Static and Dynamic Evaluation of QoS Properties

Gopal Pandurangan* Eli Upfal*

June 20, 2001

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 procedure is to accommodate as many communication requests as possible while keeping the failure (e.g. overflow) probability bounded by a pre-specified 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 than to 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. A preliminary version of this paper appeared in the proceedings of the 31st ACM Symposium on Theory of Computing (STOC), Atlanta, 1999.

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 cell loss, cell delay and cell jitter [3]. The goal (on the part of the communication provider) is to satisfy as many communication requests as possible while maintaining pre-specified QoS guarantees. In this paper we focus on cell loss failures due to link or buffer overflow. Our method can be modified to handle other statistically governed communication characteristic such as cell delay and jitter.

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* [7]. 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 [7, 10, 8] as well as in simulation studies to evaluate performance of routing algorithms in ATM networks [3]. The work in [10] 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 [2].

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 [11] 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 [3].) 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 give the first non-trivial algorithms for estimating QoS properties in statistically multiplexed networks. Our algorithms employ efficient Monte-Carlo

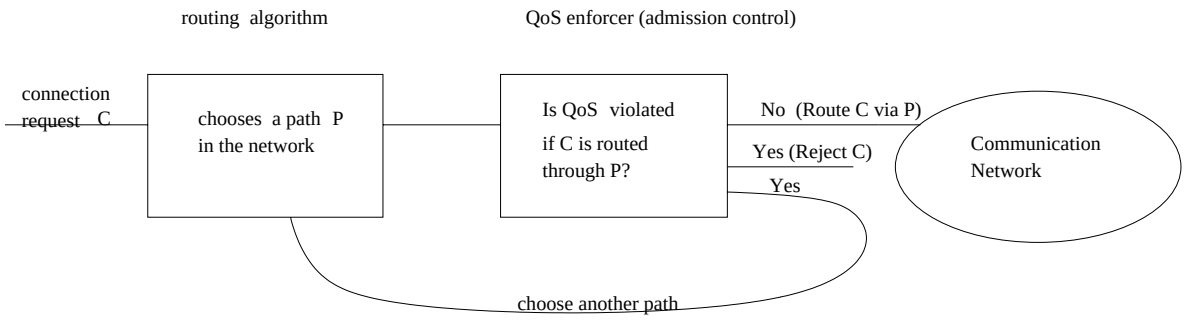

Figure 1: A schematic to illustrate how a connection is handled

method to accurately estimate the overflow probability of any set of channels in an arbitrary topology network. These estimates translate into better admission control policy that achieve higher bandwidth utilization while guaranteeing same quality of service. We study this problem under *static* and *dynamic* settings. In the static setting, we assume that a set of connections are already active (routed) in the network and we are interested in whether QoS requirements are violated. In the dynamic setting, we are interested in admitting a new connection to the network and we desire a fast admission control algorithm to decide whether to admit this connection subject to QoS requirement.

Figure 1 illustrates the procedure followed by a typical network management protocol when admitting a new connection. When a new connection request, say C arrives, the routing algorithm (for example, an algorithm used in telephone networks is the Dynamic Routing Algorithm (DAR) [9]) selects a path in the network, say P where this connection might be routed. Then the admission control algorithm decides whether routing this connection through P would violate QoS guarantee. If QoS requirement is violated, then either C is rejected or (as in the DAR) an alternative route might be selected for P and the QoS guarantee condition is again checked. (In DAR the connection request is denied if there is a violation for the second time). If QoS guarantee is met, then connection request is admitted to the network. It is useful to think of admitting a connection in this way as it separates routing and admission control, although they might be implemented as a single algorithm [3]. In this paper we focus only on admission control. We assume that when the admission control protocol is invoked a path had already been selected by the routing algorithm.

1.1 ATM Networks and QoS Guarantees

To demonstrate the use of our estimation algorithms to achieve efficient admission control we focus in this section on its application to ATM networks.

