Static and Dynamic Evaluation of QoS Properties

Gopal Pandurangan * Eli Upfal *

Abstract

Efficient utilization of modern high bandwidth communication networks relies on *statistical multiplexing* of many *logical* channels through one *physical* channel. Communication requests typically include some statistical characterization of the requested connection (such as pick value, mean values, etc). The task of the network management protocol is to accommodate as many communication requests as possible while keeping the failure (e.g. overflow) probability bounded by a prespecified parameter. When the network consists of one link, the task reduces to evaluating the probability that a sum of random variables does not exceed a given bound. Techniques such as the method of effective bandwidth give a practical solution for the one link problem. In this paper we address the more realistic setting of estimating QoS properties of multi-link networks with arbitrary patterns. The related optimization problem for that setting is $\#P$-complete even for the most simple communication characteristics.

Our main result is an efficient Monte-Carlo method for estimating the failure probability of a general network. Our method is particularly useful in a dynamic setting in which communication requests are dynamically added and eliminated from the system. The amortized cost in our solution of updating the estimate after each change is proportional to the fraction of links involved in the change rather then the total number of links in the network.

*Computer Science Department, Brown University, Box 1910, Providence, RI 02912-1910, USA. E-mail: {gopal, eli}@cs.brown.edu. Supported in part by NSF grant CCR-9731477

1 Introduction

Modern communication protocols such as ATM (Asynchronous Transfer Mode) achieve high utilization of channel bandwidth by multiplexing communication streams with different flow characteristics into one communication channel. Requests for communication are submitted to the network management protocol with some statistical characterization of the required communication. The network (flow) management protocol uses this information to *statistically* multiplex as many communication requests as possible while maintaining global network performance.

Next generation communication networks are expected to provide QoS (quality of service) guarantees when satisfying communication requests. In particular QoS protocol is expected to limit to a pre-specified value the probability of communication failure due to events such as link and buffer overflow, packet loss, jitter, etc. In this paper we focus on failures due to link overflow, our technique can be easily modified to handle any other statisticly governed communication characteristics.

In the case of a one link network, achieving QoS guarantees is reduced to bounding the probability that a sum of random variables exceeds a given bound, where each random variable represents the stream of data of one logical link, and the bound is the bandwidth capacity of the channel (or the adjacent buffers). Most previous works have focused on communication flow modeled by an *on-off source* [6]. A (q, s) on-off source sends at a peak rate of s with probability q and zero otherwise. This model captures the extreme *bursty* nature of high speed networks based on ATM and related technologies. On-off sources have been studied extensively both in theory [6, 9, 7] as well as in simulation studies to evaluate performance of routing algorithms in ATM networks [2]. The work in [9] is especially interesting as it argues that the *fractal* nature of Internet and Ethernet traffic is captured very well by on-off sources. Techniques such as effective bandwidth give efficient and practical solution for providing QoS guarantees on one link networks [1].

The problem of bounding the overflow probability in a multi-channel network of arbitrary pattern is significantly harder. In fact the related optimization problem is $\#P$-complete [10] even if all logical links are fed by on-off sources with the same parameters q and s. We are not aware of any known heuristic to this problem, other than summing the failure probability on all network links. (Which is what is done in practice [2].) Such an estimate is far too expensive, since in large networks it can significantly over estimate overflow probability, and thus under utilizing network capacity. In this paper we study Monte-Carlo solutions for achieving better approximation of QoS properties. We address two versions of this problem. In the *static* version we are given a network and a set of connections running on the network, our goal is to estimate the probability of an edge overflow in the network. In the *dynamic* version we are interested in an efficient procedure that can accumulate information about previous estimates and efficiently check whether adding a new connection would violate the network QoS guarantee, without repeating the Monte-Carlo procedure for the whole network. The dynamic result provides an efficient *admission control* procedure for enforing QoS guarantees in high speed networks.

