Performance Evaluation of Optimal Aggregate-Flow Scheduling: A Simulation Study *

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

Abstract

Providing scalable QoS-sensitive services to applications with varying degrees of elasticity using aggregate-flow scheduling is a challenging problem. In [30], we advanced a theoretical framework where an optimal differentiated services provisioning problem is formulated and solved to yield solutions for optimal per-hop control. In this paper, we extend our previous work by investigating performance evaluation and implementation issues associated with the induced optimal differentiated services architecture. We design a system that realizes the optimal per-hop control coupled with end-to-end adaptive QoS control, and implement a practical enhancement by introducing a scaling function which is applied to the TOS field label value in the IP header at each router. The scaling function allows the service provider to configure the per-hop control 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 the service provider's user base. We use simulation to study the structural and dynamical properties of differentiated services as affected by optimal aggregate-flow per-hop control.

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

†Contact author: tel: (765)494-0875; fax: (765)494-0739; URL: www.cs.purdue.edu/people/renh

‡Additionally supported by NSF grants ANI-9714707, ESS-9806741, ANI-0082861 (ITR), and grants from DARPA(FTN), Purdue Research Foundation, Santa Fe Institute, 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, 27] 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 [19, 29] limits the shapability of input traffic and restricts 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 would impede scalability.

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, 21, 23]. The differentiated services framework [1, 5, 15, 23] 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 possible volatility by flow-aggregation and aggregate-flow packet scheduling. In [30], we 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 foundation for per-hop control. We studied the stability and efficiency of the overall system when driven by selfish users and service providers in the game theoretic context.

In this paper, we investigate issues related to the implementation of the optimal differentiated services architecture introduced in [30], and carry out a comprehensive performance evaluation study using simulation. First, we design a system that implements the optimal per-hop control and end-to-end

control, and propose a practical enhancement by introducing a scaling function which is applied to the TOS field label value in the IP header at each router. The scaling function allows the service provider to configure the per-hop control 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 the service provider's user base. Using simulation, we show that the scaling function enhances system efficiency in the sense that less bandwidth is needed to achieve the same QoS requirement. Second, we study both the dynamical and structural properties of the differentiated services architecture as they relate to stability and optimality. We provide comprehensive and detailed quantitative performance results and evaluations under aggregate-flow interactions. We demonstrate the QoS shaping prowess of optimal aggregate-flow per-hop control, QoS separation and efficiency at matching diverse QoS requirements, the impact—with respect to service resolution and loss of QoS shaping power—stemming from discrete and bounded label values in TOS field, the effectiveness of end-to-end adaptive label control over optimal per-hop control, dynamical properties of adaptive label control with respect to convergence and stability, and QoS distribution across multiple routers on an end-to-end path in a wide area network. Collectively, our results show that user-specified, diverse QoS requirements can be effectively facilitated over the optimal differentiated services network substrate.

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 design a system implementing optimal per-hop control and discuss its properties. Section 5 presents simulation results covering both structural and dynamical properties of the optimal differentiated services network. 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, 18, 26, 27], 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 [19, 32] limit the shapability of input traffic and restrict 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 [31], 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], we 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 [17, 25] stability and efficiency properties of a parallel link routing system are studied, where selfish users are allowed to choose to 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, yields a aggregate-flow QoS provisioning system.

In the differentiated services area, efforts have been 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, 23]. Assured Service [5, 15] and Premium (or Expedited) Service [23, 16]—two 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 [23]. The most challenging problem—how to efficiently compute service level agreements (SLA)—is not addressed. It is the latter 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 [22], simplified models of Assured Service and Premium Service are

acvanced and analyzed them with respect to their performance when compared to simulation-based evaluations. In [20], Lin *et al.* 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], a proportional differentiation model is described which seeks to achieve robust, configurable service class separation—i.e., QoS differentiation—with the support of two candidate packet schedulers. Simulation is used to study the behavioral properties. In Smart Market [21], 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 [24] provides a framework and argument for discussing the role of service differentiation through pricing, even if network resources are plentiful. In [14], Gibbens *et al.* 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 or not are well-founded, its scope does not extend to aggregate-flow scheduling which is at the heart of differentiated services.

3 Overview of Optimal Aggregate-flow Scheduling

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 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 control inside a network.

3.2 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

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

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, $\xi : [1, L] \to [1, m]$. That is, n flows—effectively L aggregated flows (or "super flows") from the router's perspective—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 QoS Shaping Properties

There are three properties of the per-hop control which are relevant from a QoS control perspective. 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 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

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

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 Optimal Aggregate-flow Per-hop Control

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 [30], we provide a semantics and formulation of an optimal QoS provisioning problem, which then is solved to find the optimal per-hop control and its properties.

The optimality criterion is based on mean square error with the normalization step

$$\hat{\eta}_i = \begin{cases} \frac{\eta_i - \eta_{\min}}{\eta_{\max} - \eta_{\min}}, & \text{if } \eta_{\max} \neq \eta_{\min}, \\ 1, & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\eta_{\min}$, $\eta_{\max}$ are the minimum and maximum values of $\{\eta_1, \eta_2, \ldots, \eta_n\}$, respectively. The minimization problem

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (\hat{\eta}_i - \hat{\omega}_i)^2. \tag{3.2}$$

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.

Given $\boldsymbol{\eta}, \boldsymbol{\lambda} \in \mathbb{R}_+^n$, it can be shown [30] that the solution to (3.2) is

$$\alpha_i = (1 - \nu) \frac{\lambda_i \hat{\eta}_i}{\sum_{j=1}^{n} \lambda_j \hat{\eta}_j} + \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \qquad i \in [1, n], \tag{3.3}$$

where $0 \leq \nu < 1$ is a parameter defining a continuous family of solutions. The solution is unique in that any solution to (3.2) must be of the form (3.3) for some $0 \leq \nu < 1$. Moreover, the optimal per-hop control given by (3.3) satisfies properties (A1), (A2), and (B).

Note that the optimal per-flow solution is a convex combination of two parts, the first term which weights each flow by the normalized 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).

Although the optimal per-flow scheduling problem is nontrivial, the main technical problem lies in solving the *aggregate-flow* scheduling problem when $m < n$. An *aggregate-flow per-hop control with parameter* (m, L) is a function

$$\Phi_{m,L} : (\boldsymbol{\eta}, \boldsymbol{\lambda}) \mapsto (\xi, \boldsymbol{\alpha}) \tag{3.4}$$

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.

It can be shown [30] that $\Phi_{m,L}$ is an optimal aggregate-flow scheduler, i.e., satisfies (3.2) if and only if ξ is a solution to

$$\min_{\xi'} \sum_{k \in [1,m]} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\eta}^k)^2 \tag{3.5}$$

where

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

and the minimum ranges over all reduction classifiers ξ'. Thus an optimal aggregate-flow classifier must be a reduction classifier, and furthermore, it must efficiently cover—in the 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], however, the clustering problem given by (3.5) 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.