ATM is emerging as a standard for Broadband Integrated Services Digital Networks (B-ISDN) carrying a wide spectrum of new consumer services such as video-on-demand, video teleconferencing etc. Specifically, ATM is a *high-speed, virtual-circuit-oriented packet-switching technique* that uses short, fixed-length packets called *cells*. A *virtual circuit* in an ATM network is a contract between the network and the customer to deliver traffic of specified *quality of service* (QoS).

Practical ATM networks are essentially modeled as a complete graph G of say, m edges and N nodes (terminals). Nodes are endpoints of a communication request and edges are communication links with specific bandwidth limits. A connection or a communication request is a (source, destination) pair with bandwidth requirements. Each edge in G can be thought of as a *Virtual Path* (VP) and connections are *Virtual Circuits* (VC's) ([3], [6]). When establishing a VC such that a pair of terminals can communicate, a route consisting of a set of VP's is selected. Following the practices of dynamic routing in telephone networks ([3], [6]), 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 lost is not permitted to exceed a given fraction ρ.[1] Let $p_j(l)$ denote the fraction of cells lost in the input buffer of 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 [1] or by simulations [12]. We classify a VC as either a VC_j or a VC_{ij} if the VC is assigned one-edge route $\{j\}$ or two-edge route $\{i, j\}$, respectively. In an alternatively routed VC, cell loss can occur at either of the two VPs. Notice that

$$\Pr(\text{“cell lost on a } VC_{ij}\text{”} | l_i, l_j) \leq p_i(l_i) + p_j(l_j)$$

that is, the probability that a cell of a VC_{ij} is lost when there are l_i VCs using VP_i and l_j VCs using VP_j is less than the sum of two VP loss probabilities.

In [3], 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_{ik} 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 $p_j(l_j + 1) + p_k(l_k) \leq \rho$.

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

Notice that QoS-permissibility of any route involves examining VPs that are not on the route under consideration. In the above permissibility conditions in cases 1(b), 2(a) and 2(b) instead of just summing probabilities which can be a gross over estimation of the overall loss probability (especially when the two edges under consideration share a lot of common connections) we can use the methods developed in this paper to accurately estimate this probability in these cases. Thus one of the main contributions of this paper is to efficiently and accurately estimate such loss or overflow probabilities when we have arbitrary pattern of connections in an arbitrary topology network.

While we focus here on the problem of bounding the probability of an edge overflow in a given set of edges in the network (i.e. a subgraph of the network) our technique can be very easily adapted to address weaker QoS guarantees such as the *connection-based overflow constraint* [8]. A connection-based overflow constraint bounds the probability of an overflow along a given logical link. We can model overflow constraints in a network in two main ways: the *link-based overflow constraint* and the *connection-based overflow constraint* [8]. More formally, let the random variable X_i be the demand corresponding to connection c_i. Let P_i represent the *route* of the connection c_i, that is the edges of the path of c_i. The link-based overflow constraint requires that for each edge e_j, we have $\Pr[\sum_{i:e_j \in P_i} X_i > B_j] \leq Q_0$, where Q_0 is the overflow probability allowed by QoS. On the other hand, the connection-based overflow constraint requires that for each connection c_i we have $\Pr[\exists e_j \in P_i : \sum_{i:e_j \in P_i} X_i > B_j] \leq Q_0$. ¿From the perspective of providing guaranteed quality of service to users in a network, the connection-based overflow constraint is more natural. Both these constraints can be enforced by our method of accurately estimating the overflow probability in an arbitrary subnetwork. For example, bounding the probability that an edge overflows in a suitable subgraph of the network enforces the connection-based overflow constraint. Under the QoS permissibility conditions mentioned earlier (which enforce the connection-based overflow constraint) we have to check for overflow condition in two-link subgraphs.

1.2 Problem Statement and Related Work

We consider a communication network $G = (V, E)$ with physical channels (edges) denoted by the set $E = \{e_1, e_2, \ldots, e_{|E|}\}$. For each channel e_i we have a constant B_i specifying the maximum *bandwidth* of that channel. The data flow in a *logical* channel, or a *communication request* or *connection* is characterized by a distribution function specifying the probability that the channel sends a given amount of data at a given time step. Thus a communication network consists of the physical network G along with the connections active in the network. A *subnetwork* is simply a subset S of E along with the connections that are active in edges belonging to S. For

our purposes, we simply model a communication request as a random variable. Our 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 on a given set of edges in the network (i.e. a given subnetwork).