1.1 Problem Statement and Related Work

We consider a communication network with m physical channels (edges) $e_1, \ldots, e_m$. For each channel e_i we have a constant B_i specifying the maximum *bandwidth* of that channel. Let $c_1, \ldots, c_n$ be the logical channels (connections) in the network. The data flow in a *logical* channel, or a *communication request* is characterized by a distribution function specifying the probability that the channel sends a given amount of data at a given time step. The goal is to satisfy as many communication requests as possible while maintaining pre-specified *Quality of Service (QoS)* guarantees. We focus here on one such guarantee, bounding the probability of overflow in the entire network, other QoS properties (such as packet loss, jitter, etc.) can be handled in a similar way.

While we focus here on the most stringent overflow QoS gurantee, namely bounding the probability of an edge overflow in the entire network, our technique can be easily modified to address weaker QoS guarantees such as the *connection-based overflow constraint* [7]. A connection-based overflow constraint bounds the probability of an overflow along a given logical link. We discuss this variant in section 4 in the context of practical ATM network protocols.

Consider first a simple scenario in which all logical channels are fed by identical (q, s) *on-off sources*, i.e. the data flow distribution in each logical channel has only two states: a flow of rate s with probability q, no flow with probability $1 - q$, and the states on different channels are independent events. Since all logical channels have identical, independent, distributions the probability of an overflow in a given edge can be computed by the Binomial distribution. Computing the probability of an overflow in the entire network, is significantly harder and can be shown to be $\#P$-complete by a straightforward reduction from the union of sets problem [4]. If the flow distribution of the logical channels is less restricted, then even the exact computation of overflow on one edge becomes intractable. In particular, if logical channels are fed by on-off sources with different parameters (q_i, s_i), even computing the overflow probability of one edge becomes $\#P$-complete [7].

In practice, the complexity of computing the overflow on one edge is circumvented by using the method of *effective bandwidth* [3, 6, 1]:

Definition 1 *The effective bandwidth of a random variable X is*

$$\beta_p(X) = \frac{\log E[p^{-X}]}{\log p^{-1}}. \tag{1}$$

Given the effective bandwidth of individual logical channels a bound on the overflow probability of a given edge can be computed using the following bound due to Hui [3]:

Theorem 1 *Let $X_1, \ldots, X_n$ be independent random variables, and $X = \sum_i X_i$. Let $a > b$. If $\sum_i \beta_p(X_i) \leq b$, then $\mathbf{Pr}\{X \geq a\} \leq p^{a-b}$.*

1.2 New Results

Given a method for estimating the overflow probability of a given edge (either directly for simple flow distributions, or through the concept of effective bandwidth for more general distributions) we are interested in estimating the overflow probability of the entire network. More specifically we are interested in two versions of the problem:

1. **The Static Problem:** Given a network and a set of logical channels estimate the overflow probability of the network.
2. **The Dynamic Problem:** Given a network, a set of logical channels, and a sequence of add and delete communication requests, determine for each request if granting that communication request violates the global bound on network overflow probability. If the bound is not violated a new logical channel, satisfying this request, is added to the network.

Let m be the number of physical links in the network, n be the number of active logical links (connections). Our results apply to any set of on-off sources (with possible different parameters to different logical links). For the static case we present an efficient (ϵ, δ)-approximation for link overflow in the network.

Definition 2 *An (ϵ, δ) Monte-Carlo approximation for Q is a value $\tilde{Q}$ such that*

$$\mathbf{Pr}[(1-\epsilon)Q \leq \tilde{Q} \leq (1+\epsilon)Q] \geq 1-\delta,$$

where the probability depends only on random steps made by the approximation algorithm.

In section 2 we prove:

Theorem 2 *There is a Monte-Carlo algorithm that computes an (ϵ, δ) approximation of the probability of overflow of the network in $O(nm\epsilon^{-2}\log\delta^{-1})$ time.*

Henceforth, when we talk about an ϵ approximation *with high probability (whp)*, we mean that δ is $1/m^c$ for some constant $c \geq 1$. We refer to it as an ϵ approximation whp algorithm.