4 Optimal Differentiated Services Architecture

4.1 Optimal Aggregate-flow Per-hop Control Design

In our design, the optimal per-hop control consists of a classifier and a packet GPS scheduler which is configured to have m service classes. We set $m = L$ (cf. Section 3.2) 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 $\eta \in \{1, 2, \ldots, L\}$.
- Based on a fixed time interval T_s (T_s is set at $100ms$ in the results reported in Section 5):
  - 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 the procedure in Figure 4.1.
- 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. [3]

$\underline{\Phi_{L,L}(\boldsymbol{\eta}, \boldsymbol{\lambda})}$:

1. Set $\hat{\eta}^k = \hat{\eta}_i$ for $k \in [1, L]$, if for some $i \in [1, n]$, $\eta_i = k$,
   Set $\hat{\eta}^k = 0$ for all other k (no packet carries label value k).
2. For $k \in [1, L]$, set

$$\alpha_k = (1 - \nu) \frac{\lambda^k \hat{\eta}^k}{\sum_{j=1}^{L} \lambda^j \hat{\eta}^j} + \nu \frac{\lambda^k}{\sum_{j=1}^{L} \lambda^j}.$$

Figure 4.1: Structure of reduction classifier for $m = L$; α_k is the service weight allocated to service class $k \in \{1, \ldots, L\}$.

[3] 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).

In Section 5.2, we verify the above design by confirming the QoS separation and adaptive label control properties of optimal aggregate-flow per-hop control. Further fine-tuning of its components and parameters can yield additional performance gains.

4.2 End-to-end QoS Control Design

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

$$\max_{\eta_i \in [1,L]} U_i(\lambda_i, \mathbf{x}^i, p_i). \tag{4.1}$$

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

$$\frac{d\eta_i}{dt} = (-1)^a \varepsilon \, ||\mathbf{x}^i(\boldsymbol{\eta}) - \boldsymbol{\theta}^i|| \tag{4.2}$$

is a specific instance of adaptive label control and can be shown to be asymptotically stable. Here, $a = \text{sgn}(||\mathbf{x}^i(\boldsymbol{\eta})|| - ||\boldsymbol{\theta}^i||)$ where $\text{sgn}(\cdot) = 1$ if the argument is nonnegative, and 0, otherwise.

Thus, 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 results reported in Section 5), and sends feedback control packets using UDP with label value L. (control packets are given high priority).

- Adaptive label control: after the k'th feedback is received, the sender compares measured end-to-end performance with the QoS target and sets a new η value based on (4.2).

4.3 Scaling Function

From an absolute QoS shaping perspective, the optimal relative QoS exported by routers according to the procedure in Figure 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 4.2 (left) which shows the packet loss rates exported by the optimal aggregate-flow scheduler specified in Section 4.1 with $m = 16$ service classes as link bandwidth is varied[4]. We observe that optimal aggregate-flow scheduling exports relative QoS with "equal spacing". The QoS spacing between service classes has a finite resolution $\Delta c \approx 0.06$, or 0 (degenerate case), as affected 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.

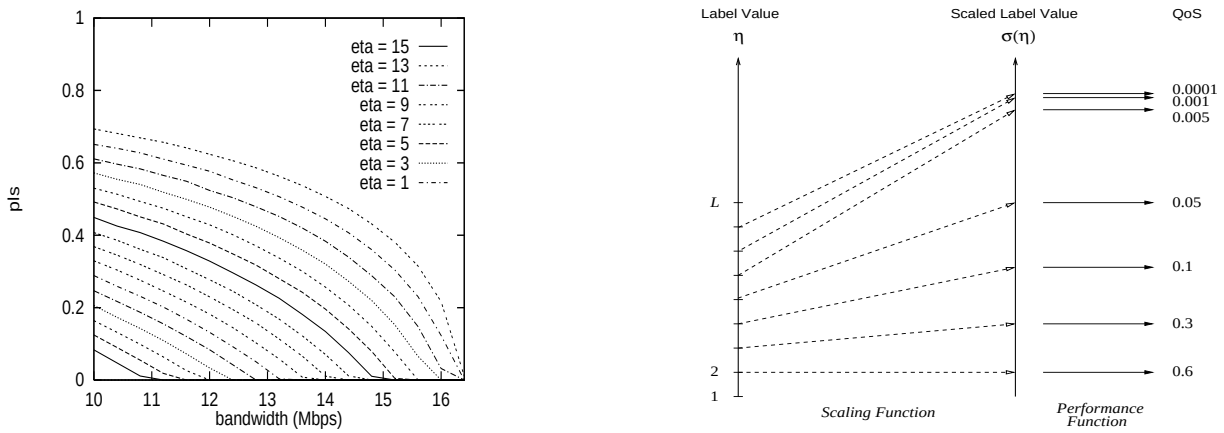

Figure 4.2: Left: "Equal spacing" QoS separation achieved by optimal aggregate-flow classifier when $L = 16$. Right: Scaling function σ affecting nonuniform stretching and contraction.

To solve the potential inefficiency and structural limitation pointed out above, we introduce a

[4]The performance results are taken from Section 5.

scaling function

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

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 the IP packets are passed to the classifier proper. This is illustrated in Figure 4.2 (right).

The modified classifier with scaling function is:

$\underline{\Phi_{L,L}(\boldsymbol{\eta}, \boldsymbol{\lambda})}$:

1. Set $\hat{\eta}^k = \widehat{\sigma(\eta_i)}$ for $k \in [1, L]$ if for some $i \in [1, n]$, $\eta_i = k$.
   Set $\hat{\eta}^k = 0$ for all other k (no packet carries label value k).

2. For $k \in [1, L]$, set

$$\alpha_k = (1 - \nu) \frac{\lambda^k \hat{\eta}^k}{\sum_{j=1}^{L} \lambda^j \hat{\eta}^j} + \nu \frac{\lambda^k}{\sum_{j=1}^{L} \lambda^j}.$$

Figure 4.3: Structure of reduction classifier with scaling function for $m = L$; α_k is the service weight allocated to service class $k \in \{1, \ldots, L\}$.

The performance gain brought by the scaling function is demonstrated and further discussed in Section 5.3.