Consider first a simple scenario in which all logical channels are fed by identical[2] (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 for example, is significantly harder and can be shown to be $\#P$-complete by a straightforward reduction from the union of sets problem [5]. 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 [8].

In practice, the complexity of computing the overflow on one edge is circumvented by using the method of *effective bandwidth* [4, 7, 2]:

Definition 1.1 *The effective bandwidth of a random variable X is*

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

The significance of the above definition will be clear from the following theorem which gives a conservative estimate of the overflow probability in the following sense: If the sum of the effective bandwidths of a set of independent connections does not exceed the link capacity, then the probability that the sum of their transmission rates exceeds twice the capacity at any instant is at most p. Given the effective bandwidth of individual logical channels a bound on the overflow probability of a given edge can be computed using the following result due to Hui[4]:

Theorem 1.1 *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 $\Pr\{X \geq a\} \leq p^{a-b}$.*

Proof: The proof is a simple application of Markov's inequality. From, $\sum_i \beta_p(X_i) \leq b$ we have $\sum_i \log E[p^{-X_i}] \leq \log p^{-b}$. Hence $\Pi_i E[p^{-X_i}] \leq p^{-b}$. Since X_i are independent, $\Pi_i E[p^{-X_i}] = E[\Pi_i p^{-X_i}] = E[p^{-\sum_i X_i}] = E[p^{-X}] \leq p^{-b}$. Now by Markov's inequality, $\Pr\{X \geq a\} = \Pr\{p^{-X} \geq p^{-a}\} \leq p^a E[p^{-X}] \leq p^{a-b}$. □

[2]Identical or almost identical connections capture ATM networks supporting *homogeneous sources* [3]. In ATM terminology this is referred to as a VP subnetwork which is a collection of VCs transporting identical sets of traffic sources i.e. with same traffic characteristics and QoS requirements. Statistical multiplexing is more effective for homogeneous traffic sources. See [3] for more details.

1.3 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 a given set of edges in a network which is the probability that *some* edge will overflow among the set of edges (a more precise definition is given in section 2. More specifically we are interested in two versions of the problem:

1. **The Static Problem:** Given a set of edges S in a network and a set of logical channels estimate the overflow probability of the set S in the network.

2. **The Dynamic Problem:** Given a network (or a subnetwork), a set of logical channels, and a sequence of add and delete communication requests, determine for each request if granting that communication request violates a predetermined bound on network overflow probability in the network (or subnetwork).

Let m be the number of physical links in a subnetwork S, n be the number of active logical links (connections) in S (i.e. the connections that use any of the edges in S). 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 overflow probability in S.

Definition 1.2 *An (ϵ, δ) Monte-Carlo approximation for Q is a value $\tilde{Q}$ such that*

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

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

Theorem 1.2 *There is a Monte-Carlo algorithm that computes an (ϵ, δ) approximation of the probability of overflow of a subnetwork S 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.

For the dynamic setting we define an *incremental* version of an ϵ approximation for determining QoS guarantee whp.

Definition 1.3 *Let Q_0 be the pre-defined QoS failure probability for a network (or subnetwork). Let Q' be the exact failure probability of the network (or subnetwork) after 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 $Q' \geq (1-\epsilon)Q_0$, or accepts the new request when $Q' \leq (1+\epsilon)Q_0$.*

Theorem 1.3 *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 [5].

We will first briefly explain the algorithm for the union of sets problem. In the union of sets problem we are given m sets $D_1, ..., D_m$ and the goal is to estimate $|\cup_{i=1}^{m} D_i|$. The idea is to estimate the union by a Monte-Carlo sampling method as follows. Sample a pair (i, s), where $1 \leq i \leq m$, and $s \in D_i$, with probability $1/\sum_{i=1}^{m} |D_i|$ and compute the fraction of sets that contain s. By iterating this step $O(m)$ times one gets a tight estimate of the "overlap" between the sets, thus obtaining an estimate of the cardinality of the union. For the union of sets problem Karp and Luby prove the following theorem.

