Statistical Quality of Service Assurances Using Probabilistic Envelopes

Satish Raghunath, Shivkumar Kalyanaraman

Abstract— **Recent work in the area of statistical QoS has demonstrated that a) statistical envelopes can lead to large gains in utilization, b) easily enforced deterministic envelopes can be used to obtain such statistical envelopes, c) end-to-end performance bounds can be provided with high utilization targets if rate-controlled scheduling disciplines are employed on every node.**

In this paper we examine achieveable performance bounds *without* the per-node specialized scheduling, for a feedforward network. Employing enforceable statistical envelopes to bound traffic entering a network, we examine techniques to compute the envelopes at an arbitrary node inside the network by using existing results on per-hop burstiness increase. Given a delay target the probability of delay violation is computed for an end-to-end performance bound.

We then examine ways to set delay targets at individual nodes so that a) the delay targets and violation probabilities can be derived without an exact traffic description, b)the network can provide meaningful end-to-end delay assurances, b) the provider can set a target utilization constraint and quantify the tradeoff in terms of delay violation probability. Thus given a network *without* any specialized schedulers per-hop, but strict admission control, we demonstrate that reasonable utilization levels can be attained, while providing assurances on QoS parameters. The utility of the proposal is illustrated with an example feedforward network using numerical simulations.

Index Terms—**QoS, statistical assurances, network calculus**

Methods Keywords - System design, Stochastic processes/Queueing theory

I. Introduction

Quality of Service (QoS) schemes for the Internet aim to provide bounds on performance metrics for a given network topology and input traffic profile. There have been several approaches to formulate proposals which realise the goal of predictable performance. The most popular QoS architectures proposed have been Integrated Services (Intserv [BrClSe94]) and Differentiated Services (Diffserv [Bl98]). By envisaging per-flow scheduling and admission control, Intserv proposes to achieve the aim of guaranteed QoS. The per-flow state requirement hinders the scalability of the architecture. Diffserv trades off the ability to provide to guaranteed QoS by employing aggregate scheduling and hence achieves scalability.

Although guaranteed service is desirable, scalable architectures are the ones that are deployable. In this regard, it becomes important to quantify the trade-off dictated by scalability. That is, while adopting aggregate scheduling, it is imperative that we understand the *effective service* seen by constituents of the of the aggregate. Though not guaranteed QoS, statistical QoS can be provided with a good understanding of aggregate scheduling. In this regard, a framework to analyse flows in a network of multiplexers becomes important. Cruz [Cr91a], [Cr91b] provided such a framework with a deterministic network calculus. The utility and power of network calculus has been evident in the insight it has provided on such questions as network stability [AnZh01] and utilization bounds for finite delays in FIFO networks [ChBo01], [Ji02]. These insights have led to engineering of new scheduling disciplines and QoS architectures [Cr98], [Zh91], [BoTh02] to achieve bounded delay services. These proposals have employed deterministic envelopes (e.g., leaky-bucket characterization) to bound the burstiness of input traffic. In order to achieve higher utilization, more admissible connections, and understanding of average-case behavior, statistical envelopes are a better choice [BoBuLi99].

With input being characterized statistically, the analysis becomes complex due to the correlation amongst flows that exit a multiplexer. To obtain end-to-end statistical bounds on QoS metrics, we can either avoid correlation amongst flows (e.g., using bufferless multiplexers [RiRoRa02], jitter control at each node [Fe92],[ZhFe94], [Li01] etc.) or resort to approximate analysis. Since it is hard to expect Internet-wide compliance to a particular node architecture, approximate methods can prove to

Satish Raghunath and Shivkumar Kalyanaraman are with the Department of ECSE, Rensselaer Polytechnic Institute, Troy, NY-12180, Email: raghus,shivkuma@rpi.edu

This project was supported in part by: NSF Grant ANI9819112, DARPA Contract F30602-00-2-0537, and grants from Intel Corp and Nortel Networks Corp.

be very useful. Kurose [Ku92] derived end-to-end per-session performance bounds with the assumption that inputs are stochastically bounded. Exploiting the fact that these bounding random variables have finite support, he obtains worst-case characterization of the stochastic envelope after passing through a multiplexer. The first drawback of this proposal is in the fact that we need to find a distribution that is stochastically larger than the actual traffic distribution and is as close to the actual as possible. Second, computations of delay involve finding a convolution of random variables. Last, the estimates lead to conservative admission control.

Instead of bounding the traffic by an additional random variable, Yaron and Sidi [YaSi93] proposed probabilistically bounding the traffic's burstiness using exponential bounds. Recognizing that Internet traffic displays sub-exponential nature Starobinski and Sidi [StSi00] proposed to bound the burstiness by more generic functions (Stochastically bounded burstiness). They demonstrate a network calculus using these envelopes. However, obtaining the burstiness characterization is not easy. Also, when the framework is applied to a network of nodes, the analysis becomes very complex and does not easily allow computing delay bounds. Moreover, these envelopes are not easy to enforce. By describing a statistical service curve [LiPaBu01] obtains an end-to-end effective service curve. Although [LiPaBu01] provides means to achieve probabilistic delay and backlog guarantees, the input traffic is still bounded by deterministic envelopes.