4.4 Load Imbalance and Local QoS Responsibility

Another practical concern for aggregate-flow per-hop control is the efficiency and QoS shaping properties 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 4.4. 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 by flow i's choice of label value η_i—would impose the same 5ms (ignoring, for simplicity of exposition, link latencies) as the local QoS responsibility at each router irrespective of its load. A more desirable—and,

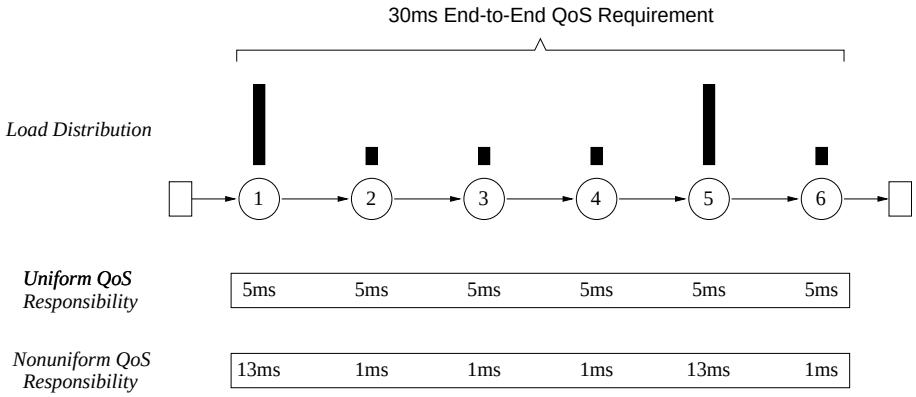

Figure 4.4: Uniform vs. nonuniform local QoS responsibility distribution to satisfy 30ms end-to-end delay requirement for a given load imbalance.

in general, efficient—allocation is one where heavily loaded routers are given less responsibility than lightly loaded ones[5]. This is illustrated in the nonuniform QoS distribution in Figure 4.4 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 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 the burden for improved QoS than heavily loaded ones. In Section 5.4, we show that both properties are satisfied by the optimal aggregate-flow scheduler.

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 controls and end-to-end controls, along with a QoS interface through which user QoS requirements can be specified. The control parameters and network configurations are configurable via a GUI. The latest version incorporates the optimal per-hop control and adaptive label control specified in Section 4.

[5]One can formally attribute this property by using congestion pricing and utilization as cost measures [3].

5.1.1 Network Configuration

We use three benchmark network topologies, a 2-switch dumbbell topology shown in Figure 5.1 (left) which isolates a bottleneck link, a 4-switch caterpillar topology shown in Figure 5.1 (middle) which is comprised of multiple bottleneck links, and an 11-switch Abilene-like topology shown in Figure 5.1 (right). The 11-switch Abilene-like topology emulates the physical topology of *Q-Bahn*—an

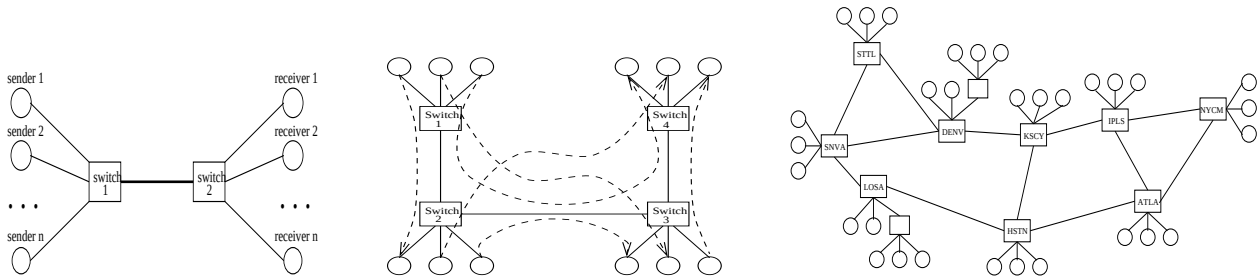

Figure 5.1: Benchmark network topology. Left: 2-switch single bottleneck link shared by n flows. Middle: 4-switch multiple bottleneck link caterpillar topology. Right: 11-switch Abilene-like topology

IP-over-SONET QoS backbone comprised of Cisco 7206 VXR routers—which is used as a QoS testbed at the Network Systems Lab at Purdue University, wherein our optimal aggregate-flow per-hop control architecture is implemented in Cisco IOS. The simulation environment allows us to carry out larger scale performance studies than is possible on the physical 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 real workloads and 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 extended by those carried out for the topology shown in Figure 5.1 (middle) and Figure 5.1 (right). The number of service classes varied from 1 to 32, and the number of user flows varied from 16 to 450. For the convenience of (3.1), the η value we used in simulation ranges from 0 to $L-1$.

5.2 Service Differentiation

In this section, we investigate the service differentiation properties of optimal aggregate-flow per-hop control with respect to its QoS separation (property (B)) and shaping (properties (A1) and (A2)) performance, the impact of the weighting parameter v (cf. Step 2 in Figure 4.1), and the impact of input traffic burstiness.

5.2.1 QoS Separation and Bandwidth

First, we show the QoS separation performance of optimal aggregate per-hop control 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). Each flow is CBR. Figure 5.2 shows

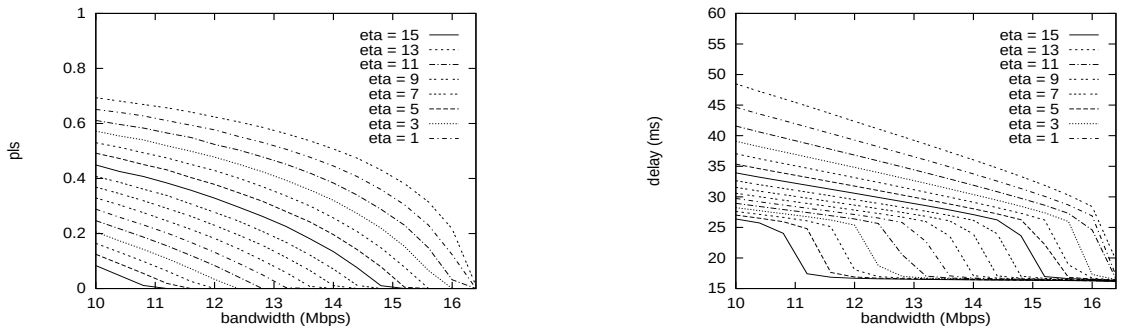

Figure 5.2: QoS separation achieved by optimal aggregate-flow classifier when $L = 16$. Left: Packet loss rate. Right: End-to-end delay.

the QoS separation achieved in the $L = 16$ service classes when executing the optimal aggregate-flow service weight assignment by the scheduler. 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 6 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 become 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 decreases as η approaches $L - 1$—while satisfying property (B). This can be explained, in part, by the fact that the waiting time W_k in service class k is its queue length divided by service rate $\alpha_k \mu$. Since the optimal aggregate-flow classifier satisfies $\frac{\alpha_L}{\lambda_L} \geq \frac{\alpha_{L-1}}{\lambda_{L-1}} \geq \cdots \geq \frac{\alpha_1}{\lambda_1}$, the shape of the end-to-end delay curve is consistent with the waiting time dependence on service rate.

