On the QoS Provisioning Power of Optimal Aggregate-Flow Scheduling*

Huan Ren   Kihong Park†
Network Systems Lab
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{renh,park}@cs.purdue.edu

CSD-TR 00-021
November 3, 2000

Abstract

Providing scalable QoS-sensitive services to applications with varying degrees of elasticity using aggregate-flow scheduling is a challenging problem. In [31], the authors advanced a theoretical framework where an optimal differentiated services provisioning problem is formulated and solved to yield solutions for optimal per-hop behavior (PHB) and absolute QoS shaping using end-to-end control. In this paper, we build on their work and achieve the following innovations. First, we generalize the mean square error criterion employed in [31] for measuring goodness of aggregate-flow scheduling by introducing a scaling function which is applied to the TOS field label value in the IP header at each router before entering its classifier. This allows the service provider to configure the PHB so as to export customized QoS separation—essential when shaping end-to-end absolute QoS over per-hop relative QoS—commensurate with the QoS profiles of its user base. Second, we show that end-to-end QoS can be efficiently provisioned using per-hop aggregate-flow scheduling across a multi-hop path. We show that adaptive label control—executed end-to-end—assigns local QoS responsibility to routers inversely proportional to their load, and this QoS imbalance holds both statically and dynamically. Third, we show that end-to-end QoS shaping power degrades gracefully as the number of service classes at routers in a multi-hop path is decreased, which admits heterogeneity in router configurations and incremental deployment. Collectively, our results demonstrate that relative QoS can be effectively harnessed to achieve absolute QoS in WAN environments if the PHB implements optimal aggregate-flow scheduling in the generalized framework.

*This work was supported in part by NSF grants ANI-9875789 (CAREER) and EIA-9972883.

†K. P. Additionally supported by NSF grants ANI-9714707, ESS-9806741, ANI-0082861 (ITR), and grants from the Purdue Research Foundation, Santa Fe Institute, Sprint, and Xerox.

1 Introduction

Architecting networks capable of providing scalable, efficient, and fair services to users with diverse QoS requirements is a challenging problem. The traditional approach uses resource reservation and admission control to provide both *guarantees* and *graded* services to application traffic flows. Analytical tools for computing and provisioning QoS guarantees [7, 8, 26] rely on overprovisioning coupled with traffic shaping/policing to preserve well-behavedness properties across switches that implement a form of GPS packet scheduling. Scale-invariant burstiness associated with self-similar network traffic [18, 29] limits the shapability of input traffic while at the same time reserving bandwidth that is significantly smaller than the peak transmission rate. This imposes a trade-off between QoS and resource utilization which limits the degree of utilization achievable while guaranteeing stringent QoS. For applications needing guaranteed services, the unconditional protection afforded by per-flow resource reservation and admission control is a necessity. However, for elastic applications that require QoS-sensitive services but not guarantees it would be overkill to provision QoS using the mechanisms of per-flow reservation and admission control. In addition to the service mismatch, overhead associated with administering resource reservation and admission control impedes scalability.

Recently, efforts have been directed at designing network architectures with the aim of delivering QoS-sensitive services by introducing weaker forms of protection or assurance to achieve scalability [2, 5, 9, 20, 22]. The differentiated services framework [1, 5, 14, 22] has advanced a set of building blocks comprised of per-hop and access point behaviors with the aim of facilitating scalable services through aggregate-flow resource control inside the network and per-flow traffic control at the edge. By performing a many-to-one mapping from the large space of individual flows to the much smaller space of aggregate flow labels, scalability of per-hop control is achieved at the expense of introducing uncertainty and volatility by flow-aggregation and aggregate-flow packet scheduling per-hop. In [31], the authors introduced a theoretical framework where an optimal differentiated services provisioning problem is formulated and solved to yield an optimal aggregate-flow scheduler which forms the per-hop behavior. The authors also study stability and efficiency of the overall system when driven by selfish users and service providers in the game theoretic context.

In this paper, we build on their work and achieve the following innovations. First, we generalize the mean square error criterion employed in [31] for measuring goodness of aggregate-flow scheduling by introducing a scaling function which is applied to the TOS field label value in the IP header at each router before entering its classifier. This allows the service provider to configure the PHB so as to export customized QoS separation—essential when shaping end-to-end absolute QoS over per-hop relative QoS—commensurate with the QoS profiles of its user base. We show that this enhances system efficiency—less bandwidth is needed to achieve the same QoS requirement—in a nontrivial way. Second, we show that end-to-end QoS can be efficiently provisioned using per-hop aggregate-flow scheduling across a multi-hop path. We show that adaptive label control, which is executed end-to-end, assigns routers local QoS responsibilities inversely proportional to their load, and this QoS imbalance holds both statically (steady-state) and dynamically (magnitude of change during transient adaptation). Third, we show that end-to-end QoS shaping power degrades gracefully as the number of service classes at routers in a multi-hop path is decreased. This admits heterogeneity in router configurations which, in turn, enables incremental deployment. Collectively, our results support the view that relative QoS can be effectively harnessed to achieve absolute QoS in WAN environments if the PHB implements optimal aggregate-flow scheduling in the generalized optimal differentiated services framework.

The rest of the paper is organized as follows. In Section 2, we give an overview of related work. Section 3 defines the network architecture and QoS provisioning set-up. In Section 4 we define the generalized optimal aggregate-flow scheduling framework and derive its solution. Section 5 presents simulation-based performance evaluation of generalized optimal aggregate-flow scheduling with respect to absolute QoS shaping, efficiency, and structural nonuniformity. We conclude with a discussion of our results and future work.

2 Related Work

The traditional approach to QoS uses resource reservation and admission control to provision guaranteed services—deterministic or statistical—with the help of leaky bucket regulators. Although research abounds [6, 7, 10, 13, 17, 25, 26], analytic tools for computing QoS guarantees rely on shaping of input traffic to preserve well-behavedness across switches which implement a form of generalized processor sharing (GPS). Real-time constraints of multimedia traffic and the scale-invariant burstiness associated with self-similar network traffic [18, 27, 30, 34] limit the shapability of input traffic while at the same time reserving bandwidth that is significantly smaller than the peak transmission rate. Statistical multiplexing—when employing a small buffer capacity/large bandwidth resource provisioning policy [29]—can, by an application of the central limit theorem, yield improved efficiency while achieving predictable performance. However, the statistical guarantee is only approximate, some efficiency is lost when fitting leaky bucket parameters to self-similar input [33], and second-order properties—even under bufferless queueing—adversely affect jitter and related second-order performance measures. Thus QoS and utilization stand in a trade-off relationship with each other and transporting application traffic over reserved channels, in general, incurs a high cost.

Aggregate-flow scheduling has been investigated in the noncooperative QoS provisioning context—explicitly for multi-class QoS and implicitly for competitive routing. In [2, 3, 28] the authors introduced aggregate-flow per-hop packet scheduling mechanisms and end-to-end controls motivated by game theoretic considerations—a router performs class-based label switching which emulates user optimal service class selection with respect to selfishness—without considering the space of all aggregate-flow per-hop controls. In [16, 24] the authors study stability and efficiency properties of a parallel link routing system where selfish users are allowed to choose which link to send their traffic commensurate with their performance requirements. The QoS experienced in each link is a function of how many users have selected that link—flow-aggregation—and interpreting each link as a service class (albeit without work conservation coupling) yields a aggregate-flow QoS provisioning system.

In the differentiated services arena, efforts are directed at designing network architectures with the aim of delivering service levels with "soft" or weak guarantees using aggregate-flow—as opposed to per-flow—traffic control [5, 22], with primary emphasis on scalability. Assured Service [5, 14] and Premium (or Expedited) Service [22, 15]—two principal proposals adopted by the IETF Diff-Serv Working Group—affect weak protection through traffic shaping/marking at the edge and differentiated packet treatment support from routers. In both cases, it is assumed that service level (i.e., QoS) is computed using admission control, and the core task revolves around providing protection from ill-behaving flows that exceed their contract specifications. This is done through 2-state (in/out) marking using RIO gateways [5], or through leaky bucket traffic shaping with routers implementing priority queueing [22]. The most challenging problem—how to efficiently compute service level agreements (SLA)—is not addressed within its scope. It is this problem, phrased in the context of shaping end-to-end absolute QoS over per-hop relative QoS, that is studied in this paper.