Theorem 2.1 *[5] 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.*

We will now describe the static algorithm. We will assume a communication network (graph) $G = (V, E)$ where all communications are fed by on-off sources with identical parameters, i.e. (q, s) on-off sources.

We will focus on a subnetwork $S \subseteq E$. [3] Let the number of edges in the subnetwork be m. In our formalization, instead of sets of elements we have m events, $\mathcal{E}_1, \ldots, \mathcal{E}_m$ where the event $\mathcal{E}_i$ is "edge $i \in E$ overflows". An event is a collection of states, where a *state* is defined as follows.

Definition 2.1 (state of the network) *Assume a network where all communications are fed by on-off sources. A state of the network is an on-off setting for all sources.*

For example a network fed by three on-off sources can be in state $s = (1, 0, 1)$, meaning that the first and third sources are in the "on" state, while the second state is in an "off" state. Given a state s, we denote its probability as $\Pr[s]$, which is simply the probability that the network will be in state s. Further, because we

[3] Of course S can be E itself, but to emphasize the full generality of the algorithm we describe the algorithm in terms of an arbitrary subnetwork. This also makes clear the fact that the algorithm does not depend on any way on the topology of the network or subnetwork, but only on the number of edges in it.

have assumed independence of sources, the above probability is the product of the probabilities of the individual sources being in their respective states. In the example above, $\Pr[s] = q^2(1-q)$, assuming that all sources are identical and independent (q, s) on-off sources.

Thus the event $\mathcal{E}_i$ contains all the states (which we can think of as atomic events) that overflow edge i in S. 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 subnetwork overflow probability which can be viewed as the probability that in a randomly chosen subnetwork state (according to the probability distribution of the connections) *some* edge in the subnetwork 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 in the (ϵ, δ) sense.)

The static algorithm is given in figure 2. As is common in Monte-Carlo algorithms, the main idea is to set up a random variable (in our case Y_t) whose expectation is equal to the desired parameter (here Q) and then estimating the expectation by sampling from this random variable. For polynomial running time we have to show that we need only a polynomial number of samples to get an accurate estimate with high probability.

We now describe the algorithm informally and show how we set up the random variable Y_t. In theorem 2.2, we show formally that this random variable correctly estimates the required parameter Q. Let p_i be the probability that edge i overflows. This probability can be calculated exactly as we have assumed identical (q, s) on-off sources and the overflow probability in a single edge can be computed by the binomial distribution. Let $P = \sum_{i=1}^{m} p_i$. The algorithm consists of many *trial* steps which is iterated repeatedly till we get an (ϵ, δ) estimate of Q. One *trial* in the static algorithm (which is step 4 in figure 2) is choosing a pair (i, s), where $i \in E$ and $s \in \mathcal{E}_i$ (i.e. s is a state in which edge i overflows) with probability $\Pr[s]/P$, and estimating the fraction of number of edges that overflow in state s (this is similar to the "overlap" between sets mentioned in paragraph 2) In step 4.3 (consisting of 5 sub-steps), we compute the number of overflow edges in a fixed state s. If we do this step in a straightforward way, that is checking for overflow in each edge one by one, one trial itself will take $O(m)$ steps, as in the worst case we have to check every edge in S. Instead we resort to a Monte-Carlo sampling to estimate the number of overflow edges. This can be done simply by sampling edges uniformly at random and checking for overflow (step 4.3.4).

