well as the most likely number of resources) needed by v is equal to λ_v/μ, while there is low likelihood to demand more than K resources.; this is a common policy in communications networks, especially in telephony.
   Note that K is a free variable in this strategy; the setting of this variable in our numerical examination will be described below.
2. **Equal spacing threshold approach based on uniform single threshold allocation:** In this strategy the allocation of resources (R_v) to the various VP's is done based on the allocation done in the Uniform Single-Threshold policy (namely, it is set to be equal to the allocation given there). Once the value of R_v is set, the actual threshold policy used for the VP is equal-spacing (with the selected value of R_v).
3. **Allocation algorithm combined with equal-spacing algorithm:** This is the **single link resource allocation algorithm** developed in Section 3.3.
4. **"Optimal" allocation combined with equal-spacing algorithm:** In this procedure a wide set of allocations for f_v are numerically examined and to each of them the equal-spacing algorithm is applied. Among these the strategy that yields the minimal number of expected granted resources is chosen. As such, this may be considered to be an "optimal" policy.

In the comparison we consider a single link occupied by 150 VP's, part of which (N_1) are each subject to arrival rate $\lambda_1 = 4$, part are each (N_2) subject to arrival rate $\lambda_2 = 36$ and part (N_3) are each subject to arrival rate $\lambda_3 = 60$. For these values of λ_i and selected values of the processing rates available to the link, R, we evaluate the total number of resources granted by each of the strategies examined.

In the first case examined we select the number of VP's to be $(N_1, N_2, N_3) = (50, 50, 50)$ (Figure 4). The value of R is chosen to be equal to the amount of processing required by the Uniform Single-Threshold $(K, 1, 1...)$ allocation for values of $K = \alpha \cdot (\lambda_v/\mu)$, where $0 < \alpha \leq 1$ is some fraction. In the graph R is normalized by $2\Lambda = \sum_{v=1}^{V} R_v^{max}$, where R_v^{max} is the processing rate of VP v with $f_v = 1$ (Λ is the total arrival rate at the link). The figure depicts the number of granted resources as a function of the amount of processing rate (R) available and the allocation strategy; for the sake of providing a comparison base-line, we also depict (solid line) the expected number of resources demanded (Λ). Note that when $R = 1$ the number of resources granted, by any of the policies, is equal to the number of resources demanded, since allocation is then done one-by-one ($K = 1$). The figure demonstrates that for a wide range of values of the processing rate ($R < 1$), our allocation algorithm (combined with equal spacing) significantly outperforms the two uniform single-threshold algorithms (with and without equal spacing). Further – the performance of our algorithm is very close to that of the "optimal" algorithm and thus cannot be improved much.

Figures 5 and 6 depict the same comparison for drastically asymmetric systems. In Case b (Figure 5) we select $(N_1, N_2, N_3) = (130, 15, 5)$, and $(\lambda_1, \lambda_2, \lambda_3) = (4, 36, 60)$, namely the majority of the VP's are of low volume. In Case c (Figure 6) we select $(N_1, N_2, N_3) = (5, 15, 130)$, and $(\lambda_1, \lambda_2, \lambda_3) = (4, 36, 60)$, namely the majority of the VP's are of high volume. Both these cases demonstrate that significant reduction in the number of granted resources can be achieved using our algorithm. They again demonstrate that the performance of our algorithm is close to "optimal".

To further demonstrate the effectiveness of the "square root rule" (Eq. 14) proposed by our policy we next compare it to an arbitrary rule that allocates the processing resources proportionally to the VP arrival rates, namely proportionally to $1/\lambda_v$ (as opposed to proportionally to $1/\sqrt{\lambda_v}$). In Figure 7 we examine the same system as Figure 4 for two different loads. We see that for very low or very high processing rates they give the same results while for intermediate processing rates the probability of blocking for the $1/\lambda$-proportional policy is twice the probability of blocking for the $1/\sqrt{\lambda}$-proportional policy. In Figure 8