Knightly [Kn97a] proposed rate-variance envelopes derived from the underlying deterministic envelopes to characterize traffic. Utilizing the concept of *Rate Controlled Scheduling Disciplines* (RCSD) and using a Central Limit theorem approximation, [Kn98] demonstrates an end-to-end statistical QoS framework. Recently, a new statistical traffic envelope (called *effective envelope*) was proposed by Boorstyn et al [BoBuLi99]. These envelopes bound the arrivals in a given interval with high probability. Although similar in function to Kurose's bounding random variables, the envelopes are shown to be easily computed for mutually independent flows. The advantage with these envelopes is that, as in [Kn97a], these are easily calculated from enforceable deterministic envelopes. However, the envelopes are computed using an assumption of independence of underlying traffic flows.

Thus the desirable characteristics of a statistical framework for QoS would feature the following: a) *Enforceable* traffic envelopes which can be easily implemented using network elements such as shapers; b) Statistical envelopes computed using the parameters of these enforceable deterministic envelope; c) Minimal requirement from the network nodes for the performance bounds to hold true. The bounds on network performance that are thus obtained should ideally lead to high utilization and should be amenable to online computation.

In this paper, we use Knightly's rate-variance envelopes to statistically characterize traffic which is shaped by (σ,ρ) shapers. Thus these envelopes are enforceable and easily computed. However, we do not resort to RCSD-type nodes. Instead, we use the *busy period bound* as in Kurose's [Ku92] paper to obtain the changed rate-variance envelopes after the flow exits a multiplexer. So we do no require the nodes to employ any special function other than FIFO scheduling. Also, we do not need the bounding random variables as in [Ku92] and computations are simple as indicated by [Kn98]. The downside of this approach is that the estimates so obtained are conservative and lead to lower utilization.

We then propose an admission control algorithm which utilizes pre-determined per-hop delay targets and violation probabilities to ensure end-to-end performance bounds. In order to decide the delay targets at each hop, we derive bounds on the delay violation probability at a given node, without the exact traffic descriptions. To achieve this we make some simple easily realizeable assumptions. Utilizing the bounds so derived, we examine how, given a network topology, one can arrive at realizeable end-to-end delay targets. We then observe that the admission control algorithm serves as the means to realize the performance bounds.

The contributions of this paper are as follows: a) Using enforceable leaky-bucket envelopes we obtain end-to-end performance bounds; b) Requiring just FIFO scheduling from every node, we examine worst-case effects on statistical envelopes; c) Without requiring an exact description of the traffic in the network, we derive means to bound delay violation probabilities and hence arrive at a means of quantifying a network's ability in terms of the delay assurances it can provide, d) We provide a middle-ground between conservative easy-to-compute deterministic bounds, which lead to low utilization, and accurate statistical bounds requiring network-wide upgrade (in the lines of RCSD) - our approach gains from the good features of statistical envelopes and the derived bounds allow the network to be just FIFO.

The rest of the paper is organized as follows. In Section II we discuss the network model and assumptions. In Section III We then present the theoretical tools that can be employed to obtain the statistical envelope of a flow at a given node in the network. We then tackle the problem of admission control and per-node delay allocation in Section IV. The results are numerically evaluated using a feedforward network in Section V. The paper is summarised and directions for future work examined in Section VI.

II. Network Model, Assumptions and Notation

In the following sections, the network is assumed to be of the feedforward-type, i.e., there are no loops in a flow's path. The term "flow" is used to mean a traffic aggregate that is policed by a leaky bucket shaper before it enters the network. $A_i(t, t + \tau)$ represents the total number of arrivals due to flow i in the interval $(t, t + \tau]$. The parameters of the leaky bucket shaper for flow i are denoted as (ρ_i,σ_i), where ρ_i and σ_i denote the rate and bucket depth respectively. The capacity of a link i is denoted as C_i. Every flow is assumed to be deterministically bounded by an envelope $A_i^*(\tau)$ so that $A_i(t, t + \tau) \leq A_i^*(\tau)$. Additionally the flows $A_i(t, t+\tau)$ are assumed to be stationary random variables, $P[A_i(t, t + \tau) \leq x] = P[A_i(t', t' + \tau) \leq x] \, \forall t \geq 0, \forall \tau \geq 0$.

Each node in the network is assumed to have multiple incoming and outgoing links. In the rest of the paper, a multiplexer in a node is used to indicate the output queue corresponding to an outgoing link. Thus the packets entering a node are switched to the right output queue depending on the destination. A multiplexer is identified by a pair (i, j) where i denotes the node id and j denotes the index of the multiplexer within the node. A path is assumed to be specified as a set of multiplexers. The effects of packetization are not taken into account and the multiplexers are treated as if their service is infinitely divisible.

III. Network Analysis

In order to obtain delay bounds for a network of nodes, it is necessary to know the statistical characteristics of the flow at each incident node. Multiplexing of flows causes distortion of the original traffic characteristics (as it was when it entered the network) of the flows. Notably, a flow's burstiness increases as it passes more hops. This is due to the fact that a flow's packets get bunched together while the link is serving other flows. An increase in burstiness means that the maximum possible busy period at queues inside the network increases. Thus any end-to-end QoS proposition has to choose a way to deal with this problem. One of the most elegant solutions has been suggested by Zhang and Ferrari in [ZhFe94]. They propose a mechanism wherein the flow's characteristics remain unchanged as it passes through a multiplexer. A rate-controller (or a jitter-controller) is proposed per-connection, which ensures that the inter-packet time remains the same as the flow exits the scheduler. While this solution takes care of the problem of increase in burstiness, it requires that every node in the path be ugraded with this capability. To allow for an incremental deployment it is useful to understand what QoS we can provide by better understanding the distortion effects of the network.