Several works have addressed the performance evaluation side of differentiated services, both quantitatively and qualitatively. In [21], the authors advanced simplified models of Assured Service and Premium Service and analyzed them with respect to their performance when compared with simulation-based evaluations. In [19] the authors investigated enhanced mechanisms that improve the throughput and fairness properties exhibited by Assured Service. Feng *et al.* [11] describe an adaptive 1-bit marking scheme and the resulting bandwidth sharing behavior is demonstrated using simulations when the priority level is controlled end-to-end. In [9], the authors describe a proportional differentiation model which seeks to achieve robust, configurable service class separation—i.e., QoS differentiation—with the support of two candidate packet schedulers. They use simulation to study the behavioral properties. In Smart Market [20], pricing—in the form of packets carrying bids—is used to resolve scheduling conflicts of packets at switches inside a network implementing priority queues. Paris Metro Pricing [23] provides a framework and argument for discussing the role of service differentiation through pricing, even if network resources are plentiful. In [32], the authors study the efficiency properties of RIO which is used as the scheduler in Assured Service. Although the concerns raised with respect to RIO being an efficient scheduler are well-founded, the authors' contribution is just that—performance analysis of a packet scheduler—and does not relate in an essential way to aggregate-flow scheduling which is at the heart of differentiated services.

3 Network Architecture

3.1 Set-up

Assume there are n flows or users. A user $i \in [1, n]$ sends a traffic stream at average rate $\lambda_i \geq 0$ (bps)[1]. Let $\mathbf{x}^i = (x_1^i, x_2^i, \ldots, x_s^i)$ denote the vector of end-to-end QoS rendered to user i. For example, x_1^i may represent mean delay, x_2^i packet loss rate, x_3^i delay jitter and so forth. We assume that all QoS measures are represented such that a smaller magnitude means better QoS. A packet belonging to user

[1]Unless otherwise specified, we will use $[a, b]$, for $a \leq b$, to denote the set of integers between a and b.

i is inscribed with a scalar $\eta_i \in \{1, 2, \ldots, L\}$ taking on L distinct values. Typically, the number of users is very large vis-à-vis the range of η_i, i.e., $n \gg L$, and per-flow identity is lost as soon as a packet enters the network. Thus by the many-to-one mapping implied by $n > L$, QoS control via aggregate-flow packet scheduling is imposed on per-hop behavior and executed at routers on an end-to-end path.

3.2 Per-hop Control: Aggregate-flow Scheduling

Per-hop control consists of a *classifier* and a *packet scheduler*. We assume a packet scheduler with m service classes and service weights $\alpha_k \geq 0$, $\sum_{k=1}^{m} \alpha_k = 1$, for an output port whose link bandwidth μ is shared in accordance with the service weights. The most important component—where decisions about differentiated treatment of packets inscribed by different η values occurs—is the classifier which is given by a map, $\xi : [1, L] \rightarrow [1, m]$. That is, n flows—effectively L aggregated flows (or "super flows") from the router's perspective since packets are scheduled by their label values only—routed to the same output port on a switch are mapped to m service classes. For *aggregate-flow* control, $n > L$ and $L \geq m$. Thus, $n > L \geq m$, and if $L > m$, this leads to a further aggregation per-hop in addition to the many-to-one mapping exercised at the edge due to $n > L$. For example, if $n = 10000$ and $L = m = 16$, then ten thousand users mark their IP packets with four bits in the TOS field of the IP header, and routers are configured with sixteen service classes.

For some choice of classifier and packet scheduler, the QoS received by flow $i \in [1, n]$ at a switch is determined—explicitly or implicitly—by a performance function x^i, $\mathbf{x}^i = x^i(\boldsymbol{\eta}, \boldsymbol{\lambda})$, where $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$. More precisely, flow i's performance, in the aggregate-flow case, is determined by the performance function $x^k(\boldsymbol{\eta}^a, \boldsymbol{\lambda}^a)$ associated with service class $k \in [1, m]$ where $k = \xi(\eta_i)$, $\boldsymbol{\eta}^a = (1, 2, \ldots, L)$, $\boldsymbol{\lambda}^a = (\lambda_1^a, \lambda_2^a, \ldots, \lambda_L^a)$, with $\lambda_j^a = \sum_{i:\eta_i = j} \lambda_i$. That is, the switch sees only (up to) L super users or aggregate flows.

3.3 Per-hop Control: QoS Shaping Properties

There are three properties of the per-hop control which are relevant from a QoS control perspective [31]. Let $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ denote the unit vector whose i'th ($i \in [1, n]$) component is 1 and 0 for all other components. The properties are[2]:

(A1) for each flow i and configuration $\boldsymbol{\eta}$, $x^i(\boldsymbol{\eta} + \mathbf{e}_i) \leq x^i(\boldsymbol{\eta})$ and $x^i(\boldsymbol{\eta} - \mathbf{e}_i) \geq x^i(\boldsymbol{\eta})$;

(A2) for any two flows $i \neq j$ and configuration $\boldsymbol{\eta}$, $x^j(\boldsymbol{\eta} + \mathbf{e}_i) \geq x^j(\boldsymbol{\eta})$ and $x^j(\boldsymbol{\eta} - \mathbf{e}_i) \leq x^j(\boldsymbol{\eta})$;

(B) for two flows $i \neq j$ and configuration $\boldsymbol{\eta}$, $\eta_i \geq \eta_j$ implies $x^i(\boldsymbol{\eta}) \leq x^j(\boldsymbol{\eta})$.

Property (A1) states that, other things being equal, increasing the label value of flow i improves the QoS received by flow i. Property (A2) states that increasing η_i will not increase the QoS received by any other flow j. Property (B) states that if flow i has a higher η value than flow j, then the QoS it receives is superior to that of flow j. We call property (B) the *differentiated service* property—the property

[2]In the definitions, the range of $\boldsymbol{\eta}$ is such that the perturbations remain in the n-dimensional lattice, i.e., $\boldsymbol{\eta} + \mathbf{e}_i, \boldsymbol{\eta} - \mathbf{e}_i \in [1, L]^n$.

focused on by IETF Diff-Serv—and we call (A1) the *QoS shaping* property. (B) has the immediate consequence $x^i(\boldsymbol{\eta}) = x^j(\boldsymbol{\eta})$ iff $\eta_i = \eta_j$. Thus there is no absolute, a priori QoS level attached to the η_i values. It is the magnitude of η_i—*relative* to other flows' label values—that will determine the QoS received by a flow i. Thus the use of the term *relative QoS*.

3.4 End-to-end QoS Control: Adaptive Label Control

The properties exported by per-hop control—if satisfied—are not sufficient by themselves to render end-to-end QoS commensurate with user requirements. End-to-end (or edge) control complements per-hop control by setting the value of η on a per-flow basis in accordance with user needs. We assume that the network exercises *access control* at the edge such that users are not permitted to assign η values to their packets arbitrarily: if every user assigns the maximum η value L to their flows, then QoS control via η loses its meaning (degenerates to FIFO-based best-effort service by property (B)).

User i's QoS requirement, in general, can be represented by a *utility function* U_i which has the form $U_i(\lambda_i, \mathbf{x}^i, p_i)$ where λ_i is the traffic rate, $\mathbf{x}^i$ the end-to-end QoS received, and p_i the unit price charged by the service provider. The total cost to user i is given by $p_i\lambda_i$. We assume that U_i satisfies the following monotonicity properties: increasing with λ_i, decreasing with $\mathbf{x}^i$, and decreasing with p_i. If η control is allowed to be exercised by the user, then a *selfish* user i can be defined as performing the self-optimization

$$\max_{\eta_i \in [1,L]} U_i(\lambda_i, \mathbf{x}^i, p_i). \tag{3.1}$$

η_i influences user i's utility U_i via its effect on the QoS received $\mathbf{x}^i$. We assume $p_i(\mathbf{x}^i)$ is a monotone (non-increasing) function of $\mathbf{x}^i$—better QoS incurs higher cost—which corresponds to the price function exported by the service provider. Consider *threshold utilities* where user preferences are specified as bounds on QoS measures—e.g., bounds on delay, packet loss rate, jitter, and so forth—and user i's QoS requirement is given by the vector of QoS upper bounds $\boldsymbol{\theta}^i$. The control law

$$\frac{d\eta_i}{dt} = (-1)^a \varepsilon \, ||\mathbf{x}^i(\boldsymbol{\eta}) - \boldsymbol{\theta}^i|| \tag{3.2}$$

is a specific instance of adaptive label control which can be shown to be asymptotically stable. Here, $a = \mathrm{sgn}(||\mathbf{x}^i(\boldsymbol{\eta})|| - ||\boldsymbol{\theta}^i||)$ where $\mathrm{sgn}(\cdot) = 1$ if the argument is nonnegative, and 0, otherwise. We use (3.2), with user QoS requirements represented as delay and packet loss bounds, in the performance evaluation section (Section 5) to affect end-to-end absolute QoS shaping.