We will now mention how we choose a pair (i, s) at the beginning of the trial step. This is done in two steps. In step 4.1 we first choose an edge $i \in E$ with probability p_i/P and then choose an *overflow* state $s \in \mathcal{E}_i$ with probability $Pr[s]/p_i$ in step 4.2. We use the *choose* algorithm in selecting a random overflow state $s \in \mathcal{E}_i$

with the appropriate probability i.e. $\Pr[s]/p_i$. 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 a state s with probability $\frac{\Pr[s]}{p_i}$

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 $\mathcal{E}_k$ be the event "c_k is on"
3.2 Set c_k to "on" with probability $b_k = \Pr\{\mathcal{E}_k | \mathcal{F}_{k-1}\}$
and to "off" with probability $1 - b_k$.
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 2.1 *The choose algorithm chooses an overflow state s with the probability* $\Pr[s]/p_i$.

Proof: Connections outside C_i are clearly independent of the event "edge i overflows". Hence these connections can be set to "on" with probability q and to "off" with probability $1-q$. Connections belonging to C_i are set to "on" or "off" with probabilities as calculated in the choose algorithm. Suppose s is a state in which edge i overflows. Let $s(c_i)$ be the status ("on" or "off") of connection c_i in state s and let $\Pr[\mathcal{E}_{s(c_i)}]$ denote the probability of the event that c_i is in state $s(c_i)$. Then the probability that a connection c_i is set to $s(c_i)$ is $\Pr[\mathcal{E}_{s(c_i)} | F_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{i-1})}]$. By the rule of conditional probability we have,

$$\Pr[\mathcal{F}_0] \times \Pr[\mathcal{E}_{s(c_1)} | \mathcal{F}_0] \times \Pr[\mathcal{E}_{s(c_2)} | \mathcal{F}_0 \cap \mathcal{E}_{s(c_1)}] \times \cdots \times \Pr[\mathcal{E}_{s(c_{|C_i|})} | \mathcal{F}_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{n-1})}] = \Pr[\mathcal{F}_0 \cap \mathcal{E}_{s(c_1)} \cap \cdots \cap \mathcal{E}_{s(c_{|C_i|})}] = \Pr[s]$$

since in state s the edge overflows. Thus probability that state s is chosen is $\frac{Pr[s]}{p_i}$ since $\Pr[\mathcal{F}_0] = p_i$. □

The conditional probabilities in the choose algorithm can be calculated using the Bayes' rule as follows. For example the probability that a connection c is "on" in a randomly chosen overflow state of a particular edge i is given by Bayes' rule as:

$$\Pr\{c\ is\ on | i\ overflows\} = \frac{\Pr\{i\ overflows | c\ is\ \text{“}on''\} * \Pr\{c\ is\ \text{“}on''\}}{\Pr\{i\ overflows\}}$$

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

In the following theorem we show that $E[Y_t]$ is Q. Also the running time is upper bounded by N which is polynomial in $m, 1/\epsilon$ and $\log(1/\delta)$.

Theorem 2.2 *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 overflow probability in the subnetwork S.*

Proof: We show that $E[Y_t] = Q$ for any index t. Let s be an overflow state in S. Let $C(s) = \{(s,i) : \text{edge } i \text{ overflows in state } s\}$. Let $R_k = \{s : |C(s)| = k\}, k = 1,\ldots,m$. That is R_k is the set of all overflow states in which exactly k edges overflow. Let $r_k = \sum_{s\in R_k} \Pr(s)$. Then, $P = \sum_{k=1}^{m} k * r_k$ and $Q = \sum_{k=1}^{m} r_k$. This is because $P = \sum_{i=1}^{m} p_i$ is the sum of the probabilities of all overflow states s with s being counted k times if $s \in R_k$ (once for each edge). On the other hand Q is simply the sum of probabilities of all overflow states s. 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, $E[Y_t] = \sum_{k=1}^{m} \Pr(s \in R_k) * P/k$. 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 $Y_1, \ldots, Y_t$ are independent and identically distributed with mean Q. We obtain an estimate of the mean by simply averaging over these t random variables. The number of steps needed to get an (ϵ,δ) approximation follows due to theorem 2.1 proved in [5]. □

We can use the static algorithm even when connections are arbitrary on-off random variables by using Hui's theorem (theorem 1.1) 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 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 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. 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