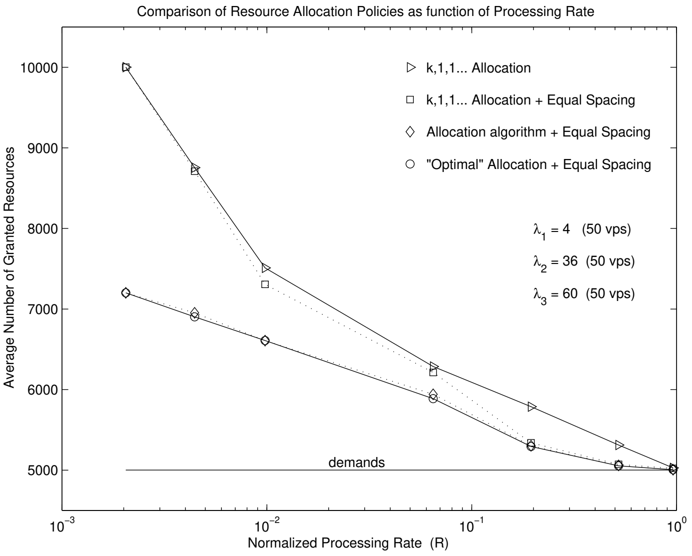

Figure 4: Comparison of Allocation Policies: Case a

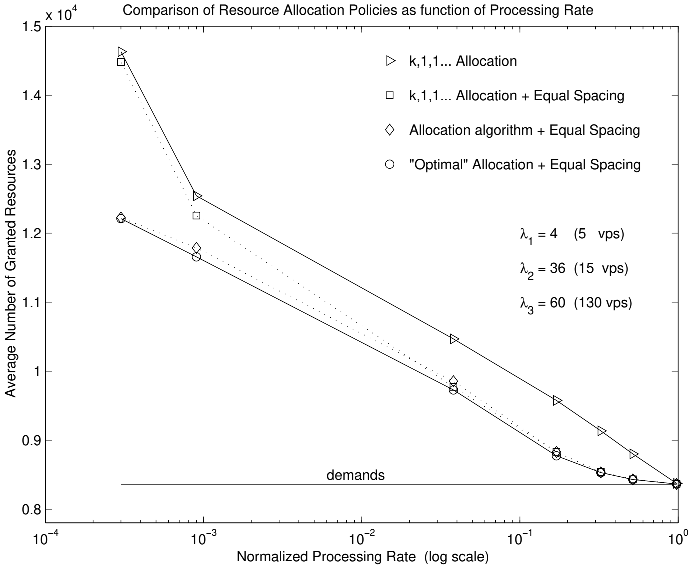

Figure 6: Comparison of Allocation Policies: Case c

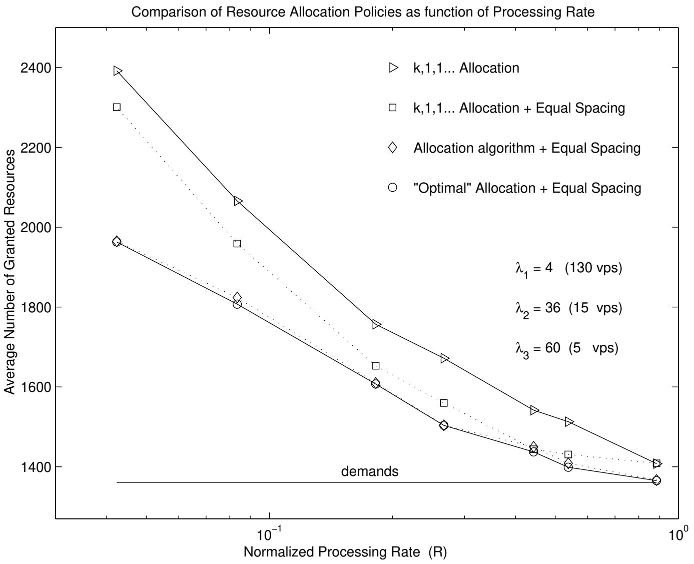

Figure 5: Comparison of Allocation Policies: Case b

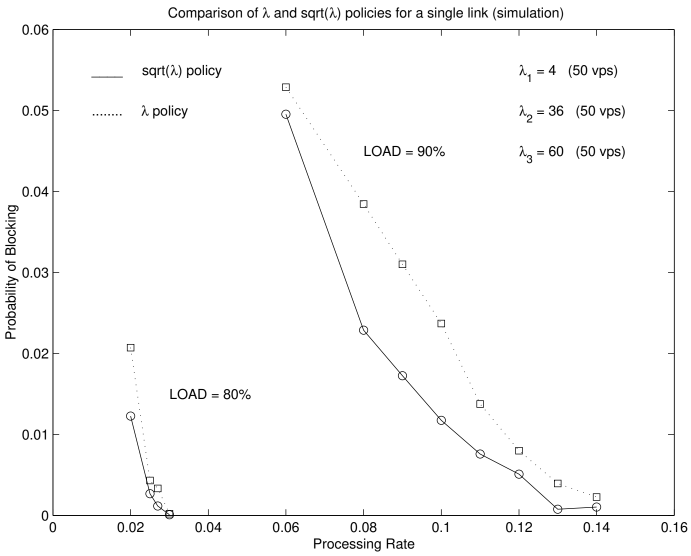

Figure 7: Comparison of Allocation Policies: Case d

we compare two cases, Case A and Case B. Case B is the same system examined in Figure 4 while in Case A the individual VP rates are closer to each other. It is evident in this case that the $1/\sqrt{\lambda}$-proportional policy outperforms the $1/\lambda$-proportional policy. It is also evident that the superiority of the $1/\sqrt{\lambda}$-proportional policy increases with the variability of the rates (note that when the rates are uniform the two policies will yield identical allocations).

6 A general network model

Having formulated the problem and derived its solution for a single link, we now extend this solution to a general network.

Consider an arbitrary network consisting of L links and assume that there are V VP's spanning over the network. Let $X_{v,l}$ be 1 if VP v passes through link l and 0 otherwise. Let $\lambda_{v,l}$ denote the call establishment rate of VP v at link l; since the call establishment rate of a VP is the same along the VP path, we have $\lambda_{v,l} = \lambda_v$ for $X_{v,l} = 1$ and 0 otherwise.

Let $\tilde{V}$ denote the set of VP's and let $\tilde{L}$ denote the set of links. Let $\tilde{L}_v$ denote the set of links through which VP v passes and Let $\tilde{V}_l$ denote the set of VP's passing through link l. Let $D_{v,l}$ and $G_{v,l}$ respectively be the number of resources demanded by and granted to VP v on link l.

The problem faced by the network is that of determining $G_{v,l}$, $1 \le v \le V$, $1 \le l \le L$, based on the demand given by $D_{v,l}$, $1 \le v \le V$, $1 \le l \le L$.

The resource allocation administration consists of a two level hierarchy: In the lower level a VP agent grants resources to connections, and in the higher level links grant resources to the VPs passing through them. Under this

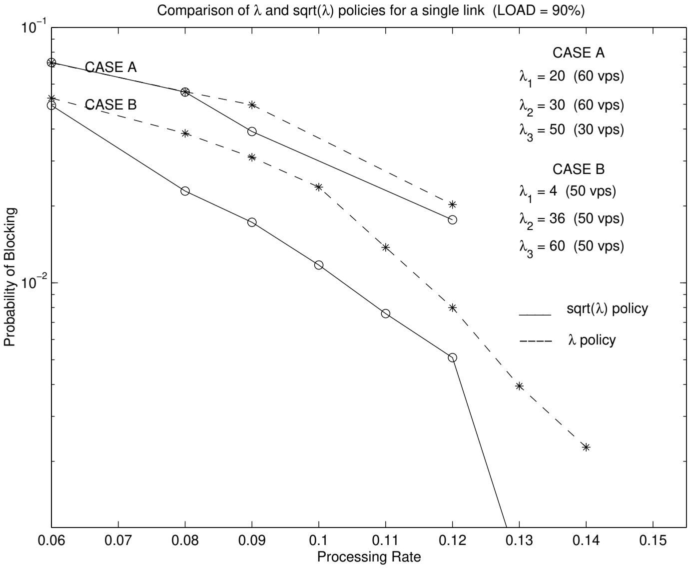

Figure 8: Comparison of Allocation Policies: Case e

setting, the processing rate of VP agent v is designed to sustain the rate of its connection requests (setup and tear-down), and thus must be greater than $2\lambda_v$.