4 Generalized Optimal Aggregate-flow Scheduling

4.1 Semantics of Optimal Per-hop Behavior

When n users mark their flows with a value η_i drawn from $[1, L]$ with property (A1) satisfied—larger η_i values, other things being equal, result in a greater apportionment of resources and thus better QoS—η_i

can be viewed as codifying a user's QoS or resource demand with respect to some measurement unit. In [31], the authors provide a semantics and formulation of optimal differentiated network provisioning with focus on PHB design, which then is solved to find the optimal solution and its properties. The optimality criterion is based on mean square error where after the normalization step $\hat{\eta}_i = (\eta_i - \eta_{\min})/(\eta_{\max} - \eta_{\min})$, if $\eta_{\max} \neq \eta_{\min}$, and 1, otherwise[3], the minimization problem

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (\hat{\eta}_i - \hat{\omega}_i)^2. \tag{4.1}$$

is solved for given $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$ with respect to the service weights allocated to each user flow. That is, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ with $\alpha_i \geq 0$, $\sum_{k=1}^{n} \alpha_k = 1$, represent the fraction of resources apportioned by the per-flow classifier to $i \in [1, n]$, and $\omega_i = \alpha_i/\lambda_i$ denotes the fraction of resources allocated to i per unit flow. If $m = n$ where m is the number of configured service classes, the router performs *per-flow* scheduling, in which case α_i directly corresponds to the service weight of class i. The optimal per-flow scheduling problem is nontrivial. However, the main technical problem lies in solving the *aggregate-flow* scheduling problem when $m < n$. That is, some of the flows must be aggregated—mapped to the same service class—and the individual service weights α_i ($i \in [1, n]$) are *induced* from the m service weights associated with the m aggregate-flow service classes. Both the aggregate-flow and per-flow scheduling problem share the same *end-to-end* objective function (4.1)—the router is treated as a black box— which is defined over the user space, and in [31] the authors show how to solve the optimal aggregate-flow scheduling problem using the solution of the optimal per-flow problem as a building block.

4.2 Generalized Mean Square Semantics

From an absolute QoS shaping perspective, the optimal relative QoS exported by routers according to criterion (4.1) has the following potential weakness. Assume a service provider, over time, observes that its customer base and corresponding QoS requirement profile varies, but includes as stable subpopulations user groups requiring 0.0001, 0.001, 0.005, 0.05, 0.1, 0.3, and 0.6 packet loss rates. Consider Figure 4.1 (left) which shows the packet loss rates exported by an optimal aggregate-flow scheduler with $m = 16$ service classes as link bandwidth is varied[4]. We observe that optimal aggregate-flow scheduling exports relative QoS with "equal spacing" as captured by the optimization criterion (4.1). The QoS spacing between service classes has a finite resolution $\Delta c \approx 0.06$ or 0 (degenerate case) as determined by m. For 0.0001-, 0.001-, 0.005-packet loss rate user flows, the consequent coarseness forces them to choose label values that map them essentially to the same service class whose QoS—for the three user groups to be satisfied—is dominated by the *most stringent QoS requirement* 0.0001. Thus 0.001- and 0.005-packet loss flows become "oversatisfied" or overprovisioned which, in turn, implies that resources are inefficiently utilized.

[3] $\eta_{\min}$ and $\eta_{\max}$ are the minimum and maximum values of $\{\eta_1, \eta_2, \ldots, \eta_n\}$, respectively.

[4] The performance results are from the simulation set-up described in Section 5

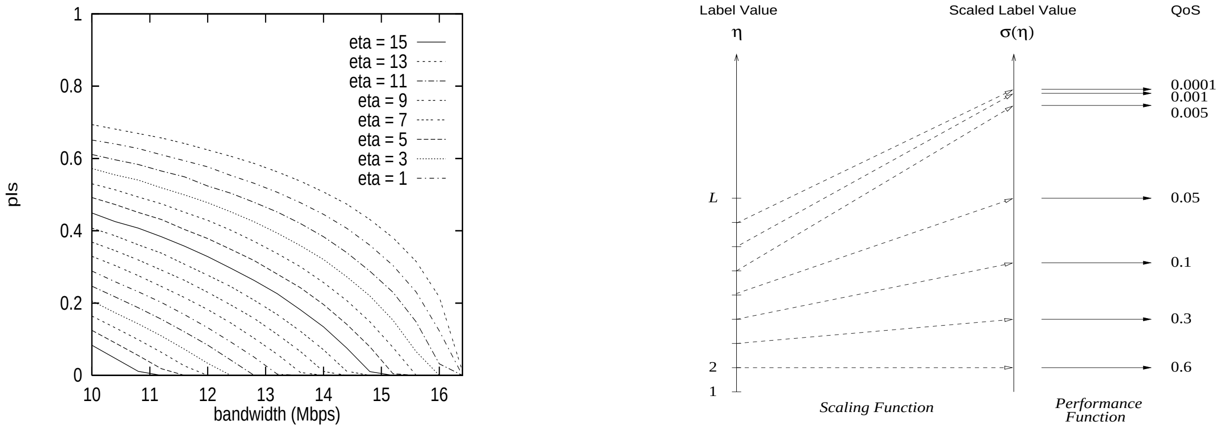

Figure 4.1: Left: QoS separation achieved by optimal aggregate-flow classifier with mean square criterion (4.1) when $L = 16$. Right: Scaling function σ affecting nonuniform stretching and contraction.

4.3 Generalized Optimal Aggregate-flow Scheduling

Given the potential inefficiency and structural limitation pointed out in the previous section, we generalize the mean square error criterion by introducing a scaling function

$$\sigma : [1, L] \rightarrow \mathbb{R}_+$$

where σ is monotone increasing in its argument. This scaling function can be configured by the service provider—we envision this to be done at larger time scales—to apply nonuniform scaling to the TOS field label values commensurate with its observed customer QoS profile before they are passed to the classifier proper. This is illustrated in Figure 4.1 (right). The generalized optimization criterion has the form

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (\widehat{\sigma(\eta_i)} - \hat{\omega}_i)^2 \tag{4.2}$$

for given $\boldsymbol{\eta}$, $\boldsymbol{\lambda}$, and σ. First, for the per-flow case, i.e., $n = m$, the solution to (4.2) is given by the next result. In the algorithm implementing generalized optimal aggregate-flow scheduling, the per-flow solution is used as a subroutine.

Theorem 4.3 *Given $\boldsymbol{\eta}, \boldsymbol{\lambda} \in \mathbb{R}^n_+$, and $\sigma(\cdot)$, the solution to (4.2) is*

$$\alpha_i = (1 - \nu) \frac{\lambda_i \widehat{\sigma(\eta_i)}}{\sum_{j=1}^{n} \lambda_j \widehat{\sigma(\eta_j)}} + \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \qquad i \in [1, n], \tag{4.4}$$

where $0 \leq \nu < 1$ is a parameter defining a continuous family of solutions. Moreover, the solution is unique in that any solution to (4.2) must be of the form (4.4) for some $0 \leq \nu < 1$.

Note that the optimal per-flow solution is a convex combination of two parts, the first term which weights each flow by the normalized scaled label values, and a second term which is void of weighting.

It is the first term which facilitates relative—and, now, programmable—QoS differentiation. Parameter ν in the second term controls how much FIFO sharing is injected, which affects QoS resolution (cf. also performance results in Section 5) whereas $\sigma(\cdot)$ determines relative QoS spacing. It can be shown[5] that (4.4) satisfied properties (A1), (A2), and (B). A proof of Theorem 4.3 is given in the Appendix.