5.2.2 Label Control: Properties (A1) and (A2)

Next we show how properties (A1) and (A2) are satisfied by the optimal aggregate-flow per-hop control. The number of service classes is configured at $L = 16$ (bottleneck bandwidth is fixed at 5 Mbps), and we let a single flow occupy service class 0 initially. Subsequently, this single user flow increases its label

value η_{16} from 0 up to the maximum value 15, and the resulting QoS exported by the 16 services classes is plotted in Figure 5.3 (left). We observe that property (B) remains satisfied, and more importantly, properties (A1) and (A2) are satisfied by the optimal aggregate-flow classifier. Property (A1) is implied

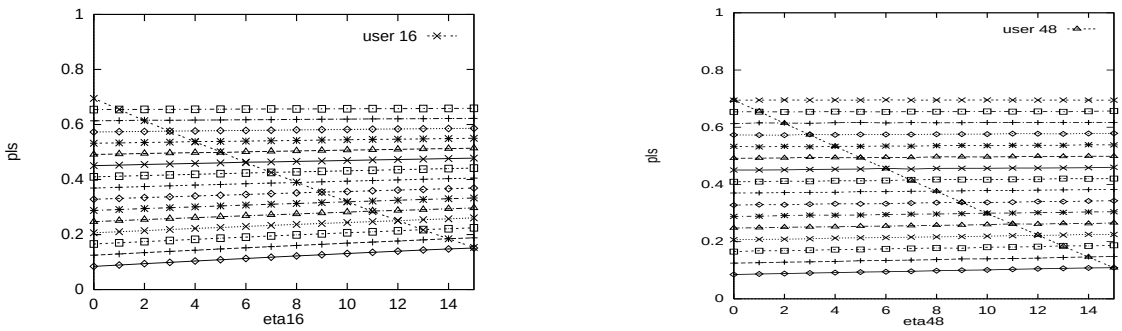

Figure 5.3: Manifestation of properties (A1) and (A2). End-to-end QoS shaping as a function of label value η_{16} of singular user flow. Left: 16 users (originally each group has one user). Right: 48 users (average group population size of 3).

by service class separation which shows that the QoS of the user who is increasing his η value is monotonically improving, and property (A2) can be confirmed by the slight positive slope of the service class performance curves as a function of η_{16}. Also, note that for $\eta_{16} \geq 1$, there are effectively only 15 service classes since class 0 becomes empty. Figure 5.3 (right) shows the corresponding performance curves when the initially service class 0 is populated by three user flows, and one of the flows increases its label value from 0 to 15 as before. Bottleneck bandwidth is set at 15 Mbps. We observe that properties (A1) and (A2) are satisfied.

5.2.3 Impact of Weighting Parameter ν

Recall that the service weight assignment, given by equation (3.3), has a weighting parameter $0 \leq \nu < 1$ 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 (the first term of (3.3)) 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 (the second term) is eliminated thus turning the classifier into a maximal differentiator. This is shown in Figure 5.4.

The left figure shows QoS separation as a function of bottleneck bandwidth for $L = 16$ when $\nu = 0.9$ which makes the system resemble a FIFO scheduler. Figure 5.4 (right) shows the corresponding plot

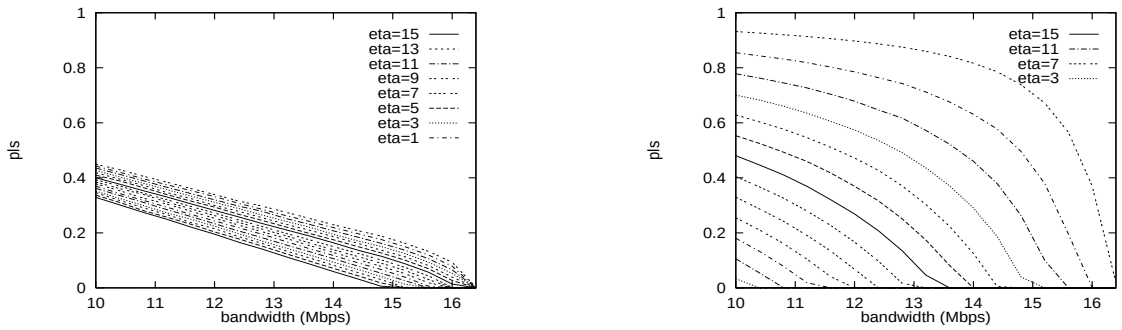

Figure 5.4: Impact of ν on QoS separation for $L = 16$. Left: QoS exported by service classes as a function of bottleneck bandwidth when $\nu = 0.9$. Right: Corresponding plots when $\nu = 0.1$.

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. 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 and their QoS profile.

5.2.4 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 and show that the qualitative behavior of the aggregate-flow QoS provisioning remains the same. Figure 5.5 shows QoS separation results across $m = 16$ aggregate-flow service classes for the same network configuration as in Figure 5.2, except that the traffic sources are VBR—in particular, Poisson— with the same average data rates.

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 a threshold effect characteristic of CBR traffic.

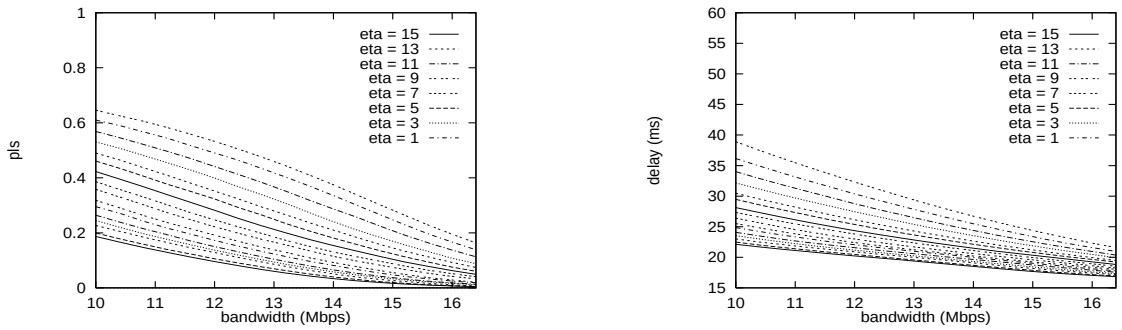

Figure 5.5: QoS separation achieved by optimal aggregate-flow classifier when $L = 16$ under VBR traffic. Left: Packet loss rate. Right: End-to-end delay.

5.3 Structural Properties of Optimal Aggregate-flow Per-hop Control

The major part of our performance evaluation study focuses on the structural and dynamical properties of differentiated services as affected by optimal aggregate-flow per-hop control. We first study the structural properties of optimal aggregate-flow per-hop control with respect to its efficiency and optimality indicated by the existence and size of $\mathcal{A}^*$ (defined below), the impact of bounded and discrete label values $\{1, 2, \ldots, L\}$, and the benefit of scaling function. Then we show the dynamical and convergence properties of adaptive label control running over the optimal aggregate-flow per-hop substrate.

5.3.1 Efficiency and Optimality

In this section, we study the structure of $\mathcal{A}^*$, where $\mathcal{A}^*$ is the set of $\boldsymbol{\eta}$ configurations where all user requirements are satisfied. First, a definition. Given a configuration $\boldsymbol{\eta}$, we will call a change in label value η_i by user i a *selfish move* if the consequent configuration gives higher utility to user i. A configuration $\boldsymbol{\eta}$ is a *Nash equilibrium* if no user can improve her utility by selfish move. If, in addition to being a Nash equilibrium, $\boldsymbol{\eta}$ belongs to $\mathcal{A}^*$, then we call $\boldsymbol{\eta}$ a *corner point*. Thus a corner point is a maximally efficient configuration where all users' QoS requirements are satisfied.