The VP agent, in turn, does not necessarily make requests to the link authorities for every connection setup (or tear-down), and this rate can be controlled by the system designer. Let $R_{v,l}$ denote the rate of requests made by VP v to link l, and let R_l^{max} denote the maximum request rate that link agent l can sustain. The resource allocation strategy must, therefore, obey:

$$\sum_{v \in \tilde{V}_l} R_{v,l} \leq R_l^{max}; \quad 1 \leq l \leq L. \tag{29}$$

A VP resource allocation algorithm can operate under two major strategies: 1) Link-uniform resource allocation, and 2) Link-independent resource allocation. In the first approach the number of resources granted by the various links to a VP agent is the same. In the second approach different links may grant different amount of resources to the VP agent. Below we discuss the first strategy. The second strategy is discussed in [17].

6.1 Link-uniform resource allocation

Under the link-uniform strategy, the number of resources to be granted to v at l, $G_{v,l}$ must obey $G_{v,l} = G_v$. This implies that the values $R_{v,l}$ must obey $R_{v,l} = R_v$.

Thus the formulation of the problem is as follows: Given the set of call arrival rates, λ_v, $v = 1,..,V$, the set of link processing capacity R_l^{max}, $l = 1,...,L$, and the network topology, it is required to select the set of processing rates R_v, $v = 1,...,V$ to be used by the VP agents.

The value of the processing rates R_v can be derived in various ways. In particular, we examine two major alternatives: 1) *Local-optimum based allocation* approach, and 2) *Global optimum allocation*. In the former approach, each link computes the resources (processing rate) it would allocate disregarding the other links in the network, thus following algorithms similar to the ones developed in Section 3. These allocation values are then used by the network to decide on the resource allocation to all VP's on all links. In the latter approach, global optimization, taking into consideration the whole network, is applied. Below we study these approaches.

6.1.1 Local-optimum based allocation

The solution approach taken here is as follows: First the analysis proposed in the single-link optimization is used to derive the optimal allocation of each link, assuming the links are in isolation. Then, the allocations derived by all links are used to derive a consistent allocation for the whole network.

Let $R_v(l)$ denote the processing rate allocated to VP v under the single link algorithm applied at link l (Section 3). Let R_v be the processing rate to be used by VP v in the network. Our interest is therefore to derive R_v, $v = 1,...,V$ as function of the values $R_v(l)$, $v = 1,...,V$, $l = 1,...,L$. To do so, we first characterize the variables $R_v(l)$.

In particular we observe that the network must contain at least one "bottleneck" link. Link l is called *a processing-rate bottleneck link*, if in the single-link optimization algorithm it allocates to each VP v passing through l the minimum allocation v receives (from all links through which v passes). Formally, let $R_v^{min} = min_{l \in \tilde{L}_v} R_v(l)$, and let l_v^{min} denote the index of a link that achieves the minimum, namely $l_v^{min} = argmin_{l \in \tilde{L}_v} R_v(l)$. Then, link l is called *a processing-rate bottleneck link* if $R_v(l) = R_v^{min}$ for all $v \in \tilde{V}_l$.

The following theorem establishes the existence of a bottleneck link.

Theorem 6 *The network must contain at least one link, say l', such that for all $v \in \tilde{V}_{l'}$, $R_v(l') = R_v^{min}$.*

Proof: For the contradiction assume that the claim does not hold. Thus, for each link l there exists (at least one) VP, say v, which passes through l and such that $R_v < R_v(l)$ (and $l_v^{min} \neq l$); let $V_{low}(l)$ denote that VP (if there are several such VP's, pick one arbitrarily) and let $L_{low}(l)$ denote the link at which $V_{low}(l)$ gets its minimum, namely $L_{low}(l) = l_{V_{low}(l)}^{min}$. This means that for every link l, $L_{low}(l) \neq l$. Due to the finite number of links in the network it is easy to see that there must exist a cycle, $l_1, l_2, ..., l_k$ ($k \geq 1$) such that $L_{low}(l_i) = l_{(i+1) mod\, k}$, $1 \leq i \leq k$. Further, it is clear that for $1 \leq i \leq k$, we have

$$R_{V_{low}(l_i)}(l_i) > R_{V_{low}(l_i)}(l_{(i+1) mod\, k}). \tag{30}$$

Since the values of $R_v(l)$ are determined by the single link allocation, it means that $R_v(l)$ is given by: $R_v(l) = \lambda_v f_v^*(l)$ where $f_v^*(l)$ is the value determined in Eq. 14 for link l. Further, from Theorem 3 this is identical to $R_v(l) = \lambda_v F_l^*(\lambda_v)$ where $F_l^*()$ is the function derived in Eq. 18 for link l. Thus, from Eq. 30, we have, for $1 \leq i \leq k$:

$$F_{l_i}^*(\lambda_{V_{low}(l_i)}) > F_{l_{(i+1) mod\, k}}^*(\lambda_{V_{low}(l_i)}). \tag{31}$$