Now, to the solution of (4.2) for aggregate-flow scheduling, i.e., $m < n$. Aggregate-flow per-hop control, whether it has many or few labels, must service n flows using $m < n$ service classes. This results in information loss and coarsified resolution which, in turn, affects a reduced ability to effectively shape end-to-end QoS with respect to the performance criterion when compared to per-flow control. An *aggregate-flow per-hop control with parameter* (m, L) is a function

$$\Phi_{m,L} : (\boldsymbol{\eta}, \boldsymbol{\lambda}) \mapsto (\xi, \boldsymbol{\alpha}) \tag{4.5}$$

where $\xi : [1, L] \to [1, m]$ is the *classifier* and $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^m)$ is the vector of service weights assigned to the m service classes. With respect to end users, $\Phi_{m,L}$ induces—explicitly or implicitly—a performance function $\varphi^i_{m,L}$ for each user $i \in [1, n]$, $\varphi^i_{m,L} : (\boldsymbol{\eta}, \boldsymbol{\lambda}) \mapsto \alpha_i$, where $\alpha_i = \varphi^i_{m,L}(\boldsymbol{\eta}, \boldsymbol{\lambda}) \geq 0$ is user i's share of the bandwidth allocated by $\Phi_{m,L}$. Consider a special type of aggregate-flow per-hop control $\Phi_{m,L}$—called *reduction classifier*—whose behavior is determined by its classifier $\xi : [1, L] \to [1, m]$ in the following way. Let $U_k = \{i \in [1, L] : \xi(i) = k\}, k \in [1, m]$, be the partition of $[1, L]$ induced by ξ. Then, a reduction classifier $\Phi_{m,L}$ reduces the L label-n user problem to an m user (or "superuser") per-flow classification problem by aggregation of component flows and centroid computation, then solves the reduced problem by applying the optimal per-flow scheduling solution. The next result shows that (4.2) is minimized by a reduction classifier.

Theorem 4.6 *Let $\Phi_{m,L}$ be a reduction classifier represented by its classifier ξ. Then $\Phi_{m,L}$ is an* optimal aggregate-flow scheduler, *i.e., satisfies* (4.2) *if and only if ξ is a solution to*

$$\min_{\xi'} \sum_{k \in [1,m]} \sum_{i \in U_k} (\widehat{\sigma(\eta_i)} - \hat{\eta}^k)^2 \tag{4.7}$$

where

$$\hat{\eta}^k = \begin{cases} 0, & \text{if } \exists i \in U_k,\ \widehat{\sigma(\eta_i)} = 0; \\ 1, & \text{if } \exists i \in U_k,\ \widehat{\sigma(\eta_i)} = 1; \\ \sum_{i \in U_k} \widehat{\sigma(\eta_i)} / |U_k|, & \text{otherwise;} \end{cases}$$

and the minimum ranges over all reduction classifiers ξ'.

Theorem 4.6 shows that an optimal aggregate-flow classifier must be a reduction classifier, and furthermore, it must efficiently cover—in the generalized mean square sense—the set of label values $\{\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_n\}$ using m centroids $\{\hat{\eta}^1, \ldots, \hat{\eta}^m\}$. Thus, aggregate-flow scheduling is, mathematically, an *optimal clustering problem*. Unlike its many brethren in higher dimensions that are NP-complete [12],

[5]This is omitted due to space constraints.

however, the clustering problem given by (4.7) has a poly-time algorithm; e.g., it can be solved by dynamic programming. When $L = m$, the practically most relevant case where there are as many labels as service classes, optimal aggregate-flow classification has a *linear time* algorithm. Thus, for $\Phi_{L,L}$, the classifier becomes trivial—degenerates to an identity function, i.e., $\xi(k) = k$, $k \in [1, L]$—which obviates the clustering computation, and all operations can be done in $O(n)$ time which is of the same order of time complexity needed to inspect the label values of incoming packets, a minimal requirement[6].

4.4 Load Imbalance and Local QoS Responsibility

In this section, we consider the efficiency and properties of QoS shaping of the optimal differentiated services architecture across *multiple* routers on an end-to-end path in a wide area network. To motivate the problem. consider a flow i with end-to-end delay requirement 30ms whose traffic is routed through a 6-hop path shown in Figure 4.2. Assume that the load distribution of the routers is nonuniform—typically, only some routers are bottlenecks or hot spots along an end-to-end path—with routers 1 and 5 highly loaded and routers 2, 3, 4, and 6 being lightly loaded. A *uniform* QoS assignment—as induced

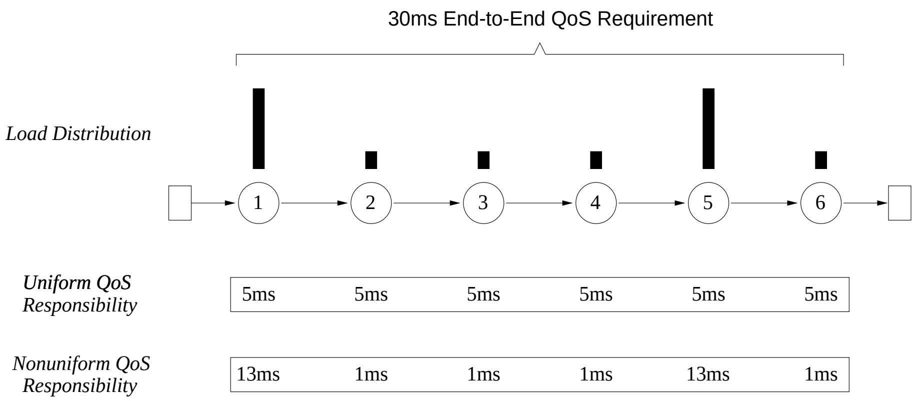

Figure 4.2: Uniform vs. nonuniform local QoS responsibility distribution to satisfy 30ms end-to-end delay requirement for a given load imbalance.

by flow i's choice of label value η_i—would impose the same 5ms (igoring, for simplicity of exposition, link latencies) as the local QoS responsibility at each router irrespective of its load. A more desirable—and, in general, efficient—allocation is one where heavily loaded routers are given less responsibility than lightly loaded ones[7]. This is illustrated in the nonuniform QoS distribution in Figure 4.2 where the highly loaded routers export 13ms local queueing delay whereas the lightly loaded hops achieve 1ms delays for a total of 30ms. The aforementioned property is not difficult to satisfy. A more subtle—and, from a control perspective—important property for end-to-end QoS shaping is the *change in local QoS responsibility as a function of* η. For example, if flow i increases its label value η_i because the current

[6]More precisely, the time complexity is $O(n \cdot L)$, however, since the number of labels L is a small constant (at most 256 if all TOS bits are used) we treat L as a constant and write $O(n)$.

[7]This can be formally attributed in several ways, for example, using congestion pricing and utilization as cost measures [3].

delivered QoS is deficient vis-à-vis its requirement, other things being equal, it is desirable that the lightly loaded routers take on a greater share of request for improved QoS than heavily loaded ones. We show next that both properties are satisfied by the optimal aggregate-flow scheduler.

Let

$$\boldsymbol{\rho} = (\rho_1, \rho_2) = \left(\frac{\sum_{j=1}^{n} \lambda_j \widehat{\sigma(\eta_j)}}{(1-\nu)\mu}, \frac{\sum \lambda_j}{\nu\mu} \right)$$

be the *load factor* at a switch with bandwidth μ. Assume flow i traverses a r-hop path, and let μ^k denote the link bandwidth of router $k \in [1, r]$. Let $\boldsymbol{\rho}^k = (\rho_1^k, \rho_2^k)$ be the load factor at switch $k \in [1, r]$. Let α_i^k represent the share of bandwidth of flow i at switch k. The bandwidth per unit flow of flow i at switch k is $\omega_i^k = \frac{\alpha_i^k \mu^k}{\lambda_i}$.

Proposition 4.8 *For all routers $k \in [1, r]$ on the end-to-end path, the resource share (and thus QoS) apportioned to flow i satisfies*

$$\omega_i^k = \frac{\widehat{\sigma(\eta_i)}}{\rho_1^k} + \frac{1}{\rho_2^k}. \tag{4.9}$$

It follows immediately that $\omega_i^k \le \omega_i^l$ if $\boldsymbol{\rho^k} \ge \boldsymbol{\rho^l}$. Other things being equal, a router that is more heavily loaded will carry a lighter QoS responsibility.