The dynamic algorithm is presented in section 3. For the dynamic setting we define an *incremental* version of an ϵ approximation for determining QoS guarantee whp.

Definition 3 *Let Q_0 be the pre-defined QoS failure probability. Let Q' be the exact failure probability when adding a new communication request. A dynamic algorithm is an ϵ approximation whp for QoS guarantee, if whp the algorithm rejects a new request when $Q' \leq (1-\epsilon)Q_0$, or accepts the new request when $Q' \geq (1+\epsilon)Q_0$.*

Theorem 3 *There is a Monte-Carlo ϵ approximation whp algorithm for the dynamic problem with $O(nf\log m)$ amortized complexity, where f is the number of edges involved in a given change in the communication (added or deleted requests).*

2 The Static Algorithm

Our static algorithm is based on the Karp, Luby and Madras (ϵ, δ) approximation algorithm for the cardinality of union of sets [4].

In the union of m sets problem we are given m sets $D_1, \ldots, D_m$, the goal is to estimate $|\cup_{i=1}^m D_i|$. The basic step in Karp et al. algorithm is choosing, with probability $1/\sum_{i=1}^m |D_i|$, a pair (i, s), where $1 \leq i \leq m$, and $s \in D_i$, and estimating the probability that i is the smallest index such that $s \in D_i$. By iterating this step $O(m)$ times one gets a tight estimate on the the overlap between the sets, thus obtaining an estimate on the cardinality of the union.

Theorem 4 *[4] There is a Monte-Carlo algorithm that computes an ϵ, δ approximation of the cardinality of the union of m sets in*

$$(8 * (1+\epsilon) * m\ln(3/\delta))/((1-\epsilon^2/8)\epsilon^2)$$

steps.

Assume first that all communications are fed by on-off sources with identical parameters. In our formalization, instead of sets of elements we have m events, $E_1, \ldots, E_m$ where the event E_i is "edge i overflows". An event is a collection of states, where a *state* is an on or off setting for all the logical channels (communications) in the system. The event E_i contains all the states that overflow edge i. Instead of estimating the cardinality of the union of sets we need to estimate the probability of the union of m events, where different states have different probabilities. That is we are interested in estimating Q, the network overflow probability which can be viewed as the probability that in a randomly chosen state (according to the probability distribution of the connections) some edge will overflow. We would like to calculate an (ϵ, δ) approximation of Q which we will denote by $\tilde{Q}$. (Henceforth, a tilde on top of a value denotes an estimate of that value.)

The static algorithm is given in Figure 1. Let p_i be the probability that edge i overflows. Let $P = \sum_{i=1}^{m} p_i$. One *trial* in our algorithm is choosing a pair (i, s) with probability $p(s)/P$, and estimating the probability that i is the smallest index edge such that state s overflows edge i. We estimate this by uniformly choosing an edge at random and checking whether it overflows (steps 4.3.4 and 4.3.5 of Figure 1). The costly step in a trial is checking for overflow in an edge. The tricky point is selecting a random state with the appropriate probability. Let $C_i = \{c_1, \ldots, c_{|C_i|}\}$ be the set of connections going through edge i. Given an edge i we choose a state that overflows that edge using the algorithm *choose* given below:

(Let $\bar{\mathcal{E}}$ denote the complement of the event $\mathcal{E}$)
Algorithm *choose*: choosing an overflow state s with probability $\frac{\mathbf{Pr}[s]}{p_i}$