With no specialized schedulers in the network, it is necessary to quantify the effect of increase in burstiness. Traffic that is deterministically constrained can be analysed with a rich set of tools available as part of deterministic network calculus [BoTh02]. Leaky-bucket shapers specified as (ρ, σ) are commonly used. It has been pointed out that envelopes like D-BIND [Kn97b] and shaper banks of the form $\min_k\{\sigma_k + \rho_k t\}$ capture multiple timescales of source traffic statistics and hence are better. In this paper we utilise leaky-bucket shapers only, since the focus here is to demonstrate that reasonable network utilization is achieveable without specialized schedulers in the network. It is possible to achieve the same goals with the other deterministic envelopes and obtain better results.

Kurose [Ku92] proposed a strategy to *push* stochastic envelopes through multiplexers. Using a bound for the busy period at each node, Kurose derives the statistical envelope for the flow as it exits the multiplexer. We shall employ the same strategy. Instead of utilizing bounding random variables as in [Ku92] we shall employ statistical envelopes that are derived from deterministic envelopes. In order to obtain the evelope of a flow at each node, we obtain the deterministic envelope of the flow at the node (Section III-C). We then deduce the statistical envelope from the deterministic envelope. As mentioned before, due to burstiness increase in the network, the violation probabilities increase for the same delay target. Setting the delay targets properly becomes important. We elaborate more on how to do this independent of traffic descriptions in Section IV and provide strategies to set per-node delay targets. Thus at some node in the network we have a statistical envelope of the incident traffic, and a predetermined delay target. This enables us to compute the delay violation probabilities at each node. These probabilities are then used to compute an end-to-end probability of delay violation.

A. Enforceable Statistical Envelopes

Recently [Kn97a] and [BoBuLi99] demonstrated that statistical envelopes can be derived from deterministic envelopes. The variance of a flow can be bounded if its deterministic envelope is known. Denoting $A(t, t+\tau)$ as the number of arrivals in an interval τ and noting that the flow is deterministically bounded by $A^*(\tau)$, it has been shown that:

$$\begin{aligned} Var[A(t,t+\tau)] &\leq \rho\tau(A^*(\tau)-\rho\tau) && (1)\\ &= \rho\sigma\tau \ for\ a\ (\sigma,\rho)\ flow && (2)\end{aligned}$$

$Var[A(t,t+\tau)]/\tau^2$ is also referred to as the *Adversarial Rate-Variance* envelope of the flow in [Kn98]. Due to the independence assumption on the flows, the aggregate process $S(t,t+\tau) = \sum_j A_j(t,t+\tau)$ can be characterized as:

$$\begin{aligned} Var[S(t,t+\tau)] &= \sum_j Var[A_j(t,t+\tau)] && (3)\\ &\leq \sum_j \rho_j\tau(A_j^*(\tau)-\rho_j\tau) && (4)\end{aligned}$$

Clearly, if the deterministic envelope is specified as a leaky-bucket envelope its long term average is given by the rate parameter of the leaky-bucket. Thus the aggregate process has a mean given by the sum of the means of the individual flows, which is known. Utilizing a Gaussian assumption, the complementary distribution of the aggregate process can be calculated as elaborated further below.

B. Maximum Busy period

Recall that to obtain the statistical envelope of a flow inside the network we need to characterize the burstiness increase at each node. Chang [Ch94] and Kurose [Ku92] noted that if the worst-case increase in burstiness is accounted for at each node, the new envelopes adjusted for the increased burstiness can again be treated as independent. Thus we are interested in quantifying the worst-case burstiness in terms of the deterministic parameters of the flow. The maximum busy period at a multiplexer can be employed to calculate the worst-case increase in burstiness for a flow passing through it. A simple means of arriving at the busy period at a node is to use deterministic envelopes. As defined by Chang [Ch94], the busy period bound can be stated as:

$$\beta = \inf\{t > 0 : A^*(t) - Ct \leq 0\} \quad (5)$$

Without considering the scheduling discipline, we can write the worst case busy period bound in terms of the leaky bucket parameters (ρ_j, σ_j):

$$\beta = \frac{\sum_j \sigma}{C - \sum_j \rho_j} \quad (6)$$

With the additional assumption that the scheduling discipline is FIFO, we can write ([BoTh02], theorem 6.2.3)

$$\beta_F = \frac{\sum_j \sigma}{C} \quad (7)$$

Thus a flow's deterministic characterization when it leaves a multiplexer is given as:

$$A^*(t) = \rho t + \sigma + \rho\beta_F \quad (8)$$

We shall use this fact to derive the deterministic envelope of a flow at an arbitrary node in the network.

C. Calculating Deterministic envelopes

In order to obtain the deterministic envelope at a given node, it is necessary to calculate the increase in burstiness due to multiplexing at hops preceding the current node. A simple way to do that is to account for all the flows that this flow encountered in the previous hops. To quantify this idea, we define an *Incident* vector, $\mathbf{I_n}$, denoting the flows incident at multiplexer n. Every multiplexer is associated with a transformation matrix, $\mathbf{T_n}$. The product $\mathbf{I_n^t T_n}$ gives the vector with the increased burstiness values for each flow.

Definition III.1: $\mathbf{I_n}$, associated with multiplexer n, is a $M \times 1$ vector with the j^{th} element $I_n^j = \sigma_j$

Definition III.2: $\mathbf{T_n}$, associated with multiplexer n, is a $M \times M$ matrix with the element (i,j) defined as:

$$T_n^{(i,j)} = \begin{cases} 1 + \frac{\rho_j}{C} & if\ i = j\\ \frac{\rho_j}{C} & if\ i \neq j\end{cases}$$

Clearly, the element j in the product $\mathbf{I_n^t T_n}$ would be of the form $\sigma_j + \frac{\rho_j \sum_j \sigma_j}{C}$. This is the form of the burstiness factor as seen in Equation (8).