The next result shows that nonuniform QoS responsibility holds even with respect to a change of η_i. We show the expression using continuous notation for expositional clarity (the discrete counterpart is more cluttered).

Theorem 4.10 *For all $k \in [1, r]$,*

$$\frac{\partial \omega_i^k}{\partial \widehat{\sigma(\eta_i)}} = \left(\frac{1}{1 + B_i^k} \right) \frac{1}{\rho_1^k} \tag{4.11}$$

where $B_i^k = \frac{\lambda_i \widehat{\sigma(\eta_i)}}{\sum_{j=1}^{n} \lambda_j \widehat{\sigma(\eta_j)}}$ is the portion of virtual rate for flow i at switch k.

As with Proposition 4.8, the share of change in QoS responsibility is inversely proportional to ρ_1^k, however, it does not depend on ρ_2^k (FIFO term).

5 Performance Results

5.1 Simulation Set-up

We use the LBNL Network Simulator *ns* (version 2) as our simulation platform. We have extended *ns* into a wide area network QoS benchmarking environment—called *QSim* [4]—which was used in our earlier work on WAN QoS performance evaluation [2, 3]. QSim exports a selection of per-hop behaviors and end-to-end controls, along with a QoS interface through which user QoS requirements can be specified. The control parameters and network configuration including topology, network resources, flow pattern, and the choice of packet scheduling, classifier, and end-to-end QoS control are configurable via a GUI. The latest version incorporates the optimal per-hop behavior and adaptive label control advanced in [31] and benchmarked in this paper.

5.1.1 Per-hop and End-to-end Protocol Specification

The optimal per-hop control consists of a classifier and a packet GPS scheduler (the weighted fair queue implementation resident in *ns*) which is configured to have m service classes. In the following, $m = L$ (cf. Section 4.4) where L is the total number of label values. For each incoming packet, the optimal aggregate-flow per-hop control does the following:

- The classifier inspects the η value carried in the TOS field of the packet's IP header, and puts the packet with value η into service class[8] $\eta \in \{1, 2, \dots, L\}$.
- Based on a fixed time interval T_s (T_s is set at $100ms$ in the results reported here):
  - The classifier periodically measures the average arrival rates $\lambda_1, \cdots, \lambda_m$ of all the classes over T_s (a simple counting operation).
  - Based on the measured "instantaneous" arrival rate of each service class, the classifier adjusts the service weights of all classes according to (4.4).
- The packet GPS scheduler then schedules the packets based on the new service weights. (The scheduler uses the current weights to compute the finish time of backlogged packets.)[9]

The end-to-end adaptive label control performs the following two tasks:

- Measurement and feedback: the receiver measures the QoS received over a fixed window of length T_u ($T_u = 1s$ in our simulation), and sends feedback packets using UDP with label value $L - 1$.
- Adaptive label control: after the k'th feedback is received, the sender compares measured end-to-end performance with the QoS target and sets the new η value based on (3.2).

The above implementation scheme is a straightforward adaptation of the logical control protocol. Further fine-tuning of its components may yield additional performance gains which are being explored in the implementation work.

5.1.2 Network Configuration

We use two benchmark network topologies, the 2-switch topology shown in Figure 5.1 (left) which isolates a bottleneck link, and the 4-switch caterpillar topology shown in Figure 5.1 (right) which is comprised of multiple bottleneck links. The 4-switch caterpillar topology emulates the physical topology

[8] To be precise, η ranges from 0 to $L - 1$, the literal concatenated bit patterns available in the TOS field.

[9] We remark that the original weighted fair queue implementation of *ns* is a rough approximation of ideal GPS (it uses real-time, instead of virtual time, which can dilate the fairness bound of PGPS); however, it has sufficed for our performance evaluation purposes. More accurate implementations can only help, although, we suspect incrementally.

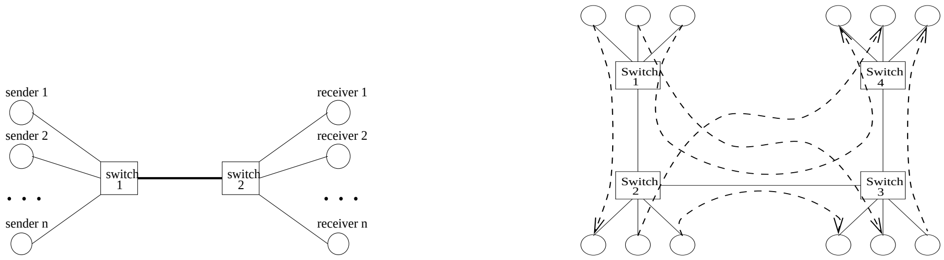

Figure 5.1: Benchmark network topology. Left: 2-switch single bottleneck link shared by n flows. Right: 4-switch multiple bottleneck link caterpillar topology.

of *Purdue Infobahn*—an IP-over-SONET QoS backbone comprised of Cisco 7206 VXR routers—used as a QoS testbed at the Network Systems Lab at Purdue University, wherein our optimal aggregate-flow per-hop control architecture is being implemented in Cisco IOS. The simulation environment allows us to carry out larger scale performance studies than is possible on the phyisical network (e.g., processing speed of interface processors and number of interface ports), whereas the latter's primary purpose lies in experimental studies with QoS-enabled network applications driven by human users. Most of the performance results reported in the paper are from the bottleneck topology shown in Figure 5.1 (left) whose results are confirmed by those carried out for the topology shown in Figure 5.1 (right). The number of service classes were varied up to 32, and the number of user flows from 16 up to 450.

5.2 Service Differentiation

In this section, we investigate the power of generalized optimal aggregate-flow per-hop control to provide service differentiation. We show how the weighting parameter ν and scaling function $\sigma(\cdot)$ can affect the QoS separation among service classes. In practice, these two knobs can be tuned by ISP to provide customized service differentiation. The benefit of scaling function is further examined by efficiency aspect. We show that system efficiency can be improved if proper scaling function is adopted.

First, we examine the QoS separation performance of optimal aggregate per-hop control without scaling function. We show the QoS shaping as a function of bottleneck bandwidth when $L = 16$ and there are two flows per class (the number of flows, in this instance, is irrelevant since their impact is subsumed by volume).

Figure 5.2 shows the QoS separation achieved in the $L = 16$ service classes when executing the optimal aggregate-flow service weight assignment by the classifier. The left figure shows the shape of the performance function for packet loss rate and its QoS separation across the 16 service classes (all classes are nonempty) as bottleneck bandwidth is increased from 10 Mbps to about 16 Mbps. We observe that even, uniform separation is preserved (property (B) but in a stronger form), with packet loss rate degenerating to zero as resources becomes plentiful. Only the odd η values are plotted to reduce cluttering. Figure 5.2 (right) shows the corresponding performance results for end-to-end delay (in ms). We observe that the performance function for delay is less uniform—the performance gap

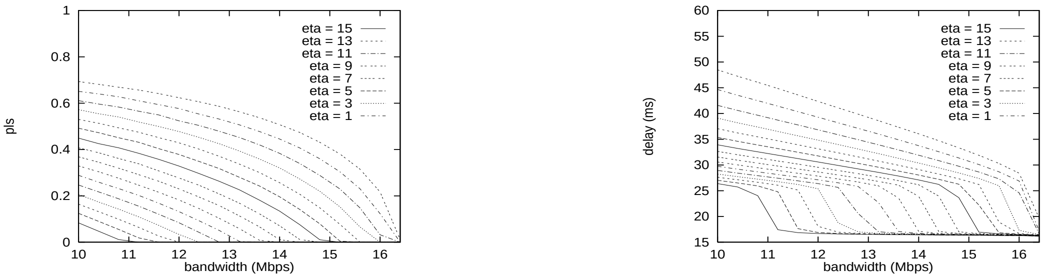

Figure 5.2: QoS separation achieved by optimal aggregate-flow classifier when $L = 16$. Left: packet loss rate. Right: end-to-end delay.

decreases as η approaches $L-1$—while satisfying property (B). This can be explained, in part, by the fact that waiting time W_k in service class k is queue length Q_k divided by service rate $\alpha_k \mu$. Since the optimal aggregate-flow classifier satisfies $\omega_L \geq \omega_{L-1} \geq \cdots \geq \omega_1$, the shape of the end-to-end delay curve is not unexpected.