The Static Algorithm
(m is the size of the subnetwork)

1 $gtime = 0$ /* $gtime$ counts the global number of steps executed */
2 $t = 0$ /* counts the number of trials */
3 $N = (8 * (1 + \epsilon) * m \ln(3/\delta))/(1 - \epsilon^2/8)\epsilon^2$
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** $gtime > N$ **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 $t = t + 1$
    4.6 $Y_t = P * t(s)/m$
    4.7 **go to trial**
5 $\tilde{Q} = \frac{\sum_{j=1}^{t} Y_t}{t}$

Figure 2: Static Algorithm for estimating overflow probability in a subnetwork

is the total number of edges in the network. In this section for simplicity, we focus on estimating the overflow probability of the entire network. The same analysis carries over if we focus on the overflow probability of any subnetwork. Since the static algorithm requires $O(nm \log m)$ steps [4] 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 $O(n|F| \log m)$ work 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 $E - F$ overflows in a randomly chosen network state. Clearly

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

[4] 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. Also $|E|$, the size of the network is assumed to be m.

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 3.

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.
Fix $\epsilon' = \frac{\epsilon^2}{2(1+\epsilon)}$ and $\delta' = 1/(3m^3)$. $FLAG$ is a global logical variable)

1 Estimate the overflow probability $\tilde{Q'_F}$ for the set F (after the new connection was added) using the algorithm of Figure 1, with the parameters (ϵ', δ').
If $\frac{\tilde{Q'}_F}{1-\epsilon'} \leq \frac{\epsilon\tilde{Q}}{2(1+\epsilon)}$ **then**
  $\tilde{Q'} = \tilde{Q}$
  set $FLAG =$ "true"
  exit
else set $FLAG =$ "false" and go to step 2.
2 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:
2.1 **for** $k = 1$ **to** $N = \frac{2+\epsilon}{\epsilon-\epsilon'} * (4 \ln(2/\delta')/(\epsilon')^2)$ **do**
    2.1.1 randomly choose $i \in \{1, \ldots, m\}$ such that i is chosen with probability $p_i/\tilde{Q}$
    2.1.2 choose a state s with probability $Pr[s]/p_i$
    (The above two steps choose an "overflow" state s with probability $Pr[s]/\tilde{Q}$)
    2.1.3 **if** any edge in F overflows **then** set $Z_k = 0$
    2.1.4 **else** set $Z_k = 1$
2.2 **endfor**
2.3 $\tilde{W} = \sum_k Z_k/N$
3 $\tilde{Q'} = \tilde{Q'}_F + \tilde{W} * \tilde{Q}$
4 **If** $\tilde{Q'} \leq Q_0$ **then**
    "QoS is not violated: admit connection request"
else
    "QoS is violated: reject connection request"

Figure 3: Incremental Algorithm for determining QoS guarantee after a connection is added or deleted

The main result of this section is the following theorem.

Theorem 3.1 *The incremental algorithm of Figure 2 satisfies definition 1.3 and takes $O(n|F|\log m)$ steps, where n is the total number of connections in the network.*

Proof: We will show that the algorithm gives an ϵ approximation of Q'. The idea used is as follows. Since we use the formula $\tilde{Q'} = \tilde{Q'}_F + \tilde{W} * \tilde{Q}$, we lose some accuracy because the second term is in the form of a product and we have assumed that we have an ϵ estimate of Q. To compensate for this loss, we estimate Q'_F with a greater accuracy, which can be done if Q'_F is not "too small". On the other hand, if Q'_F is "very small" we can directly bound Q'.