Noting that each node features multiple incoming and outgoing links, we introduce an additional dimension to identify a multiplexer within a node. Thus, within the n^{th} node, the k^{th} multiplexer is associated with $\mathbf{T_{n,k}}$ and $\mathbf{I_{n,k}}$. In order to compute the deterministic envelopes incident at a given queue, the following procedure would then be followed. Consider a path consisting of multiplexers $[\mathbf{T_1}, \ldots, \mathbf{T_k}]$. After passing through these multplexers, the vector I_l would be transformed as $I_l\mathbf{T_1}\ldots\mathbf{T_k}$. Thus the incident vectors at any queue can be iteratively calculated given the initial vector.

D. Delay at a queue

Given the deterministic envelope, we can arrive at a bound on the variance of the flow. Note that the deterministic envelope has been obtained by allowing for the worst-case increase in burstiness. Thus the envelopes can be treated as being representative of independent flows. As in Section III-A we can characterize the aggregate process $S(t)$ as a Normal random variable with variance being the sum of the variances of the individual flows. The mean is given in terms of the sum of the leaky-bucket rates of the individual flows. Thus the mean and variance turn out to be $(t\sum_j \rho_j, t\sum_j \rho_j\sigma_j)$. As shown in [Kn98], we can write,

$$Pr\{D > d\} \approx \max_{0\le t\le\beta} P(S(t) > C(t+d)) \quad (9)$$

$$\beta = \frac{\sum_j I_n^j}{C} \quad (10)$$

Given that the probability of delay violation at a multiplexer on a path is d_i, we can write the end-to-end probability of delay violation for the path as $1-\prod_i(1-d_i)$. This is due to the fact that these probabilities have been calculated after taking into account the worst-case possibilities at each node.

IV. Admission Control Algorithm

In this section we utilize the ability to compute the delay violation probability at an arbitrary node to define an admission control algorithm.

A. Algorithm

Given the incident vector at a multiplexer, we saw how delay violation probabilities can be calculated. It is then clear that an admission control regime can be built if these incident vectors are maintained to indicate the current committments. Specifically, assume that there is a matrix of incident vectors. Each element in the matrix gives the vector indicating the flows that are incident at that multiplexer. Now, consider the problem of admitting a flow that traverses a fixed path between ingress i and egress j. Clearly, this path can be specified as a vector of multiplexers, each multiplexer being identified by a node number and an index within the node. Hence a path could be $\{(1,1),(2,1),(5,2),(7,3)\}$ when a flow is traversing the path between nodes 1 and 7. The problem of admission control then reduces to checking if adding this flow to the incident vectors at these multiplexers violates a delay committment.

Algorithm 1 specifies such a procedure. If the delay violation probabilities at a node exceed a pre-determined quantity ϵ, the flow is rejected. For the algorithm to work, the delay to be assured and the violation probabilities must be known. We shall examine this issue in the next subsection and again in numerical simulations in Section V. The admission control algorithm mentions assumptions on the busy period bound and allowable leaky-bucket parameters. These are obtained in the next section.

Algorithm 1 Admission Control Algorithm

$(\rho_{max}, \sigma_{max})$ are the maximum allowable parameters for a flow
$\beta_{max}^{(i)}$ is maximum busy period allowable at ingress i
(ρ, σ) are the leaky-bucket parameters of the flow being admitted
$I_i^{(j)}$ is the incident vector at the $j^(th)$ multiplexer of i^{the} node
P_{ij} is the path vector with each element (k, l) denoting a multiplexer from Ingress i to Egress j
d_j is delay assured at node j with violation probability less than ϵ
$n = 1$
if $(\rho,\sigma) > (\rho_{max},\sigma_{max})$ **then**
  Reject flow;
end if
if $\frac{\sigma_j+\sum_k \sigma_k}{C} > \beta_{max}^i$ for each flow j through ingress i with capacity C **then**
  Reject flow;
end if
while $n < num\ elements\ in\ P_{ij}$ **do**
  $(k,l) = P_{ij}(n)$
  $I_k^{(l)} \leftarrow l^{th}\ multiplexer\ at\ node\ k$
  Add (ρ,σ) to $I_k^{(l)}$
  if $P\{D > d_j\} > \epsilon$ **then**
    Remove (ρ,σ) from $I_k^{(l)}$
    Reject flow
  else
    continue
  end if
end while

B. Delay Allocation

If the network does not provide per-flow scheduling, delay experienced is dependent on the character and number of other participating flows in the system. In such a situation, if a lower delay is set as the target (at some violation probability), the utilization in the network might be very low. On the other hand, if the delay target is set higher, the increase in burstiness due to multiplexing of flows is higher. Thus there is a tradeoff between assuring a particular delay (at a certain probability of violation) and the network utilization.

One strategy could be to set a constant delay as the target at every node in the path. In such a case, the probability of delay violation increases as the flow progresses along the path. This is due to the increase in burstiness due to multiplexing. Thus a node further downstream can allow much lesser utilization for a fixed delay and violation probabilities. If the utilization target is known, the vi-

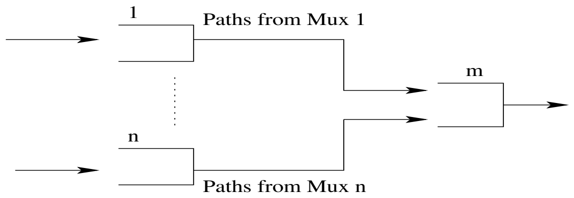

Fig. 1. Traffic feeding into a multiplexer from different paths

olation probability should be such that a constrained link towards the end of the path can provide that utilization. Without an accurate characterization of the traffic matrix, it is hard to compute the delay and the violation probabilities. Thus setting same constant delay as a target at each node is fraught with issues.