5.2.1 Impact of Weighting Parameter ν

Recall that the service weight assignment, equation (4.4), has a weighting parameter $0 \leq \nu \leq 1$ which can be controlled to determine the relative importance of service differentiation at the router. As discussed in Section 4, $\nu \to 1$ diminishes the contribution of the differentiation component (first term of (4.4)) thus turning the classifier into a FIFO scheduler where label values are effectively ignored. Conversely, as $\nu \to 0$, the contribution from the FIFO component (second term) is eliminated thus turning the classifier into a maximal differentiator. This is shown in Figure 5.3.

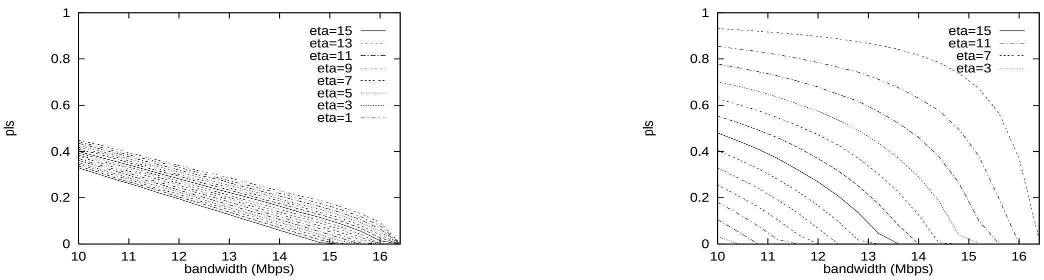

Figure 5.3: Impact of ν on QoS separation for $L = 16$. Left: QoS exported by service classes as a function of bottleneck bandwidth when $\nu = 0.9$. Right: Corresponding plots when $\nu = 0.1$.

The left figure shows QoS separation as a function of bottleneck bandwidth for $L = 16$ when $\nu = 0.9$ which make the system close to FIFO scheduler. Figure 5.3 (right) shows the corresponding plot when the weighting parameter is set to $\nu = 0.1$ thus amplifying the contribution of the differentiation component. We observe that the *range* of QoS differentiation is significantly larger for $\nu = 0.1$ than $\nu = 0.9$—the QoS spacing between $\eta = 0$ and $\eta = 15$ is about 0.1 for the latter whereas it is about 0.8 for the former. However, this occurs at the cost of reduced resolution with respect to fine-granular QoS spacing. Thus, when viewing L as a resource, ν influences how QoS is to be shaped at the switch—to achieve fine-granular QoS spacing or coarse-granular QoS separation (i.e., range). From a network management perspective, ν is a control parameter that the service provider may engage to reflect the overall needs of its customer base.

5.2.2 The Role of Scaling Function

In this section, we study the impact of the scaling function. Figure 5.4 shows how the scaling function can affect the QoS separation among service classes.

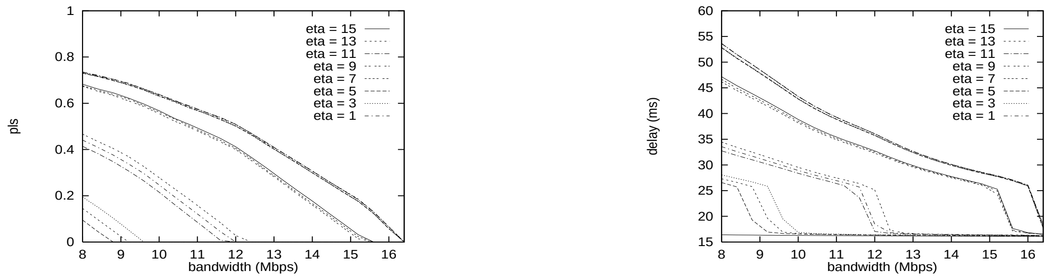

Figure 5.4: QoS separation achieved by optimal aggregate-flow classifier with scaling function. $\sigma(\eta)$, $\eta \in [0, 15]$: 1.0, 1.1, 1.2, 10, 11, 12, 100, 110, 120, 500, 550, 600, 1000, 1100, 1200, 2000. Left: packet loss rate. Right: delay.

Next we show that the scaling function can improve system efficiency. First define $\mathcal{A}^*$ as the set of configurations where all user requirements are satisfied. For a system with certain label set and scaling function, given the QoS targets and traffic patterns of all the users, the existence or nonemptiness of $\mathcal{A}^*$ ("Nirvana") is decided by total system resources which is bottleneck bandwidth in our case. We will examine system efficiency by finding the minimum bottleneck bandwidth needed to achieve $\mathcal{A}^* \neq \emptyset$.

The size of the label set L can impact the existence or nonemptiness of $\mathcal{A}^*$ ("Nirvana") which is shown in Figure 5.5. The plot shows the minimum bottleneck bandwidth needed to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L. We observe that, other things being equal, too small an L value incurs a high QoS shaping penalty with respect to the network system being able to satisfy the QoS requirements of all users, even if sufficient bandwidth is available. On the other hand, the marginal or incremental benefit of increasing L diminishes for given user QoS requirements, which points toward the possibility that with "not-too-large" L (perhaps a subset of the TOS bits) the QoS requirements of a multitude of users can be met.

In Figure 5.5, the top curve is obtained without scaling function. The second one is a good estimation of what optimal scaling function can achieve. In other words, for given L, if we are allowed to choose σ, what is the minimal bandwidth needed. (Note we may choose different σ for different L).

We observe the following:

- The system achieves maximum efficiency when $L \geq 8$. This is because we configured 8 groups of users with different QoS requirements.

- The scaling function does not help when $L = 1, 2$. This is due to the system is FIFO when $L = 1$, and we always get $\widehat{\sigma(\eta)} = 0, 1$ after hat transformation when $L = 2$.

- The increase in efficiency is large when $L = 4, 8$. It however becomes smaller when $L > 8$.

For this particular configuration, the improvement in efficiency by scaling function is significant.

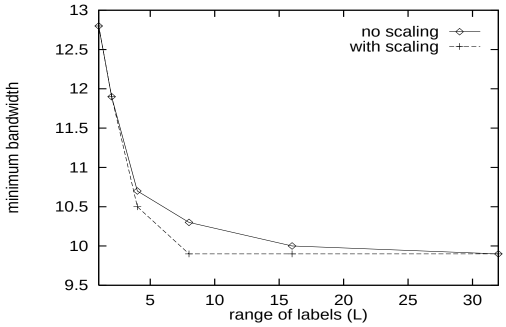

Figure 5.5: Impact of scaling function on existence of $\mathcal{A}^*$: minimum bottleneck bandwidth required to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L.

5.2.3 Impact of Burstiness

The previous results have shown the QoS shaping properties of the optimal aggregate-flow per-hop control when input traffic is CBR. In this section, we inject burstiness—an ever-present orthogonal dimension to QoS separation—and show that the qualitative behavior of the aggregate-flow QoS provisioning remains the same. Figure 5.6 shows QoS separation results across $m = 16$ aggregate-flow service classes for the same network configuration as Figure 5.2 and Figure 5.4 except for the difference that the traffic sources are VBR—Poisson—with the same average data rates.

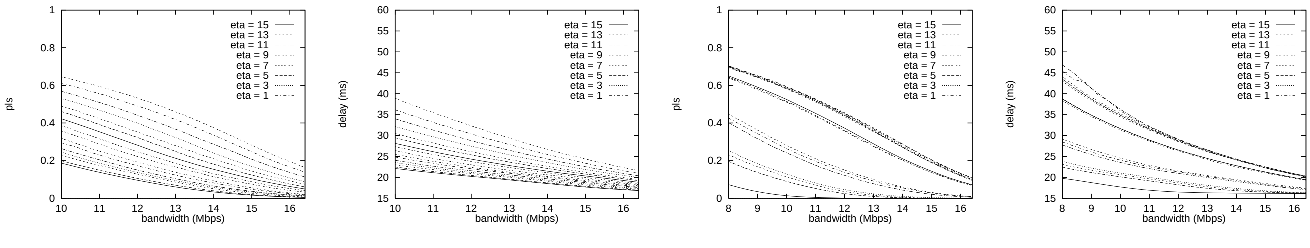

Figure 5.6: QoS separation achieved by optimal aggregate-flow classifier when $L = 16$ under VBR traffic. Left two figures: packet loss rate and end-to-end delay achieved without scaling function. Right two figures: packet loss rate and end-to-end delay achieved with scaling function $\sigma(\eta)$, $\eta \in [0, 15]$: 1.0, 1.1, 1.2, 10, 11, 12, 100, 110, 120, 500, 550, 600, 1000, 1100, 1200, 2000.