1 Set all connections not belonging to edge C_i
  to "on" with probability q
  and "off" with probability $1 - q$
2 Let $\mathcal{F}_0$ be the event "edge i overflows"
3 **for** $k = 1$ **to** $|C_i|$ **do**
  3.1 Let $\mathcal{E}_i$ be the event "c_k is on"
  3.2 Set c_k to "on" with probability
    $b_i = \mathbf{Pr}\{\mathcal{E}_k | \mathcal{F}_{k-1}\}$
    and to "off" with probability $1 - b_i$.
  3.3 **if** c_k is set to "off" **then** $\mathcal{F}_k = \bar{\mathcal{E}}_k \cap \mathcal{F}_{k-1}$
  3.4 **else** $\mathcal{F}_k = \mathcal{E}_k \cap \mathcal{F}_{k-1}$

Lemma 1 *The choose algorithm chooses an overflow state s with the correct probability.*

Proof: Connections outside C_i are clearly independent of the event "edge i overflows". Connections in C_i are fixed using the appropriate conditional probabilities. □

The choose algorithm can be implemented in $O(n)$ time by pre-computing the conditional probabilities and using a look up table.

Theorem 5 *The run-time of the static algorithm given in Figure 1 is $O(nm\epsilon^{-2}\log\delta^{-1})$ and it produces an (ϵ, δ) approximation of the network overflow probability.*

Proof: We show that $E[Y_t] = Q$.
Let s be an overflow state of the network.
Let $C(s) = \{(s, i) : \text{edge } i \text{ overflows in state } s\}$. Let $R_k = \{s : |C(s)| = k\}$. Let $r_k = \sum_{s \in R_k} \mathbf{Pr}(s)$. Then, $P = \sum_{k=1}^{m} k * r_k$ and $Q = \sum_{k=1}^{m} r_k$. Now, $E[Y_t] = E[t(s)] * P/m$. Since the random variable $t(s)$ depends only on k, $t(s)$ is geometrically distributed with mean m/k.
Hence, $E[Y_t] = \sum_{k=1}^{m} \mathbf{Pr}(s \in R_k) * P/k$. Steps 4.1 and 4.2 choose a state s with probability $Pr[s]/P$. Hence, the probability that in a trial a state $s \in R_k$ is chosen is $k * r_k/P$, and we have $E[Y_t] = Q$. The random variables $Y_1, \ldots, Y_{N_T}$ are independent and identically distributed with mean Q. The number of steps needed to get an (ϵ, δ) approximation follows due to theorem 4 proved in [4]. □

The Static Algorithm

1 $gtime = 0$ /* $gtime$ counts the global number of steps executed */
2 $N_T = 0$ /* counts the number of trials */
3 $T = (8 * (1+\epsilon) * m \ln(3/\delta))/((1-\epsilon^2/8)\epsilon^2)$
4 **trial:**
  4.1 randomly choose $i \in \{1, \ldots, m\}$ such that i is chosen with probability p_i/P
  4.2 choose a state s with probability $Pr[s]/p_i$ using the choose algorithm
  4.3 $t(s) = 0$ /* $t(s)$ is the *step number*, counts the number of steps in this trial */
  step:
    4.3.1 $t(s) = t(s) + 1$
    4.3.2 $gtime = gtime + 1$
    4.3.3 **if** $gtime > T$ **then go to** 5
    4.3.4 randomly choose $j \in \{1, \ldots, m\}$ with probability $1/m$
    4.3.5 **if** j does not overflow in s **then go to step**
  4.5 $N_T = N_T + 1$
  4.6 $Y_t = P * t(s)/m$
  4.7 **go to trial**
5 $\tilde{Q} = \frac{\sum_{t=1}^{N_T} Y_t}{N_T}$

Figure 1: Static Algorithm for estimating system overflow probability

We can use the static algorithm even when connections are arbitrary on-off random variables by using Hui's theorem (theorem 1) as an upper bound on the overflow probability. For example, we can choose b in the above theorem to be the bandwidth capacity of the edge and a to be $2b$. Then Hui's theorem gives an upper bound on the probability that twice the bandwidth capacity will be exceeded. Then our Monte-Carlo algorithm will find an overall network estimate of this upper bound. To choose a state s with the appropriate probability in the choose algorithm we again use Hui's theorem when calculating the conditional probabilities.