An alternate strategy would be to allocate higher delays as targets to nodes further downstream. One way to do this is to observe that the busy period at a node is dependent on the burstiness increase in the flows due to upstream multiplexers. Suppose that a maximum busy period and utilization is set at the ingress nodes. Then a bound on the achieveable delay targets for the downstream nodes can be derived. Thus we make the following assumptions:

1) At the entry of the network (the ingress), flows are admitted such that the maximum busy period at the ingress multiplexers is bounded by a deterministic value β_{max}. This can be ensured by checking at any time that $\sum_j \sigma_j \leq C\beta_{max}$.
2) At the ingress the maximum rate and burst parameter for any flow's leaky bucket specification is bounded by $(\rho_{max}, \sigma_{max})$.
3) The network has pre-determined "virtual" paths between each ingress and egress. Each path i is considered to have a path capacity of PC_i. No reservations need be made inside the network. There is no isolation of traffic assumed between paths. Thus these paths are "virtual".
4) Each path P_i has a utilization target u_i.

With these assumptions we are ready to examine the problem of setting per-node delay bounds independent of instantaneous traffic descriptions. We show that given the busy period bound at the ingress, the busy period at every succeeding multiplexer in the path is bounded due to the bounded utilization target. We then show that the calculation of the target delays can be done without knowledge of number of flows or their description, given a particular violation probability. For the following paragraphs, consider multiplexers 1 to n feeding into m. Let their capacities be C_i, and their busy period bounds be β_i.

Proposition IV.1: Consider a flow j with leaky-bucket parameters (ρ_j, σ_j) at the ingress node. Its worst-case burstiness increase is bounded by $\rho_{max}\beta_{max}$ and its deterministic envelope after passing the multiplexer is bounded by $\rho_{max} + \sigma_{max} + \rho_{max}\beta_{max}$.

Proposition IV.2: Consider a multiplexer fed by only ingress nodes. Let its busy period bound be $\beta^{(1)}_{max}$ and let the busy period bounds and capacities of the ingress multiplexers be β_i and C_i respectively. Then its busy period is bounded by $\frac{\sum_i \beta_i C_i}{C}$.

Proof:

$$\beta^{(1)} = \frac{\sum_{Flows\,i} \sigma_i}{C} \tag{11}$$

$$= \sum_{\{Muxes\,j\}} \sum_{\{Flows\,i\ from\ mux\,j\}} \frac{\sigma_i}{C} \tag{12}$$

$$\leq \frac{\sum_{Muxes\,j} C_i\beta_i}{C} \tag{13}$$

■

Now consider a multiplexer that is further downstream. The maximum burstiness of a flow at that node can be bounded by burstiness parameters of the upstream multiplexers as demonstrated by Proposition IV.1. Similar iteration can yield the busy period bound in terms of the busy period bound at the ingress nodes. Consider the aggregate process $S(t)$ at the multiplexer. As discussed in Section III, the probability of delay violation can be written in terms of an approximate Gaussian cumulative probability distribution with mean $\sum_i \rho_i\tau$ and variance bounded by $\sum_i \sigma_j\rho_j\tau$. We then have the following proposition.

Proposition IV.3: Consider a multiplexer m with its busy period bound being β_m and the envelopes of the flows being bounded by (ρ_m, σ_m). Let P_i denote the $i^(th)$ path passing through the multiplexer. By the assumptions previously mentioned their path capacities are known to PC_i and utilization target bounded by u_i. Let the mean and variance of the aggregate process be $m(\tau)$ and $v(\tau)$ respectively. Then we have:

$$m(\tau) \leq \sum_i u_i PC_i \tau = M(\tau) \tag{14}$$

$$v(\tau) \leq \sigma_m\tau(\sum_i u_i PC_i) = V(\tau) \tag{15}$$

Proof:

$$m(\tau) = \tau\sum_j \rho_j \tag{16}$$

$$= \tau \sum_{Paths\,i} \sum_{Flow\,j\in P_i} \rho_j \tag{17}$$

$$\leq \tau \sum_{Paths\,i} u_i PC_i \tag{18}$$

$$v(\tau) = \tau\sum_j \sigma_j\rho_j \tag{19}$$

$$\leq \quad \tau\sigma_m \sum_j \rho_j \tag{20}$$

$$\leq \quad \tau\sigma_m (\sum_i u_i PC_i) \tag{21}$$

■

We now have the mean and variance bounded for the aggregate process at a given multiplexer. We now have the following proposition.