Both the packet loss and delay plots show that QoS separation is achieved, and moreover, burstiness imparts a more gradual (or smooth) change in the overall QoS as a function of bottleneck bandwidth due to the absence of threshold effect characteristic of CBR traffic.

5.3 Load Imbalance

In this section, we demonstrate the results about load imbalance that we derived in section 4.4.

The simulation setup is follows: let user flows traverse multiple hops where they share the bandwidth with cross traffics. The cross traffics are configured such that each class of the switches will receive some amount of cross traffic. Therefore, by varying the volumes of cross traffics, we can effectively and precisely control the load of the switches.

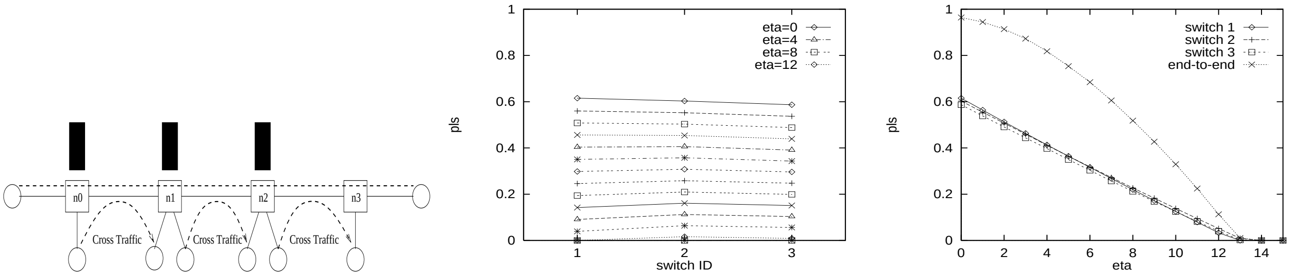

Figure 5.7: QoS distribution on multi-hop path with three medium-loaded switches. Left: switch loads. Middle: static QoS distribution. Right: dynamical QoS distribution.

We first consider the scenario when all the switches are evenly loaded. Figure (5.7) shows the QoS distribution feature of the system. The left figure describes the topology and the load on each switch. The middle figure shows the QoS achieved by each class $1, \cdots, L$ at each switch. Since all the switches have similar load, the QoS experienced by each class at different switches are almost the same. The right figure shows the change in end-to-end QoS and the QoS achieved at each switch when one user changes its η value from 0 to 15. As expected, since all switches are offered similar loads, the switches take the same responsibilities to improve the QoS of the user.

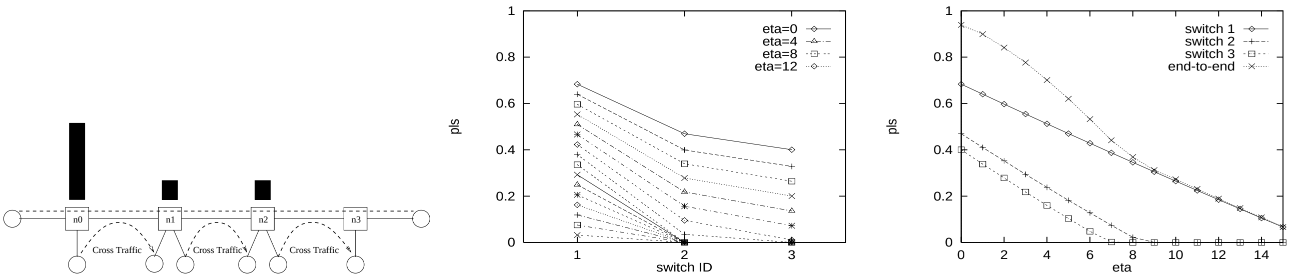

Figure 5.8: QoS distribution on multi-hop path with one heavy-loaded switch and two light-loaded switches. Left: switch load. Middle: static distribution. Right: dynamical distribution.

Next we study the QoS distribution among switches when they are applied different loads as described by Figure 5.8 (left). Figure 5.8 (middle) shows the static QoS distribution among the switches by plotting the QoS of all the classes $1, \cdots, L$ at each switch. As stated by Proposition 4.8 in section4.4, the classes in lightly-loaded switch experience better QoS than the corresponding class in heavily-loaded switch. Figure 5.8 (right) shows the more subtle feature, the dynamic QoS distribution among switches.

It depicts the change of end-to-end QoS and the QoS achieved at each switch when one user changes its η value from 0 to 15. In section4.4, we stated that when one user increases η value to improve its QoS, the efficient way is to put more responsibility on the lightly-loaded switch. Theorem 4.10 section4.4 states that our optimal aggregate-flow per-hop control has such property. This is verified by Figure (5.8 (right). As η increases, we observe the packet loss rate at the lighted-loaded switch decreases faster than it does at the heavily-loaded switches.

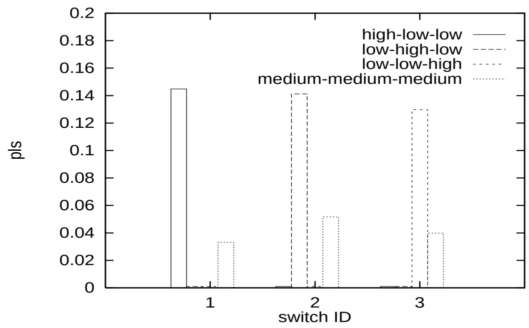

Figure 5.9: To satisfy given QoS target, the actual QoS achieved at each switch with respect to different load imbalance patterns.

The load imbalance property of the system is further demonstrated by Figure 5.9 where the switches along the path are configured with four different patterns of loads: (High, low, low), (low, high, low), (low, low, high), and (medium, medium, medium) respectively. The Figure describes the actual QoS achieved at different switches for these four different patterns to satisfy a given QoS target (packet loss rate = 0.15). We see that the system adaptively distributes QoS responsibility among the switches to meet the end-to-end QoS requirement.

5.4 Impact of the number of service classes m

In this section, we show the performance impact of the number of service class m. If $m = 1$, then we know that the optimal aggregate-flow control degenerates to FIFO scheduling. Thus small m diminishes the QoS resolution achievable by the classifier and increasing m amplifies it. However, we show that the end-to-end QoS shaping power degrades gracefully as the number of service classes at routers is decreased. This property admits heterogeneity in router configurations and thus enables incremental deployment.

Figure 5.10 shows the dependence. For $n = 32$ user flows, we show QoS exported by m service classes as a function of bandwidth as m is increased $m = 1, 2, 4, 8, 16, 32$. Thus at $m = 32$, the system degenerates to per-flow control. The weighting parameter ν of the optimal classifier (cf. equation (4.4)) is set to 0.5.

Figure 5.10 (leftmost plot) shows QoS shaping when $m = 1$, i.e., FIFO with no QoS separation. As m is increased, however, we observe that increased m endows increased QoS resolution with finer levels of QoS spacing. In fact, for $m = 2, 4, 8, 16, 32$ (i.e., excluding $m = 1$), we observe that the performance

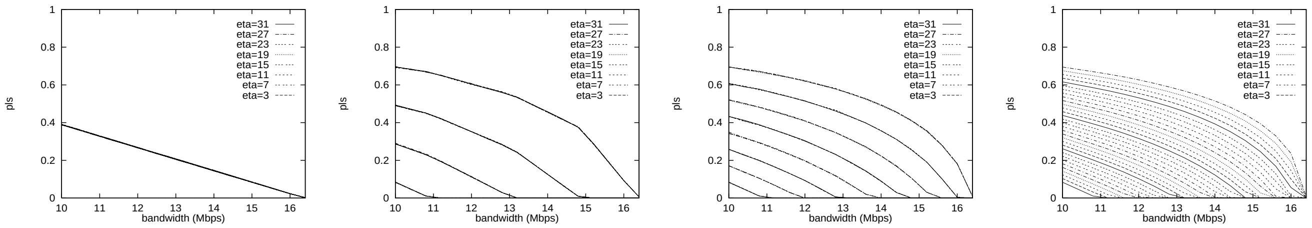

Figure 5.10: QoS exported by the service classes as a function of bottleneck bandwidth for m=1, 4, 8, 32.

curves are strict supersets of each other as indexed by m. This can be shown to follow from the form of label value normalization $\hat{\eta}$. (for details, see [31]). Since for all finite $m > 0$, $\eta \in \{1, 2, \ldots, m\}$ is mapped to a subset of the closed unit real interval $\hat{\eta} \in [0, 1]$, this also explicates the effect of the label set being discrete: a closed continuous interval $[a, b]$, $a < b$, maps *onto* $[0, 1]$, and $\mathbb{Z}_+$ is mapped to a dense subset of $[0, 1]$. Thus m translates to "QoS resolution" in a straightforward fashion.