3 The Dynamic Algorithm

We consider now the dynamic problem in which the network protocol needs to react sequentially to a sequence of add and delete requests. For each add request, the protocol needs to decide if adding that request violates the system's QoS requirement. If the requirement is not violated a new logical channel is added, satisfying that request. The goal is to use past estimates in order to minimize the work of each evaluation. Assume that the change in the network (add or delete) involves a subset $F \subseteq E$ of edges, where E is the total number of edges in the network. Since the static algorithm requires $O(nm \log m)$ steps [1] to check for QoS guarantee without prior information, we are looking for an incremental algorithm that can check the same in $|F|/m$ of that complexity or $O(n|F|\log m)$ steps. Since error probabilities accumulate, we will need to compute a new estimate for the whole network after a long sequence of changes, however, the amortized complexity will remain $O(n|F|\log m)$ steps per addition or deletion.

Let Q and Q' denote the overflow probability of the network before and after the connection was added. Let $\tilde{Q}$ and $\tilde{Q}'$ denote their respective estimates. Let Q_F and Q'_F denote the probability that an edge in F overflows in a randomly chosen state before and after the change. We also define Q_{E-F} to be probability that *only* an edge in $E - F$ over-

[1] From now on, we assume that δ is $1/m^c$ for some $c \geq 1$ to guarantee estimation with high probability, ϵ is a fixed constant.

flows in a randomly chosen network state. Clearly

$$Q = Q_F + Q_{E-F} \tag{2}$$

And since the change involves only edges in F

$$Q' = Q'_F + Q_{E-F} \tag{3}$$

Let W be the probability that a random state s that overflows before the change, overflows *only* edges in $E - F$, then $Q_{E-F} = WQ$. Thus, instead of estimating Q' directly (which requires $O(nm \log m)$ time) we can estimate Q'_F and W, focusing only on $|F|$ edges. The *incremental algorithm* for estimating the overflow probability after a change that involves the set of edges F is given in Figure 2. In step 3.1.1 of this algorithm, we have to randomly choose an overflow state s of the network with probability $\mathbf{Pr}[s]/\tilde{Q}$. This can be done using the following algorithm *global-choose*. This is essentially like the choose algorithm, but chooses a state s such that some edge overflows in the *whole* network with the appropriate probability.

(Let $\bar{\mathcal{E}}$ denote the complement of the event $\mathcal{E}$)
Algorithm *global-choose*: choosing an overflow state s with probability $\frac{\mathbf{Pr}[s]}{\tilde{Q}}$

1 Let $\mathcal{F}_0$ be the event "some edge in the network overflows" ($\mathbf{Pr}\{\mathcal{F}_0\} = \tilde{Q}$)
2 **for** $i = 1$ **to** m **do**
  2.1 Let $\mathcal{E}_i$ be the event "edge E_i overflows"
  2.2 Set edge E_i to overflow with probability $b_i = \mathbf{Pr}\{\mathcal{E}_i|\mathcal{F}_{i-1}\}$
    and *not* to overflow with probability $1 - b_i$.
  2.3 **if** E_i is set to overflow **then**
    2.3.1 Choose an overflow state s using the choose algorithm
    2.3.2 stop
  2.4 **else**
    2.4.1 Set the connections belonging to edge E_i such that the edge does *not* overflow similar to step 3 of the choose algorithm (i.e. do not set the status of connections not belonging to E_i)
    2.4.2 $\mathcal{F}_i = \bar{\mathcal{E}}_i \cap \mathcal{F}_{i-1}$

The main result of this section is the following theorem.