More precisely, we have an ϵ approximation of Q and ϵ' approximations for Q'_F and W. This gives rise to eight different cases. We will analyze the most interesting among them. We will show that

$$(1-\epsilon)Q' \leq \tilde{Q'} = (1+\epsilon')Q'_F + (1+\epsilon)Q(1+\epsilon')W \leq (1+\epsilon)Q' \tag{4}$$

The left side of the inequality is obvious. We will show the right side. That is we have to show that,

$$\epsilon' * Q'_F + Q * W(\epsilon + \epsilon' + \epsilon\epsilon') \leq \epsilon * Q' \tag{5}$$

Simplifying we have,

$$\epsilon' \leq \frac{\epsilon * Q' - \epsilon * Q * W}{Q'_F + (1+\epsilon)WQ} = \frac{\epsilon Q'_F}{Q'_F + (1+\epsilon)WQ} \tag{6}$$

Now, by our condition in step 1, whp we have $Q'_F/Q \geq \epsilon/2$. Using the fact that $W = 1 - Q_F/Q$, we can show that the above inequality holds.

Now we can show that this gives an incremental ϵ approximation for determining QoS guarantee. This is clear if we proceed to step 2, since we compute Q' with ϵ accuracy. On the other hand, if we finish at step 1, then because of the condition stated, $Q'_F \leq \frac{\epsilon Q}{2}$ whp. Since $Q' \leq Q + Q'_F$, we still don't violate the QoS guarantee whp.

Finally, we show that running time of the algorithm is $O(n|F|\log m)$. Step 1 takes $O(n|F|\log m)$ time. Step 2 takes $O(n|F|\log m)$ time because we have to check for overflow only in edges belonging to F (in step 2.1.3). The upper bound we choose for the number of trials needed to estimate W needs some explanation. Step 2 is simply an application of the zero-one estimator theorem of [5]. To bound the number of trials needed, we calculate a lower bound on W needed for an ϵ estimation of Q'. We do this as follows. We again have eight different cases, but the inequality in 4 gives the lower bound on W. Strengthening inequality 5 we require,

$$\epsilon' * Q' + Q' * W(\epsilon + \epsilon' + \epsilon\epsilon') \leq \epsilon * Q' \tag{7}$$

because $Q'_F \leq Q'$ and $Q \leq Q'$. So we have,

$$W \leq \frac{\epsilon - \epsilon'}{2 + \epsilon} \tag{8}$$

Hence it's enough if we perform $N = \frac{2+\epsilon}{\epsilon-\epsilon'} * (4 \ln(2/\delta')/(\epsilon')^2)$ trials. □

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 3) is an addition of two terms, one of which is in the form of a product. Hence if the error in each estimate is δ, the error of the total estimate adds up to $3 * \delta$. So, to maintain an ϵ approximation whp, we run the first and all calls of the incremental algorithm with $\delta = 1/(3m^3)$. Every m^2 calls we re-evaluate Q using the static algorithm for the entire network (with $\delta = 1/(3m^3)$). Since a call to the static algorithm takes $O(mn \log m)$ time the amortized work done for every addition or deletion is still $O(n|F| \log m)$.

We use the logical variable $FLAG$ to check whether we proceeded to step 2 or not in the incremental algorithm. In other words, if $FLAG =$ "true", then we have $Q'_F/Q \leq \epsilon/2$ with high probability, and we didn't proceed to step 2. Although this means, that we are still within the "safe" range as far as QoS guarantee is concerned, we have to keep track of additions or deletions, where this occurs. Otherwise, errors can accumulate. However, note that once we proceed to step 2 of the incremental algorithm ($FLAG$ will then be "false") we will have ϵ accurate estimate of Q'. Hence till this happens, F will be *union* of edges of all new connections added/deleted since the last time $FLAG$ was "false". Again, this does not change the amortized complexity of $O(n|F| \log m)$ work done per addition or deletion.

References

[1] D. Anick, D.Mitra and M.M. Sondhi, "Stochastic Theory of a Data Handling System with Multiple Sources". *Bell Systems Technical Journal*, **61**(8), 1982, 1871-1894.

[2] 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.

[3] S. Gupta, K. Ross and M. Zarki. On Routing in ATM Networks. In [9].

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

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

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

[7] 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.

[8] J. Kleinberg, Y. Rabani and E. Tardos. Allocating Bandwidth for Bursty Connections, In *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, 1997, 664-673.

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

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

[11] L. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, **8**, 1979, 410-421.

[12] 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.