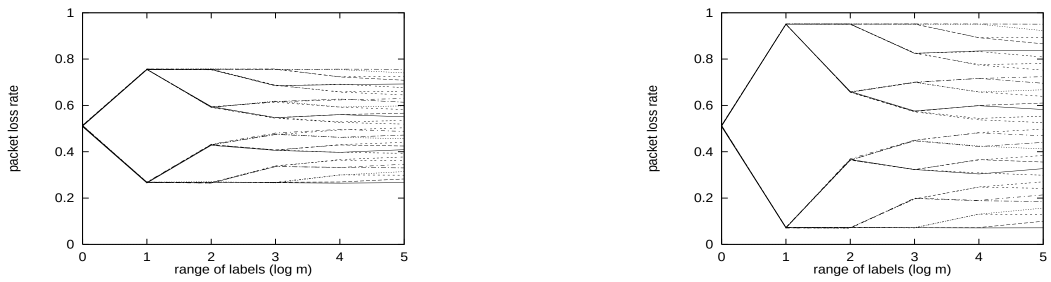

Figure 5.11: The impact of m on QoS shaping. Left: QoS exported in m service classes as a function of m for $\nu = 0.5$. Right: Corresponding plot for $\nu = 0.1$.

Figure 5.11 shows the joint influence of m and ν on QoS shaping. The left figure shows the QoS exported by m service classes as a function of m (actually its logarithm) for $m = 1,\ 2,\ 4,\ 8\ 16,\ 32$ when the weighting parameter is set to $\nu = 0.5$. As a function of m, this generates a *binary QoS tree* whose leaves at level $\log m$ show the range and values of QoS covered by the m service classes. Figure 5.11 (right) shows the corresponding plot when $\nu = 0.1$. We observe that the width of the covering has significantly amplified, however, at the cost of a more coarse-granular QoS spacing. Thus, as indicated earlier, the m service classes can be used to cover the QoS space (here, the packet loss rate space $[0, 1]$) sparsely but uniformly, or densely but concentrated in a subspace of the entire QoS space. For $m = 2^\ell$ ($\ell \geq 1$) the subset relation $\mathcal{C}_2 \subset \mathcal{C}_4 \subset \cdots \subset \mathcal{C}_m \subset \cdots$ of the coverings $\mathcal{C}_m \subset [0, 1]$ as a function of m—which is implied by the form of the hat normalization —can be observed in Figure 5.11.

6 Conclusion

In this paper, we have given a comprehensive simulation-based performance evaluation of the optimal aggregate-flow per-hop control and adaptive label control architecture advanced in [31] for facilitating

QoS provisioning in differentiated services networks. We have shown that the theoretical framework and analysis presented in [31] are confirmed in simulated dynamical network systems. Moreover, we have shown quantitative performance evaluations of both structural and dynamical properties, thus extending, in addition to complementing, our previous theoretical results. The performance results, overall, show that user-specified services can be efficiently and effectively achieved over the optimal aggregate-flow per-hop behavior network substrate when coupled with either open-loop or closed-loop (adaptive label control) edge control.

Current work is directed at implementing the QoS control architecture in Cisco IOS for benchmarking over the IP-over-SONET QoS testbed Purdue Infobahn (see Figure 5.1) which is comprised of Cisco 7206 VXR routers.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, 1998.

[2] S. Chen and K. Park. A distributed protocol for multi-class QoS provision in noncooperative many-switch systems. In *Proc. IEEE International Conference on Network Protocols*, pages 98–107, 1998.

[3] S. Chen and K. Park. An architecture for noncooperative QoS provision in many-switch systems. In *Proc. IEEE INFOCOM '99*, pages 864–872, 1999.

[4] Shaogang Chen. *Stratified Best-effort QoS Provisioning in Noncooperative Networks.* PhD thesis, Purdue University, 2000.

[5] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Trans. Networking*, 6(4):362–373, 1998.

[6] R. L. Cruz. A calculus for network delay, part I: network elements in isolation. *IEEE Trans. Inform. Theory*, 37(1):114–131, 1991.

[7] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE J. Select. Areas Commun.*, 13(6):1048–1056, 1995.

[8] G. de Veciana, G. Kesidis, and J. Walrand. Resource management in wide-area ATM networks using effective bandwidths. *IEEE J. Select. Areas Commun.*, 13(6):1081–1090, 1995.

[9] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proc. ACM SIGCOMM '99*, 1999.

[10] A. Elwalid and D. Mitra. Analysis, approximations and admission control of a multi-service multiplexing system with priorities. In *Proc. IEEE INFOCOM '95*, pages 463–472, 1995.

[11] W. Feng, D. Kandlur, D. Saha, and K. Shin. Adaptive packet marking for providing differentiated services in the Internet. In *Proc. IEEE International Conference on Network Protocols*, pages 108–117, 1998.

[12] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[13] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, 1996.

[14] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597, 1999.

[15] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. RFC 2598, 1999.

[16] Y. Korilis, A. Lazar, and A. Orda. Architecting noncooperative networks. *IEEE J. Select. Areas Commun.*, 13(7):1241–1251, 1995.

[17] A. Lazar and G. Pacifici. Control of resources in broadband networks with quality of service guarantees. *IEEE Network Magazine*, pages 66–73, October 1991.

[18] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2:1–15, 1994.

[19] W. Lin, R. Zheng, and J. Hou. How to make assured services more assured? In *Proc. IEEE International Conference on Network Protocols*, pages 182–191, 1999.

[20] J. MacKie-Mason and H. Varian. Economic FAQs about the Internet. In L. McKnight and J. Bailey, editors, *Internet Economics*, pages 27–63. MIT Press, 1996.

[21] M. May, J. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services schemes for the Internet. In *Proc. IEEE INFOCOM '99*, pages 1385–1394, 1999.

[22] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. Internet Draft, 1997.

[23] Andrew Odlyzko. Paris Metro Pricing: The minimalist differentiated services solution. In *Proc. IEEE/IFIP International Workshop on Quality of Service*, 1999.

[24] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Trans. Networking*, 1(5):510–521, 1993.

[25] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Networking*, 1(3):344–357, 1993.

[26] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Networking*, 2(2):137–150, 1994.

[27] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proc. IEEE International Conference on Network Protocols*, pages 171–180, 1996.

[28] K. Park, M. Sitharam, and S. Chen. Quality of service provision in noncooperative networks: heterogeneous preferences, multi-dimensional QoS vectors, and burstiness. In *Proc. 1st International Conference on Information and Computation Economies*, pages 111–127, 1998.

[29] K. Park and W. Willinger, editors. *Self-Similar Network Traffic and Performance Evaluation*. Wiley-Interscience, 2000.

[30] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. In *Proc. ACM SIGCOMM '94*, pages 257–268, 1994.

[31] H. Ren and K. Park. Toward a theory of differentiated services. In *Proc. IEEE/IFIP International Workshop on Quality of Service*, pages 211–220, 2000.

[32] R.J.Gibbens, S.K.Sargood, F.P.Kelly, H.Azmoodeh, R.Macfadyen, and N.Macfadyen. An apporach to service level agreement for ip networks with differentiated services. http://www.statslab.cam.ac.uk/ richard/research/papers/sla, preprint.

[33] J. W. Roberts. Engineering for quality of service. In K. Park and W. Willinger, editors, *Self-Similar Network Traffic and Performance Evaluation*. Wiley-Interscience, 2000.

[34] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Proc. ACM SIGCOMM '95*, pages 100–113, 1995.