Proposition IV.4: Let the flows incident at a multiplexer m with busy period bound β_m, be characterized by leaky bucket parameters (ρ_j, σ_j). If the probability of delay violation for a target delay d is approximated as:

$$d_v = P\{D > d\} \tag{22}$$

$$\approx \max_{0 \leq t \leq \beta_m} \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(C(t+d) - t\sum_j \rho_j)^2}{2t\sum_i \rho_i \sigma_i}\right) \tag{23}$$

then we have, $d_v \leq d'_v$ where

$$d'_v = \max_{0 \leq t \leq \beta_m} \frac{1}{\sqrt{2\pi}} exp\left(-\frac{(C(t+d) - M(t))^2}{2M(t)}\right)$$

with M(t) and V(t) given as in Proposition IV.3
Proof: Using Proposition IV.3 to compare the fractions in the exponent parts in the two expressions, the result follows. ■

The result in Proposition 22 indicates that without relying on the exact traffic descriptions we can calculate the probability of delay violation at any given multiplexer in a network. Note that to obtain the probability for a given delay target, we need the network topology. We already know that the delay violation probability for a path is obtainable in the product form. Thus given an end-to-end delay violation probability requirement, one could start at the end of the path and calculate the permissible delay target using Proposition 22. Then moving on to upstream multiplexers on the path, one can reduce the violation probability and get better delay targets. Once the delay targets are calculated for for every node, the end-to-end delay assurance can then be obtained by summing the individual targets.

Thus without being dependent on the exact traffic descriptions we can set delay targets for each multiplexer in the network so that a particular violation probability is achieved for each path. The admission control algorithm would then also have to enforce the additional assumptions mentioned earlier in the section.

C. Scalability

For the admission control algorithm to work, we need to maintain a central entity that possesses information about all aggregates admitted into the network. Note that this algorithm is intended for a network service provider. Hence the flows in question are actually aggregates from customers. Although, the number of aggregates will be high, it is expected not to be in the scale of the number of end-to-end sessions. Given that fact, the maximum size of the incident vector at a multiplexer will be bounded by (Number of Paths through this multiplexer)*(Number of Aggregates on the path). Each multiplexer is associated with one incident vector. Hence the matrix of incident vectors will have about (Number of Nodes)*(Number of Multiplexers per node).

If we consider the core network of a service provider, the number of nodes we are dealing with might not be too large. In cases where number of nodes is in fact very large, the network might be split into multiple managed domains of smaller sizes. Then there would have to be a co-ordination among the admission control entities of these sub-domains.

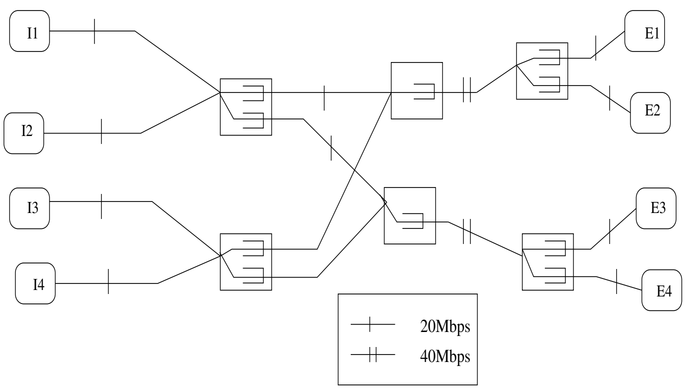

Fig. 2. Network with cross traffic and multiple hops of multiplexing

V. Example analysis for a feedforward network

In this section we illustrate the use of the admission control scheme developed in the previous sections. Using a feedforward network as shown in Figure 2 we evaluate the scheme. We are interested in examining: a) the means to set the parameters of target utilization and allowable burstiness, b) the end-to-end delay targets that can be met for the sample scenario in comparison with a network-wide upgrade, c) applying these techniques to a Diffserv network in order to obtain QoS assurances.

A. Network Topology

The network has four "ingresses" and four "egresses". For the numerical simulations, two paths for each ingress are considered, bringing the total number of paths to 8.

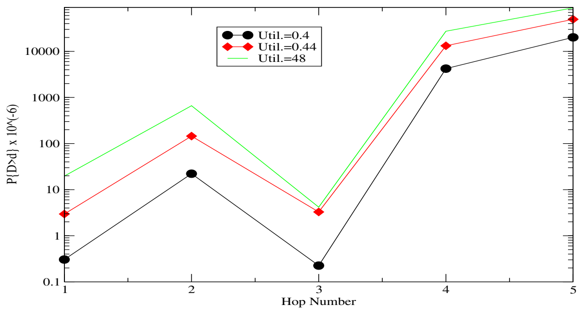

Fig. 3. Variation of probability of delay violation with each hop with constant fixed delay target

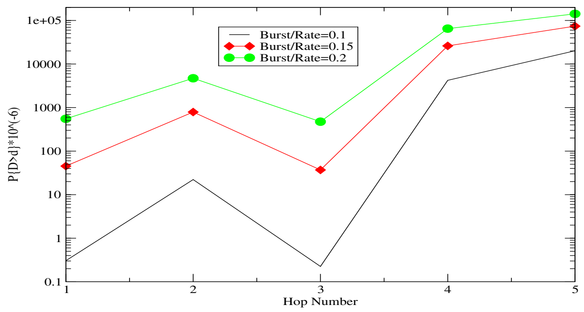

Fig. 4. Variation of probability of delay violation with different burstines

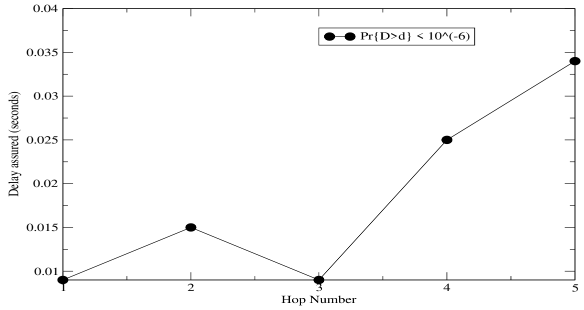

Fig. 5. Variation of probability of delay violation with each hop with different delay targets per-hop

E.g., flows from I1 to E1 and E3 are considered. The bottleneck capacities (double vertical lines) are chosen to be 40Mbps while the other links are chosen to be 20Mbps. Each flow was chosen to be characterized by a leaky-bucket specification of (400Kbps, 40Kbits) to represent $(\rho_{max}, \sigma_{max})$. The path chosen to be analysed was the one from I1 to E1. Note that structure of the network is such that all the 8 paths in the simulation have identical characteristics.