Now, from the shape of the functions $F_l^*()$ (as given in Eq. 18), it is easy to see that for every two such function $F_l^*()$ and $F_{l'}^*()$, if $F_l^*(x_0) > F_{l'}^*(x_0)$ for some $x_0 > 0$, then $F_l^*(x) \geq F_{l'}^*(x)$ for every $x > 0$.

Thus, Eq. 31 implies $F_{l_1}^*(\lambda_{V_{low}(l_1)}) > F_{l_2}^*(\lambda_{V_{low}(l_1)}) \geq F_{l_3}^*(\lambda_{V_{low}(l_1)}) \geq ... \geq F_{l_1}^*(\lambda_{V_{low}(l_1)})$ which is a contradiction.

By the contradiction the theorem is proved.

□

Theorem 6 implies that the set of minimum VP allocations, R_v^{min}, $v = 1, ..., V$ possesses the following two properties: 1) This is a consistent and feasible set, in the sense that it violates no processing rate constraint. 2) There exists at least one link (which we called "bottleneck link") for which this set forms an optimal allocation. We call this an *optimal allocation link*.

These properties, therefore, suggest a simple processing rate allocation algorithm, as follows:

Minimal Processing Rate (MPR) Algorithm:

1. For each link l compute the allocation $R_v(l)$ for all VP's passing through l, as computed by the single link allocation.
2. For each VP v select uniform processing rate R_v as follows: $R_v = R_v^{min} = min_{l \in \tilde{L}_v} R_v(l)$.

Having selected in all links the minimal processing rate allocation, Algorithm A may end up with many processing resources non utilized in various links. The algorithm, nonetheless, guarantees that no processing resources are wasted in the bottleneck link.

A more efficient use of the network resources can therefore be achieved by the following improvement: After conducting the minimal processing rate algorithm, remove from the network the optimally allocated link(s) and all the VP's passing through it (them). For each link remaining in the network update the maximal processing rate by reducing it in the amount allocated to the removed VP's. Now, on the resulted network, conduct the minimal processing rate algorithm again. This approach is formalized in the following algorithm:

MPR Iterative Removal Algorithm:

Input: 1) Links $l = 1, ..., L$, 2) VP's $v = 1, ..., V$, 3) The network topology (represented by the sets $\tilde{L}_v$, $1 \leq v \leq V$ and $\tilde{V}_l$, $1 \leq l \leq L$, 4) The VP arrival rates, λ_v, $1 \leq v \leq V$ and 5) The link processing rates R_l^{max}, $1 \leq l \leq L$.

Output: The allocation of link processing rates to the various VP's, given by the variables R_v, $1 \leq v \leq V$. The allocation attempts to reduce wasted resources.

Method: A greedy approach in which at each stage local optimization at optimally allocated links is enforced.

1. For each link l compute the allocation $R_v(l)$ for all VP's passing through l, as computed by the single link allocation.
2. For each VP v, compute $R_v = min_{l \in \tilde{L}_v} R_v(l)$.
3. Find all optimally allocated links, remove them from the network and fix their processing rate to R_v.
4. For each optimally allocated link l, remove from the network all VP's $v \in \tilde{V}_l$.
5. For each removed VP v, for each link $l \in \tilde{L}_v$ update the link maximum processing rate as follows: $R_l^{max} = R_l^{max} - R_v$.
6. If the number of VP's remaining in the network or the number of links remaining in the network is zero – stop. Otherwise return to 1.