Figure 5.6 (left) shows the Nash equilibria and corner point structure of $\mathcal{A}^*$ when 25 user flows are grouped into 8 QoS requirement groups given by the packet loss rate bound profile $(0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 1.0)$. Thus the population group with bound 0.01 has the most stringent QoS requirement, and the last group corresponds to a "best-effort" user group in the worst-case sense. The Nash equilibria and corner points are found by brute-force search. For bottleneck bandwidths below 10 Mbps, $\mathcal{A}^*$ is empty but Nash equilibria nonetheless exist (see [30] for game-theoretic

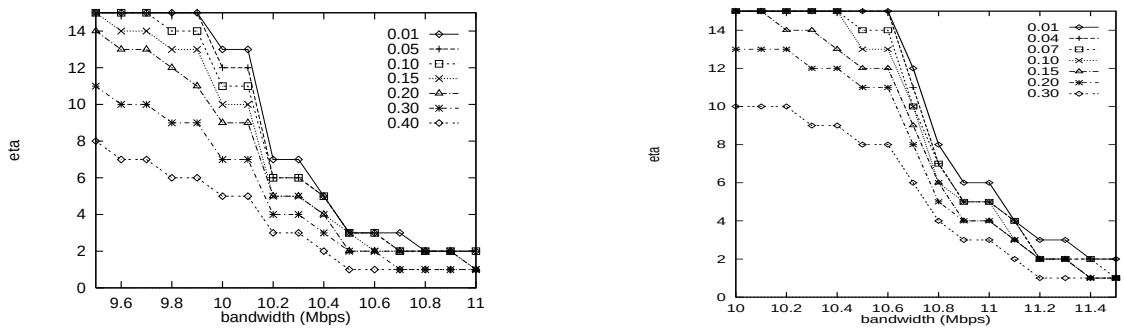

Figure 5.6: Structure of $\mathcal{A}^*$. Left: The change in Nash equilibria as we increase bottleneck bandwidth for user population with QoS requirement profile $(0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 1.0)$. At 10 Mbps, Nash equilibria become corner points of $\mathcal{A}^*$. Right: Corresponding landscape for more stringent user population QoS profile shown in the legend.

results). User flows with packet loss requirements 0.01, 0.04, and 0.07 are not satisfied (their actual packet loss rates received exceed their respective bounds), and their η values have saturated at the maximum value 15. At 10 Mbps bottleneck bandwidth, however, $\mathcal{A}^* \neq \emptyset$, and the distribution of seven label values correspond to the corner configuration. We observe that the range of individual label values is large (from 5 to 13 for a distance of 8), and then subsequently shrinks as bandwidth is increased. The optimal aggregate-flow classifier, is also *efficient* in the sense that if an aggregate-flow resource assignment exists that satisfies all user requirements, then it is apt to find it. Figure 5.6 (right) shows the plots for a more stringent user population profile $(0.01, 0.04, 0.07, 0.1, 0.15, 0.2, 0.3, 1.0)$. Nash equilibria turn into corner points at a higher bottleneck bandwidth 10.7 Mbps, but the reduction in the distance between label values at corner point configurations persists.

5.3.2 Impact of Bounded and Discrete Label Values (L)

In this section, we show the performance impact of bounded and discrete label values given by L. If $L = 1$, then we know that the optimal aggregate-flow control degenerates to FIFO scheduling. Thus small L diminishes the QoS resolution achievable by the classifier and increasing L amplifies it. Figure 5.7 shows this dependence. For $n = 32$ user flows, we show the QoS exported by L service classes as a function of bandwidth as L is increased $L = 1,\ 4,\ 8,\ 32$. Thus at $L = 32$, the system degenerates to per-flow control. The weighting parameter ν of the optimal classifier (cf. equation (3.3)) is set to 0.5.

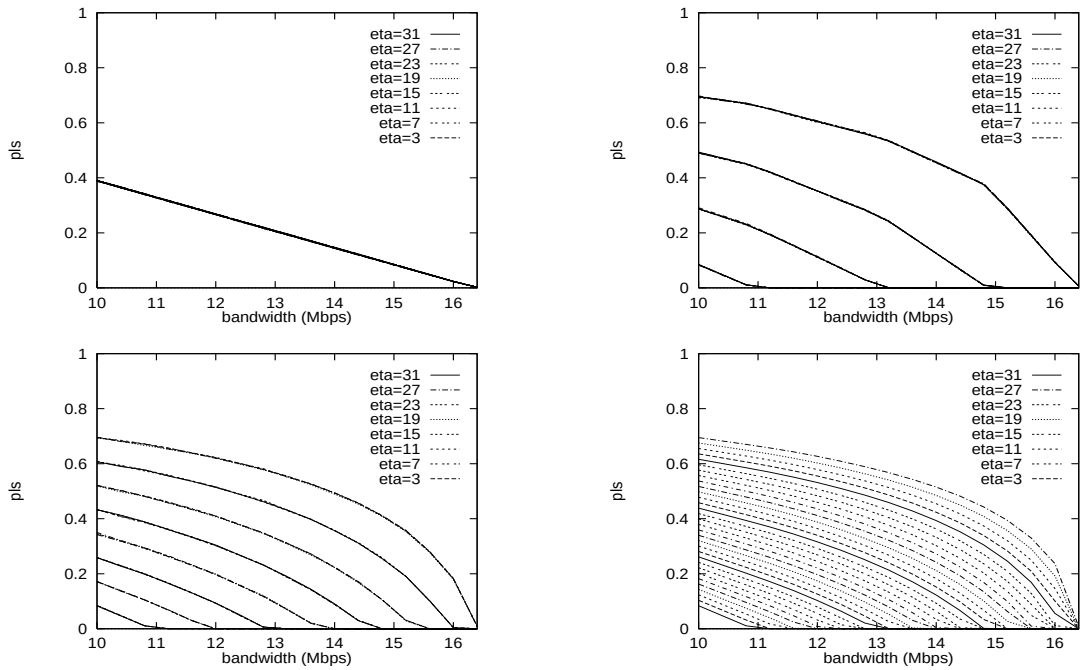

Figure 5.7: Impact of bounded label set size L on QoS exported by the service classes as a function of bottleneck bandwidth for L=1, 4, 8, 32.

Figure 5.7 (top row, left plot) shows QoS shaping when $L = 1$, i.e., FIFO with no QoS separation. As L is increased, however, we observe that increased L endows increased QoS resolution with finer levels of QoS spacing. In fact, for $L = 2, 4, 8, 16, 32$ (i.e., excluding $L = 1$), we observe that the performance curves are strict supersets of each other as indexed by L. This can be shown to follow from the form of label value normalization $\hat{\eta}$ defined by the map (3.1) (see Section 4). Since for all finite $L > 0$, $\eta \in \{1, 2, \ldots, L\}$ is mapped by (3.1) 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* $[0, 1]$ by the action of (3.1), and $\mathbb{Z}_+$ is mapped to a dense subset of $[0, 1]$. Thus L translates to "QoS resolution" in a straightforward fashion.