B. Utilization target

A higher utilization target clearly leads to higher delays and delay violation probabilities. The variables to be considered include the delay violation probability and end-to-end delay. If a constant per-hop delay target is set we see in Figure (3) that the violation probability increases with each hop. The order of magnitude of the end-to-end delay violation probability is decided by the node with the highest probability of violation. Thus if we vary utilization and observe the effect on the violation probability at the final hop we will have a fair idea of the end-to-end probability of delay violation. Thus given a maximum leaky-bucket parameter pair, we could set the target utilization for a particular end-to-end delay violation probability.

For example, if we set a utilization target of 0.4 for a maximum leaky-bucket parameter set of (400kbps, 40k) we obtain a delay violation probability of approximately 10^{-2} for the constant per-hop delay of 0.01.

C. Effect of maximum burstiness

As noted in the previous subsection the values for utilization and maximum leaky-bucket parameters are closely tied together. Two of these parameters have to be set beforehand and the effect of varying others has to be examined. Here we fix the utilization target at 0.4 and maximum rate at 400kbps and vary the burstiness. We then obtain a set of indicative numbers about what violation probabilities can be obtained for given burstiness. Increase in burstiness causes delay violation probabilities to be higher as seen in Figure (4).

D. End-to-end Delays

In the previous discussions we had fixed a constant per-hop delay of 0.01 to evaluate the effect of other parameters. For lower probabilities of violation we need to set the delays higher at the nodes further downstream on the path. If we keep the delay violation probability fixed at 10^{-6}, utilization at 0.4, maximum leaky-bucket parameters at $(400kbps, 40kbits)$ and examine the per-hop delays that can be assured, we observe the plot in Figure (5). The total end-to-end delay for the path in the above simulation was $0.092s$. If there were RCSD style schedulers in the whole network, the per-hop delay would have been the same as that in the first hop. The total delay in the RCSD network would be thus less than $0.045s$. Thus the gains of network-wide upgrade are phenomenal. However, with the current scheme we still can assure a delay with high confidence.

Note that the results presented here are highly dependent on the topology and can only be viewed as a verification of the utility of the proposed scheme. The actual delay assurances that can be assured are highly case-specific. The problem solved here is that deciding on delay targets and an admission control regime so that an assurance can be provided with just a FIFO feedforward network.

E. Diffserv Expedited forwarding

In the network considered above, we assumed FIFO queues. Traffic from all aggregates were multiplexed into just one single queue. The Diffserv Expedited Forwarding [Da02] specification endeavors to provide a minimum rate guarantee at each node in the network for a packet tagged to belong to the EF class. One of the suggested mechanisms is to implement a static priority queue at each node, with no class given higher priority than the EF class. Denoting the EF class with priority 1 and other traffic as priority 2, we can once again examine the delay violation probabilities. As indicated in [Kn98] the probability of delay violation is given as below:

$$P\{D_i > d_i\} \approx \max_{0 \le t \le \beta_1} \frac{1}{\sqrt{2\pi}} exp\left(\frac{(C(t+d_i) - m_i(t))^2}{2v_i(t)}\right)$$

where

$$\begin{aligned}
m_1(t) &= \sum_{\{j \in EF\ Class\}} \rho_j t && (24)\\
v_1(t) &\le \sum_{\{j \in EF\ Class\}} \rho_j \sigma_j t && (25)\\
and & && (26)\\
m_2(t) &= \sum_{\{j \in EF\ Class\}} \rho_j (t + d_1) + && (27)\\
& \quad \sum_{\{j\ not\ in\ EF\ Class\}} \rho_j t && (28)\\
v_2(t) &\le \sum_{\{j \in EF\ Class\}} \rho_j \sigma_j (t + d_1) + && (29)\\
& \quad \sum_{\{j \in EF\ Class\}} \rho_j \sigma_j t && (30)
\end{aligned}$$

The techniques in Section IV are easily extended to obtain bounds for $m_i(t)$ in the above equations.

The fraction of traffic allowed to be in the EF class has an impact on the sevice of all the flows. If the number flows in the EF class are small, the burstiness increase is small and edge-to-edge delay violation probabilities are low. But only a few aggregates can avail of the advantages of the EF class. On the other hand, if the number of flows in the EF class are high, the edge-to-edge delay violation probabilities for all flows will be high. This is because, the flows in EF class will obtain service at the cost of other flows and since there are a lot EF flows being multiplexed their burstiness increase is also more. Using the techniques derived in this paper, one could easily vary the fraction of EF traffic and observe the effect on the delay violation probabilities for both classes.

F. Incremental Deployment

In order to provide QoS assurances in the Internet, the core of the network requires mechanisms to counter traffic distortions. Rate-Controlled service disciplines [ZhFe94] have been proposed to reconstruct the characteristics of the flow at each hop. By avoiding increase of burstiness at each hop, RCSD is able to provide a delay bound dependent only on the per-hop scheduler capabilities and the delays in the intervening links. However, without a network-wide upgrade to such a discipline there is little gain from the framework. If such an option is prohibitive, the ability to characterize network's performance metrics will prove invaluable.