Remark 1 *Steps 1, 2 and 3 can be performed in an alternative, significantly more efficient, approach. Close examination of Eq. 14 reveals that each link is characterized by a specific constant a_l that determines the value of f_v^* for all values λ_v on the link. Accordingly, Eq. 14 can be rewritten as*

$$f_v^* = \begin{cases} 1 & v \leq \tilde{v} \\ \frac{a_l}{\sqrt{\lambda_v}} & v > \tilde{v} \end{cases}, \tag{32}$$

where a_l is given by:

$$a_l = \frac{R_{max} - 2\sum_{i=1}^{\tilde{v}} \lambda_i}{2\sum_{u=\tilde{v}+1}^{V} \sqrt{\lambda_u}}. \tag{33}$$

Now, note that a link l' whose a_l' is the smallest among all links, possesses the following property: For each VP v passing via l', $R_v = R_v(l')$ (see definition of R_v at item 2 above). Hence, this link is an optimally allocated link and can be removed from the network. We use this fact to propose the alternative two steps: 1. For each link l compute the constant a_l.
2. Find the link with minimum a_l, remove it from the network and fix its VP's processing rates to R_v. The computational complexity of this algorithm per iteration, given at this iteration the network has L links and V VP's is $O(L)$, while the computational complexity of the former algorithm is $O(L + \sum_v |\tilde{L}_v|)$.

An important advantage of the optimally-allocated-link iterative removal algorithm is that it can be easily conducted in a *distributed manner*. The only network-global operations conducted boil down to the exchange of messages between link authorities l and VP authorities v such that $v \in \tilde{L}_v$.

6.1.2 Global optimum allocation

In the global optimum allocation approach, the amount of granted resources, $G_{v,l}$, is based on all network parameters. We now extend the approach taken in Section 3 and apply it to the whole network.

Similarly to Section 2 let $W_{v,l}$ be a random variable denoting the number of v's resources wasted on link l (these will be defined only if $X_{v,l} = 1$). Similarly to our previous analysis we will use a stochastic approximation and assume that $W_{v,l}$ is geometrically distributed with parameter $f_{v,l}$. Let W_l be a random variable denoting the number of resources wasted at link l and W the number of resources wasted in the whole network. Thus we have:

$W_l = \sum_{v:X_{v,l}=1} W_{v,l}$ and $W = \sum_{l=1}^{L} W_l$. Let R_l^{max} denote the maximal processing rate available on link l.

Following the single link formulation our incentive is to minimize the wasted resources, in particular their expected values. To this end, note that we are facing a multiple objective problem (the set of variables $\{W_l\}$) and thus one could formulate various objective functions. A natural choice of that function, which we make here, is the expected value of the total waste in the system, namely $E[W] = E[\sum_{l=1}^{L} W_l]$.

Using the stochastic (randomization) approach for evaluating the wasted resources (as in Section 2) we get

$$E[W_{v,l}] = \frac{1 - f_{v,l}}{f_{v,l}}, \tag{34}$$

which, due to the uniformity of the allocation made to v over all the links, gives:

$$E[W_{v,l}] = \frac{1 - f_v}{f_v}. \tag{35}$$

The expected value of the total waste is therefore given by:

$$E[W] = \sum_{v=1}^{V} H(v) \frac{1 - f_v}{f_v}, \tag{36}$$

where $H(v)$ is the length of v (counted in number of links).

The constraints imposed on the system are on the processing rate at each link, namely $\sum_{v \in \tilde{V}_l} 2\lambda_v f_v \leq R_l^{max}$, for $l = 1, ..., L$.

The optimization problem is therefore:

$$\begin{aligned} Min_{\{f_v\}} \quad & E[W] = \sum_{v=1}^{V} H(v) \frac{1 - f_v}{f_v} \\ s.t. \quad & \sum_{v \in \tilde{V}_l} 2\lambda_v f_v \leq R_l^{max}; \quad l = 1, ..., L \\ & 0 \leq f_v \leq 1; \quad v = 1, ..., V. \end{aligned} \tag{37}$$

Note that this is a non-linear optimization problem with linear constraints. It can be solved using commercial LP programs after linearization of the objective function.

6.2 Numerical results

Below we compare the efficiency of the three network algorithms presented by numerically evaluating them on a realistic network. We consider a network consisting of 26 nodes and 41 links which may represent typical wide area networks. A VP is formed between every source and destination yielding 650 VP's [2] The arrival rates of the VP's are assumed to be inverse proportional to the VP length, and the average load on the links is 80% (done by selecting the link capacity properly).