Figure 5.8 shows the joint influence of L and ν on QoS shaping. The left figure shows the QoS exported by L service classes as a function of L (actually its logarithm) for $L = 1,\ 2,\ 4,\ 8,\ 16,\ 32$ when the weighting parameter is set to $\nu = 0.5$. As a function of L, this generates a *binary QoS tree* whose leaves at level $\log L$ show the range and values of QoS covered by the L service classes. Figure 5.8 (right) shows the corresponding plot when $\nu = 0.1$. We observe that the width of the covering has significantly widened, however, at the cost of a more coarse-granular QoS spacing. Thus,

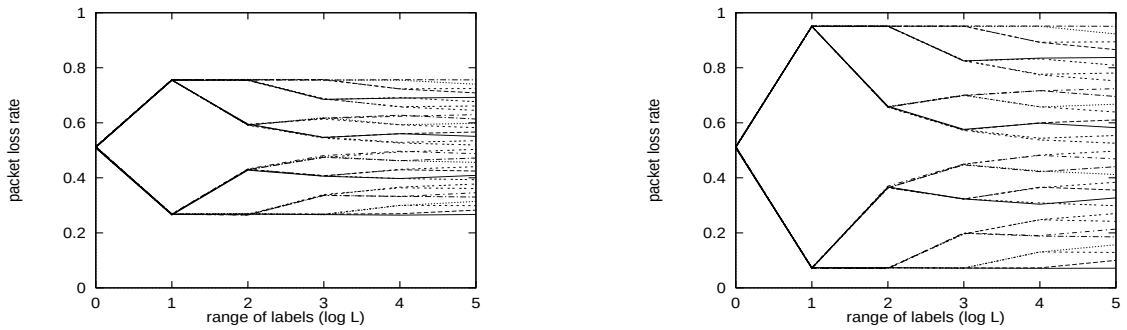

Figure 5.8: The combined impact of L and ν on QoS shaping. Left: QoS exported in L service classes as a function of L for $\nu = 0.5$. Right: Corresponding plot for $\nu = 0.1$.

as indicated earlier, the L 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 $L = 2^\ell$ ($\ell \geq 1$) the subset relation $\mathcal{C}_2 \subset \mathcal{C}_4 \subset \cdots \subset \mathcal{C}_L \subset \cdots$ of the coverings $\mathcal{C}_L \subset [0, 1]$ as a function of L—which is implied by the form of the normalization (3.1)—can be observed in Figure 5.8.

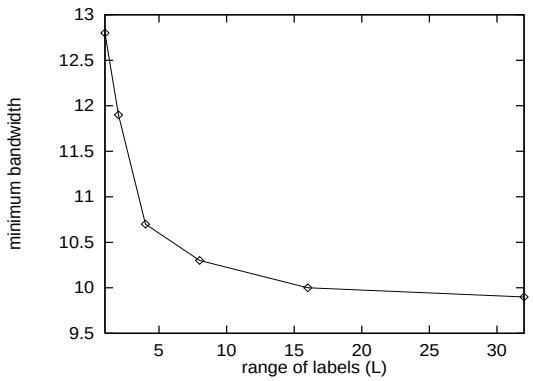

Figure 5.9: Impact of L on existence of $\mathcal{A}^*$: minimum bottleneck bandwidth required to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L.

The size of the label set L also impacts the system efficiency, the existence or nonemptiness of $\mathcal{A}^*$ which is shown in Figure 5.9. 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 were 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.

5.3.3 The Role of Scaling Function

In this section, we study the performance impact of the scaling function. Figure 5.10 shows how the scaling function can affect the QoS differentiation among service classes. Comparing it with Figure 5.2, we see that the scaling function can achieve non-uniform QoS differentiation.

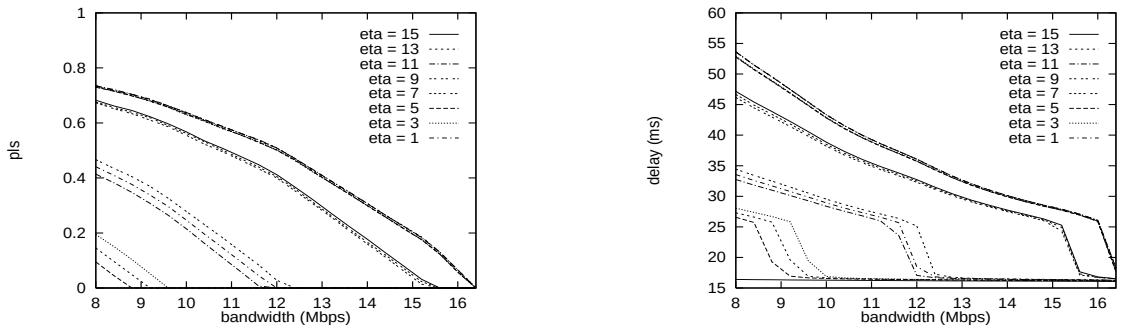

Figure 5.10: QoS differentiation achieved by optimal aggregate-flow classifier with scaling function. $\sigma(\eta)$ for $\eta \in [0, 15]$: 1.0, 1.1, 1.2, 10, 11, 12, 100, 110, 120, 500, 550, 600, 1000, 1100, 1200, 2000. Left: Packet loss rate. Right: End-to-end delay.

Next we show that the scaling function can improve system efficiency. As in Section 5.3.2, system efficiency is represented by the minimum bottleneck bandwidth needed to achieve $\mathcal{A}^* \neq \emptyset$. Configure users into 8 groups with different QoS requirements. Figure 5.11 shows the minimum bottleneck bandwidth needed to achieve all users' QoS targets as a function of L. The top curve is obtained without the scaling function. This is essentially the same curve described in Figure 5.9. The bottom curve is obtained with an "optimal" scaling function, i.e., for a given L, if we are allowed to choose σ, what is the minimal bandwidth needed.

We observe the following: First, with a scaling function, the system achieves its maximum efficiency when $L \geq 8$; i.e., for the bottom curve, the minimum bottleneck bandwidth required to satisfy all the users' QoS targets does not further decrease when $L > 8$. Second, 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 the transformation (3.1) when $L = 2$. Third, the increase in efficiency, which is represented by the difference of the minimal bottleneck bandwidths in the two curves, is large when $L = 4, 8$. However, it becomes smaller when $L > 8$.

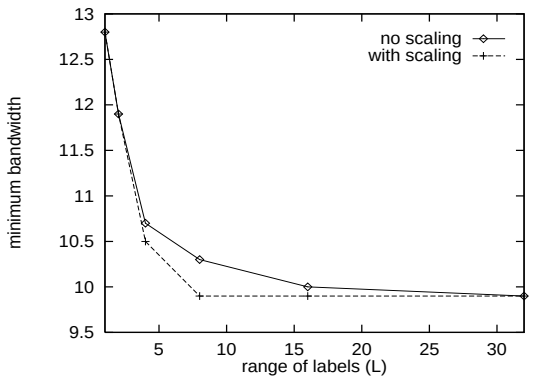

Figure 5.11: Impact of scaling function on system efficiency: minimum bottleneck bandwidth required to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L.