Given a central entity in a provider network that has knowledge of the aggregates that are admitted into the core, there can be regulators inserted at specific points in the network to reshape the traffic. The holding times at these regulators could add to the delay. Still, the capability of the network as a whole might dramatically improve.

VI. Conclusions and Future Work

In this paper we examined techniques to obtain end-to-end quality of service assurances without specialized per-hop schedulers. We presented a strategy wherein enforceable statistical envelopes are used to achieve admission control. To deal with the effect of traffic distortion inside the network we characterized the increase in burstiness as flows progress along the hops. We demonstrated a method to provide assurances on delay violation probability for a given network topology without exact traffic descriptions. The techniques introduced need parameters in terms of maximum allowable leaky-bucket parameters for admitted flows, target utilization constraint and a delay violation probability. We examined the applicability of the technique for a sample feedforward network and discussed possible extensions to be applied to the Diffserv architecture.

We have utilized adversarial rate variance envelopes in the analysis. While easily enforced, they are conservative as indicated in previous studies. Also the leaky-bucket characterization cannot capture burstiness information on multiple timescales and hence is restrictive. The parameters required for the technique to work include an upper bound on the leaky-bucket parameters of the flow. If there is a lot of variation in the nature of flows admitted into the system, we once again are faced with poor utilization. Future work would concentrate on handling these problems and making parameter setting easier. The results need to be further examined in more realistic network situations.

REFERENCES

[Bl98] S. Blake, et al, “An Architecture for Differentiated Services,” RFC 2745, December 98

[BoTh02] Jean-Yves Les Boudec, Patric Thiran, “Network Calculus”, Parts I, II, III, Online version of the book, Springer Verlag LNCS 2050, Version Jan 16, 2002

[BrClSe94] R. Braden, D. Clark and S. Shenker, “Integrated Services in the Internet Architecture: an overview,” IETF RFC 1633, July 1994

[BoBuLi99] R. Boorstyn, A. Burchard, J. Liebeherr, C. Oottamakorn, “Statistical Multiplexing Gains of Link Scheduling Algorithms in QoS Networks,” Tech. Report, University of Virginia, CS-99-21

[Ch94] C.S. Chang, “Stability, Queue Length, and Delay of Deterministic Stochastic Queueing Networks,” IEEE Trans. on Aut. Control, Vol. 39, No. 5, May 1994

[ChBo01] A. Charny, J.-Y. Le Boudec, “Delay Bounds in a Network with Aggregate Scheduling,” Proc. QoFIS, Berlin, Germany, October 2000

[Cr98] R.L. Cruz, “Sced+: Efficient management of quality of service guarantees,” IEEE Infocom 98

[Cr91a] R.L. Cruz, “A calculus for network delay, part I: Network elements in isolation,” IEEE Transactions on Information Theory, Vol.37., No. 1, January 1991, pp.114-131

[Cr91b] R.L. Cruz, “A calculus for network delay, part II: Network Analysis” IEEE Transactions on Information Theory, Vol.37., No. 1, January 1991

[Da02] B. Davie, et al, “An Expedited Forwarding PHB (Per-Hop Behavior),” RFC 3246, March 2002.

[Fe92] D. Ferrari, “Design and Application of a Delay Jitter Control Scheme for Packet-switching Internetworks,” Computer Communications, 15(6):367-373, July 1992

[Ji02] Y. Jiang, “Delay Bounds for a Network of Guaranteed Rate Servers with FIFO Aggregation,” IEEE ICC 2002

[Kn97a] E.W. Knightly, “Second Moment Resource Allocation in Multi-Service Networks,” SIGMETRICS 97

[Kn97b] E.W. Knightly, “D-BIND, An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic,” IEEE/ACM Trans. on Networking Vol 5, No 2, April 1997

[Kn98] E.W. Knightly, “Enforceable Quality of Service Guarantees for Bursty Traffic Streams,” INFOCOM 98

[Ku92] J. Kurose, “On Computing Per-session Performance Bounds in High-Speed Multi-hop Computer Networks,” SIGMETRICS 92

[LiPaBu01] J. Liebeherr, S. Patek, A. Burchard, “A Calculus for End-to-end Statistical Service Guarantees,” University of Virginia Tech Report CS-2001-19 (revised).

[MaSi97] G. Mayor, J. Silvester, “Time Scale Analysis of an ATM Queueing System with Long-Range Dependent Traffic,” INFOCOM 97

[RiRoRa02] M. Riesslein, K.W. Ross and S. Rajagopal, “A Framework for Guaranteeing Statistical QoS,” IEEE/ACM Transactions on Networking, Vol 10, Issue 1, Feb2002

[StSi00] D. Starobinski, M. Sidi, “Stochastically bounded burstiness for communication networks,”IEEE Transactions on Information Theory, Vol.46, No. 1, Jan. 2000, pp. 206-212

[YaSi93] O. Yaron, M. Sidi, “Performance and Stability of Communication Networks via Robust Exponential Bounds,” IEEE/ACM Transactions on Networking, 1993

[ZhFe94] “Rate Controlled Service Disciplines,” Journal of High Speed Networks, 1994

[ZhTo95] Z. Zhang, D. Towsley, J.F. Kurose, “Statistical Network Performance Guarantees with Generalized Processor Sharing Scheduling,” JSAC 95

[AnZh01] M. Andrews, L. Zhang“Stability of FIFO networks”

[Zh91] Z. Zhang, “Fundamental Trade-offs in scheduling,”ICNP 2001

[Li01] J. Liebeherr, “End-to-end quality of service guarantees,” IwQoS 2000