[2]The exact path taken by the VP is selected via an optimization program that aims at optimizing the route selection.

Figure 9 depicts, for each of the allocation policies discussed, the normalized blocking rate [3] of the network versus the processing rate granted to the network. The network blocking rate is normalized by the total arrival rate such that when the ratio is one all arriving connections are blocked [4]. The processing rate is normalized by the maximal processing rate, which is the rate when the VP authority issues a request for each connection setup.

The figure depicts the results of the three algorithms, Minimal Processing Rate (MPR), MPR-iterative, and Global Optimum (the latter is achieved via a Linear Programming program). For the sake of comparison we also provide the results of a Link Independent algorithm, in which for each link resource allocation is optimized via Eq. 14, and where the each VP might use different rates of requests (non-synchronized) to the different links. The extra overhead resulting from this lack of synchronization is not accounted for in the figure, and thus this curve does not represent a practical algorithm, but rather a lower bound.

We observe that for all examined algorithms the blocking rate monotonically decreases with the processing rate and is sensitive in the range of 30%-70% processing rate. We should point out that the probability of blocking in a global network (as depicted in the figure) is significantly larger than the probability of blocking at a single link. This is because blocking at one of the links along the VP blocks the connection establishment on all other links along the VP path as well.

The results show that the iterative MPR algorithm is somewhat better than the (single phase) MPR algorithm, though not drastically. They also show superiority of the Global Optimum allocation algorithm on the two other algorithms. This superiority is somewhat significant but not drastic, implying that one can achieve decent results with the simple algorithms. All three algorithms lie above the Link Independent results which serve as a lower bound. The gap between them can be interpreted as an upper bound on how much the algorithm results can be improved.

References

[1] R.G. Addie, J.L. Burgin, S.L. Sutherland, "B-ISDN protocol architecture," IEEE Globecom'88, Florida,

[3]The blocking rate is computed as follows: First, the network algorithm determines the processing rate f_v (and R_v) for each VP v. The equal spacing algorithm and Eq. 3 are then used to compute the distribution of granted resources of v. Combining these, via a convolution, for all VP's passing at link l yields the distribution of bandwidth resources granted by the link (Eq. 7). The blocking probability is then the probability that this exceeds the maximal bandwidth resources G_{max}. Having derived this, we now have for each VP v the blocking rates (blocking probabilities) on all the links l passes through (denote this $B_{v,l}$). This can be used to approximate v's blocking rate by assuming independence of blocking over the different links and computing $1 - \prod_l (1 - B_{v,l})$. Summarizing blocking rates for all VP's (weighted by their arrival rates) yields the network blocking rate.

[4]The values of the curve at processing rate approaching zero (showing blocking probability of 1) represent a non-practical range of the algorithm, *as implemented here*, since it accounts for processing overhead for the highest threshold, and thus does not represent well static policies. Obviously, at these low values a static policy would yield lower blocking probability than 1.

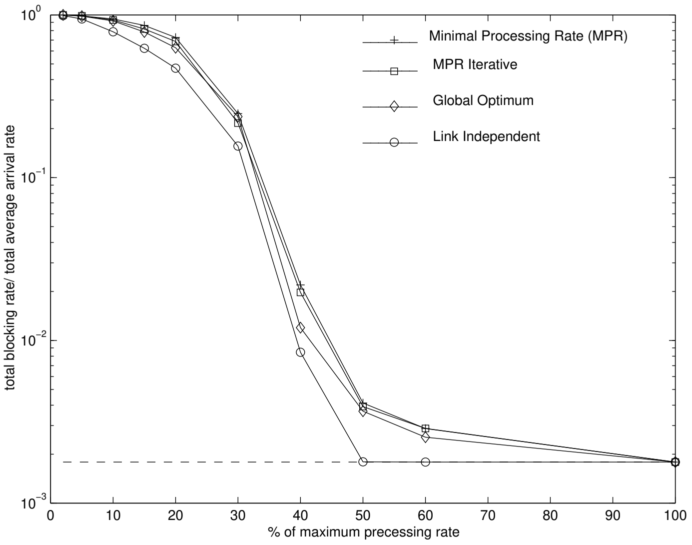

Figure 9: Comparison of Network Algorithms

USA, December 1988.