Theorem 6 *The incremental algorithm (Figure 2) satisfies definition 3 and takes $O(n|F| \log m)$ steps, where n is the total number of connections in the network.*

Proof: It is easy to see that $E[Z_k] * \tilde{Q} = Q_{E-F}$. If we proceed to step 3, we compute Q' with ϵ accuracy (step 4). On the other hand, if we skip step 3, then because of the condition, $Q'_F \geq \frac{\epsilon Q}{2}$ whp. Since $Q' \leq Q + Q'_F$, we still maintain an ϵ accuracy.
To show that running time of the algorithm is $O(n|F| \log m)$ we note that step 1 takes $O(n|F| \log m)$ time. Step 3 takes $O(n|F| \log m)$ time because we have to check for overflow only in edges belonging to F (in step 3.1.2). The upper bound we choose for the number of trials needed to estimate W is an application of the zero-one estimator theorem of [4]. Since W is lower bounded by $\epsilon/2$ whp, it is enough if we perform only $8 \ln(2/\delta)/(\epsilon)^3$ trials. $\square$

We notice that in the incremental algorithm we "lose" a small constant factor in the confidence probability in each call to the algorithm. This is because our estimate in the incremental algorithm (step 2.1 or step 4) is an addition of two terms. Hence if the error in each estimate is δ, the error of the total estimate adds up to 2δ. To maintain an ϵ approximation whp, we run all calls of the incremental algorithm with $\delta = 1/(2m^3)$. Every m^2 calls we re-evaluate Q using the static algorithm (with $\delta = 1/(2m^3)$). Since a call to the static algorithm takes $O(mn \log m)$ time, the amortized work done for any addition or deletion of a path of F edges is still $O(n|F| \log m)$.

4 Applications to ATM Networks

We briefly mention how QoS requirements are handled in practice in some ATM networks and point out how our techniques of accurately estimating the error probabilities can be incorporated to improve network throughput.

Practical ATM networks are essentially modeled as a complete graph G (of say, m edges and N terminals). Each edge can be thought of as a *Virtual Path* (VP) and connections are basically *Virtual Circuits* (VC's) ([2], [5]). When establishing a VC so that a pair of terminals can communicate, a

Incremental Algorithm
(Q_0 is a parameter fixed by QoS.
$\tilde{Q}$ is the current ϵ estimate of the network overflow probability.
$\tilde{Q}'$ is the new estimate after a connection is added or deleted.)

1 Estimate the overflow probability $\tilde{Q'_F}$ for the set F
using the algorithm of Figure 1, with the parameters $(\epsilon/2, \delta)$.
2 **if** $\tilde{Q}'_F \geq (\epsilon/2)\tilde{Q}$ **then**
2.1 $\tilde{Q}' = \tilde{Q} + \tilde{Q}'_F$
2.2 **go to** 5
else go to 3
3 Let W be the probability that a randomly chosen network state overflows
(before the new connection was added or deleted) *only* in $E - F$ and not in F.
The following steps compute an (ϵ, δ) approximation $\tilde{W}$ for W:
3.1 **for** $k = 1$ **to** $N = (8\ln(2/\delta)/(\epsilon)^3)$ **do**
3.1.1 randomly choose an overflow state s with probability $Pr[s]/\tilde{Q}$
using the algorithm *global-choose*
3.1.2 **if** any edge in F overflows **then** set $Z_k = 0$
3.1.3 **else** set $Z_k = 1$
3.2 **endfor**
3.3 $\tilde{W} = \sum_k Z_k/N$
4 $\tilde{Q}' = \tilde{Q}'_F + \tilde{W} * \tilde{Q}$
5 **if** $\tilde{Q}' < Q_0$ **then** "accept request"
else "reject request"

Figure 2: Incremental Algorithm for determining QoS guarantee after a connection is added or deleted

route consisting of a set of VP's is selected. Following the practices of dynamic routing in telephone networks ([2], [5]), routes that consist of more than *two* hops are excluded. If a route consists of one VP, it is a *direct* route; otherwise, it is an *alternative route.*

To provide QoS guarantees, we suppose that the fraction of cells (which are simply packets in ATM network terminology) lost is not permitted to exceed a given fraction ρ.[2] Let $p_j(l)$ denote the fraction of cells lost in VP_j of the network, when l VCs are being routed through VP_j. We assume that these single link probabilities can be determined by cell level analysis [2] or by simulations [11]. We classify a VC as either a VC_j or a VC_{ij} if the VC is assigned route $\{j\}$ or route $\{i, j\}$, respectively. In an alternatively routed VC, cell loss can occur at either of the two VPs.