5.4 Dynamics and Convergence

5.4.1 Time Evolution

In this section, we show that end-to-end adaptive label control—in conjunction with the optimal aggregate-flow per-hop control network substrate—leads to stable time evolutions. The QoS exported by the service classes and the end-to-end QoS received by individual user flows is commensurately stable.

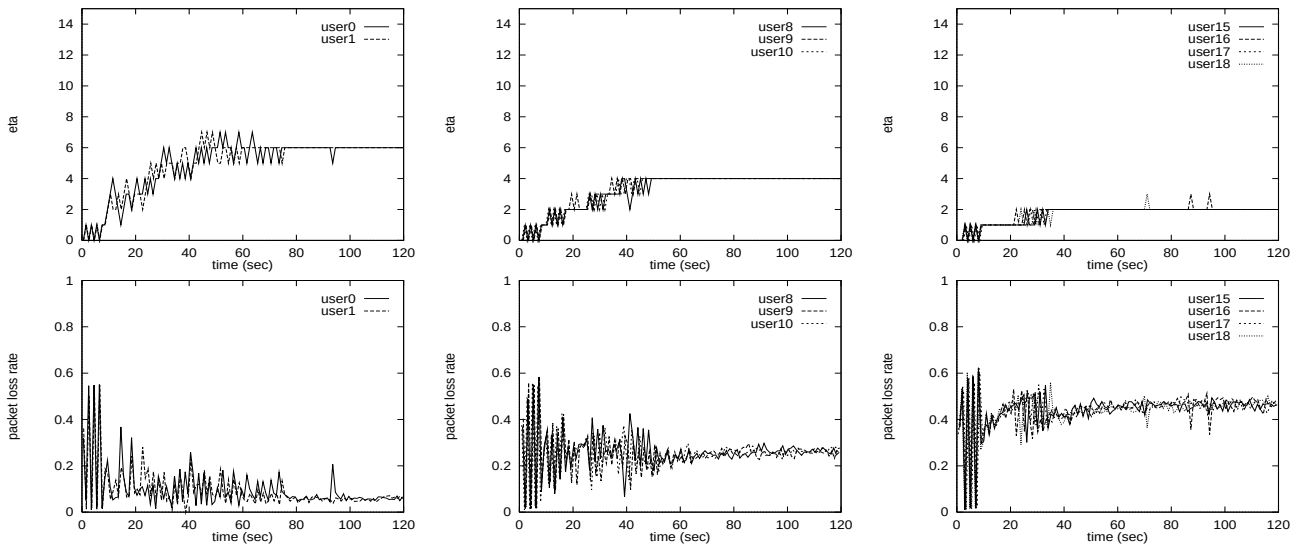

Figure 5.12: Time evolution of adaptive label control and end-to-end QoS. Top row: Evolution of label values shown for three user groups with common QoS requirements (0.1, 0.3, and 0.5). Bottom row: Corresponding trace of measured end-to-end QoS for user flows belonging to the three QoS groups.

Figure 5.12 (top row) shows the label value dynamics of adaptive label control for an $L = 16$ system with $n = 25$ users grouped into 8 groups with common packet loss rate requirements 0.1 (users 0 and 1), 0.15 (users 2–4), 0.2 (users 5–7), 0.3 (users 8–10), 0.4 (users 11–14), 0.5 (users 15–18), 0.6 (users 19–21), and best-effort (users 22–24). We show the η value traces for three of the eight user groups. First, we observe that in spite of each individual user flow executing its end-to-end adaptive label control *independently* of all other user flows—including those in the same group to which each flow is oblivious—user flows with the same QoS requirement aggregate to the same service class. Second, the η values converge to equilibrium values after a transient period. The length of the transient period is a function of feedback control parameters that are not specific to optimal aggregate-flow scheduling. Third, the brief label value perturbations in the groups comprised of users 0, 1, and users 15–18 show that QoS is not guaranteed. Figure 5.12 (bottom row) shows the corresponding measured packet loss rate traces observed at the receiver. We observe that the rendered end-to-end QoS is stable, with the label perturbations reflecting the corresponding QoS perturbations in the packet loss trace.

5.4.2 Stringency of QoS Requirement

Figure 5.13 shows the impact of increased stringency in the QoS requirement profile on QoS provisioning performance. For the first QoS requirement profile $(0.1, 0.15, 0, 2, 0.3, 0.4, 0.5, 0.6)$, when the bottleneck bandwidth is less than 8.3 Mbps, $\mathcal{A}^*$ is empty—i.e., there does not exist an aggregate-flow ($L = 16$) service weight assignment and configuration that is able to satisfy the QoS requirements of all user flows. This shows up as "clustering" of the stringent QoS flows which stems from saturation of label values at the maximum L. At bottleneck bandwidth 8.3 Mbps or higher, however, $\mathcal{A}^* \neq \emptyset$ and adaptive label control finds a label assignment (and corresponding service weight assignment at routers controlled by the classifier) where all user flows are satisfied. A similar behavior is observed for the three successively more stringent QoS profiles whose initial clustering is more pronounced, and the QoS exported lies in a narrower QoS range—as required to satisfy the more stringent QoS requirements—with fine-granular QoS spacing (note that the ordinate has been scaled accordingly).

5.4.3 Time Evolution in Many-switch Topology

We have performance results for the many-switch system (shown in Figure 5.1) when adaptive label control is used. The many-switch results are analogous to the bottleneck topology results (Figure 5.12), and are omitted due to space constraints.