[2] N. Anerousis and A. A. Lazar, "Virtual Path Control for ATM Networks with Call Level Quality of Service Guarantees," IEEE/ACM Tx. on Networking, Vol. 6, No. 2, pp. 222-236, April 1998.

[3] D. O. Awduche, "MPLS and traffic engineering in IP networks," IEEE Communication Magazine, pp. 42–47, December 1999.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "RFC 2702 Requirements for Traffic Engineering over MPLS," www.ietf.org/rfc/rfc2702.txt, September 1999.

[5] J. Burgin, "Broadband ISDN Resource Management," Computer Networks & ISDN Systems, Vol. 20, No. 1, pp. 323-331, 1990.

[6] J. Burgin and D. Dorman, "Broadband ISDN Resource Management: The Role of Virtual Paths," IEEE Comm. Magazine, Vol. 29, No. 9, pp. 44-48, Sept 1991.

[7] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," IEEE INFOCOM 2001, 1300-1309.

[8] S. Z. Evans, "Optimal Bandwidth management and capacity provision in a broadband network using virtual paths," Performance Evaluation, 13 (1), pp. 27-43, 1991.

[9] M. Gerla, J. A. S. Monteiro and R. Pazos, "Topology design and Bandwidth Allocation in ATM Nets," IEEE J. on Selected Areas in Communications, Vol. 7, No. 8, Oct 1989, pp. 1253-1262.

[10] M. Herzberg, "A Linear Programming Model for Virtual Path Allocation and Management in B-ISDN," ABSSS'92, Melbourne, July 1992.

[11] M. Herzberg and A. Pitsillides, "A Hierarchical Approach for the Bandwidth Allocation, Management and Control in B-ISDN," IEEE International Conference on Communications, ICC'93, Geneva, May 1993.

[12] S. Ohta and K.-I. Sato, "Dynamic Bandwidth Control of the Virtual Path in an Asynchronous Transfer Mode Network," IEEE Tx on Communications, Vol. 40 (7), July 1992, 1239-1247.

[13] L. Kleinrock, Queueing Systems, Vol. 1, Theorey, John Wiley, 1975.

[14] L. Kleinrock, Queueing Systems, Vol. 2, Computer Applications, John Wiley, 1976.

[15] H. Levy, T. Mendelson and G. Goren, "Dynamic Allocation of Resources to Virtual Path Agents," technical report, www.cs.tau.ac.il/˜hanoch/Papers/Dynamic-alloc.ps.

[16] H. Levy, T. Mendelson and G. Goren, "Optimal Use of Virtual Paths for Connections Setup Reduction: The Single Link Problem," INFOCOM'2000, March 2000, Tel-Aviv, Israel. http://www.ieee-infocom.org/2000/papers/117.ps.

[17] H. Levy, T. Mendelson and G. Goren, "Optimal Use of Virtual Paths for Connections Setup Reduction," Technical Report, Tadiran Telecomm, September 1998.

[18] D. Medhi, "Models for Network Design, Servicing and Monitoring of ATM Networks based on the Virtual Path Concept, Computer Networks and ISDN Systems, Vol. 29, No. 3, pp. 373 - 386, 1997.

[19] A. Orda, G. Pacifici and D. Pendarakis, "An Adaptive Virtual Path Allocation Policy for Broadband Networks," pp. 1285-1293 in Proceedings of IEEE INFOCOM'96, San Francisco, CA, USA, March 1996.

[20] A. Pitsillides, G.Stylianou, C. S. Pattichis, A. Sekercioglu, A.Vasilakos, "Aggregated Bandwidth Allocation: Investigation of Performance of Classical Constrained and Genetic Algorithm Based Optimisation Techniques," Computer Communications Journal, Vol 25, Issue 16, pp.1443-1453, October 2002.

[21] K.-I. Sato, S. Ohta and I. Tokisawa, "Broadband-band ATM network architecture based on virtual paths," IEEE Tx., on Communications, vol. 38, pp.1212-1222, August 1990.

[22] G. Swallow, "MPLS Advantages for Traffic Engineering," IEEE Communications Magazine, Vol. 37, No. 12. Dec. 1999, pp.54-57.