In [2], the *QoS-permissibility conditions* for setting up a new virtual circuit are as follows:

1. **direct route** j: (a) $p_j(l_j + 1) \leq \rho$ and
(b) For every VP_k such that a VC_{jk} is in progress, $p_j(l_j + 1) + p_k(l_k) \leq \rho$.
Notice that the first of these conditions ensures that the additional VC will not cause cell loss to be excessive for any of the directly routed VCs on VP_j. The second condition ensures that cell loss will not be excessive for any of the "overlapping VCs" that is, the alternatively routed VCs employing VP_j.
2. **alternate route** $\{i, j\}$: (a) $p_i(l_i+1)+p_j(l_j+1) \leq \rho$ and
(b) For every VP_k such that a VC_{ij} is in progress, $p_i(l_i+1)+p_k(l_k) \leq \rho$; and for every VP_k such that a VC_{jk} is in progress $p_j(l_j+1)+p_k(l_k) \leq \rho$.

Notice that QoS-permissibility of any route involves examining VPs that are not on the route un-

[2] A typical value of ρ ranges from 10^{-6} to 10^{-9}.

der consideration. This is similar to the connection-based overflow constraint mentioned in section 1.1. Applying our method to the two-link subgraphs in cases 1(b), 2(a) and 2(b), we get accurate estimates of the overflow probability, instead of just summing the overflow probabilities which can be a gross over estimation of the loss probability (especially when the two edges under consideration share a lot of common connections). The incremental algorithm is especially attractive, since a connection spans only two edges, which means $O(n \log m)$ running time. We are presently conducting simulations to study the effectiveness of our methods in a variety of network and load settings.

References

[1] A.Elwalid and D.Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks. *IEEE/ACM Trans. Networking*, **1**(3), 1993, 329-343.

[2] S. Gupta, K. Ross and M. Zarki. On Routing in ATM Networks. In [8].

[3] J.Y. Hui. Resource Allocation for Broadband Networks. *IEEE J. Selected Areas in Comm.*, **6**, 1988, 1598-1608.

[4] R.M. Karp, M.G. Luby and N. Madras. Monte-Carlo Approximation Algorithms for Enumeration Problems. *Journal of Algorithms*, **10**, 1989, 429-448.

[5] F.P. Kelly, Modelling Communication Networks, Present and Future, Proc. R. Soc. Lond. A (1995) **444**, 1-20.

[6] F.P. Kelly, Notes on Effective Bandwidths, *Stochastic Networks: Theory and Applications*, Vol. 4, Royal Statistical Society Lecture Notes Series, Oxford University Press (1996), 141-168.

[7] J. Kleinberg, Y. Rabani and E. Tardos. Allocating Bandwidth for Bursty Connections, In *Proceedings of the 29th ACM STOC*, 1997, 664-673.

[8] M.E. Steenstrup (editor). *Routing in Communications Networks*, Prentice Hall, 1995.

[9] M. Taqqu, W. Willinger and R.Sherman. Proof of a Fundamental Result in Self-Similar Traffic Modeling. *Computer Communication Review*, **27**, 1997, 5-23.

[10] L. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, **8**, 1979, 410-421.

[11] Q. Wang and V.S. Frost. Efficient Estimation of Cell Blocking Probability for ATM Systems, *IEEE/ACM Transactions on Networking*, 1993 **1**(2), 230-235.