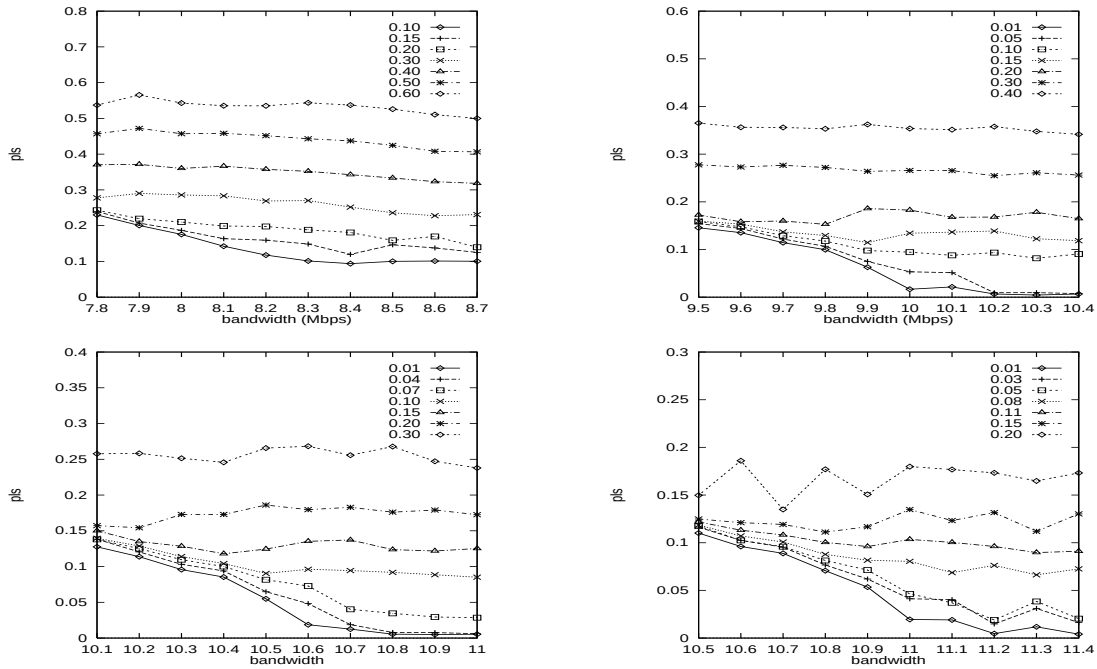

Figure 5.13: End-to-end QoS achieved under adaptive label control as a function of bottleneck bandwidth for successively more stringent QoS requirement profiles (shown in the legends).

5.4.4 Load Imbalance

In Section 4.4, we indicated that an efficient way to provide QoS across multiple switches on an end-to-end path is to give more responsibility to the lightly loaded switches than to the heavily loaded ones. In this section, we demonstrate that the optimal aggregate-flow per-hop control exports such properties both statically and dynamically.

The simulation set-up is as follows: let user flows traverse multiple hops where they share the bandwidth with cross traffic. The cross traffic sources are configured such that each class of the switches receives some amount of cross traffic. Therefore, by varying the volumes of cross traffic injected into all the classes, we can effectively control the load of the switches.

We first consider the scenario when all the switches are evenly loaded. Figure 5.14 shows the QoS distribution pattern of the system. The left figure depicts the topology and the load on each switch. The middle figure shows the QoS achieved by each class at each switch. Since all the switches have similar load, the QoS experienced by the same class at different switches are cpmparable. 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

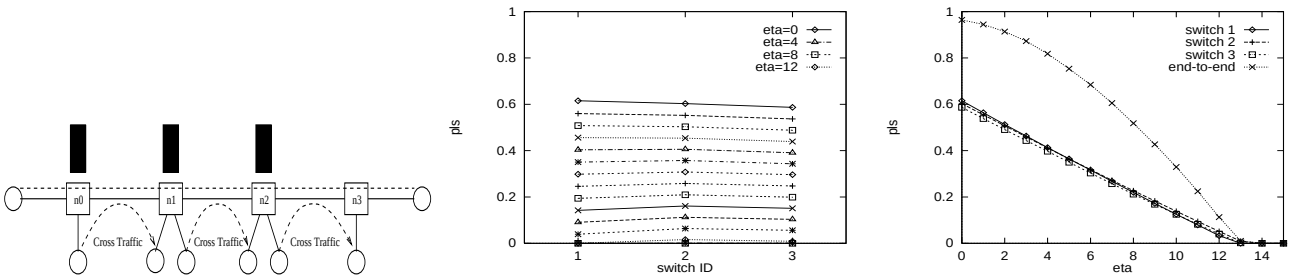

Figure 5.14: QoS distribution on multi-hop path with three medium-loaded switches. Left: switch loads. Middle: Static QoS distribution. Right: Dynamical QoS distribution.

same responsibilities to improve the QoS of the user.

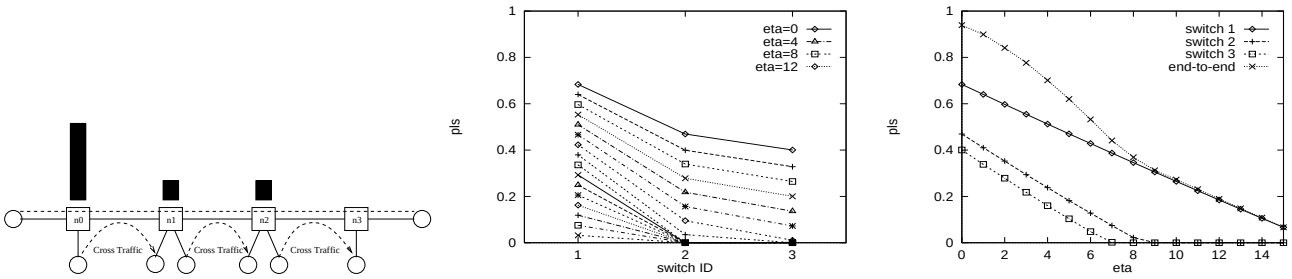

Figure 5.15: QoS distribution on multi-hop path with one heavy-loaded switch and two light-loaded switches. Left: switch loads. Middle: static QoS distribution. Right: dynamical QoS distribution.

Next we study the QoS distribution among switches when they are under different load conditions as depicted in Figure 5.15 (left). Figure 5.15 (middle) shows the static QoS distribution among the switches by plotting the QoS of all the classes at each switch. As stated in Section 4.4, the classes in the lightly-loaded switches experience better QoS than the corresponding classes in the heavily-loaded switch. Figure 5.15 (right) shows the more subtle feature, the dynamic QoS distribution among switches. It depicts the change of the end-to-end QoS and the QoS achieved at each switch when one user changes its η value from 0 to 15. In Section 4.4, we stated that when one user increases η value to improve its QoS, an efficient way is to put more responsibility on the lightly-loaded switch. Figure 5.15 (right) verifies that our optimal aggregate-flow per-hop control exhibits this property. As η increases, we observe the packet loss rate at the lighted-loaded switch decreases faster than it does at the heavily-loaded switch.

6 Conclusion

In this paper, we advance the previous work on the optimal aggregate-flow scheduling. We have designed a system that implements the optimal aggregate-flow per-hop control and end-to-end QoS control, and given a comprehensive simulation-based performance evaluation of the system. We have shown quantitative performance evaluations of both structural and dynamical properties, thus extending, in addition to complementing, the previous theoretical results [30]. The performance results, overall, show that user-specified services can be efficiently and effectively achieved over the optimal aggregate-flow per-hop control network substrate when coupled with adaptive label control.

In collaborating with Cisco Systems, we have implemented the optimal aggregate-flow per-hop control in Cisco IOS. Current work is directed at comprehensive performance evaluation and benchmarking over the IP-over-SONET QoS testbed Q-Bahn which is comprised of Cisco 7206 VXR routers and follows Abilene's physical connectivity.

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] 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. In *Phil Trans Royal Soc Lond A*, pages 2165–2182, 2000.

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

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

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

[26] 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.

[27] 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.

[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] 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.

[31] 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.

[32] 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.