AGGREGATE-FLOW SCHEDULING: THEORY AND PRACTICE

A Thesis

Submitted to the Faculty

of

Purdue University

by

Huan Ren

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2002

To Xin and Eric

ACKNOWLEDGMENTS

I would like to show my sincere gratitude to my advisor Professor Kihong Park for his guidance during my Ph.D. study. The hundreds of hours that we spent together in the past four years are still vivid in my memory. His devotion to science, enthusiasm for perfection, and creative thinking provide an example of researcher that I shall follow in my life. I thank him not only for giving me copious amounts of insightful criticism on my research but also for his far-reaching advices on my professionalism and future career.

I would like to thank Professors Sonia Fahmy, Ness Shroff, and Wojciech Szpankowski for being on my advisory committee. The computer networking seminar course I took from Professor Fahmy helped me start working on the topic of this dissertation. Prof. Shroff is a true mentor to his students; I was fortunate to get many wise advices from him on various subjects throughout my Ph.D. study. Prof. Szpankowski always encourages students to explore unknown world, and I benefited greatly from the algorithm analysis course he taught. I also thank Professor Steven Low at California Institute of Technology for serving on my final examining committee. He made a long trip to attend my final defense and provided insightful comments on this dissertation. I thank Dr. Gorman for all his help in the department.

Special thanks should be given to my wife, Xin Liu, for her love, support, encouragement, and for being an intelligent colleague as well. The past four and a half years that we studied together at Purdue are the happiest and most memorable time in my life. Both of us get our Ph.D. degrees from Purdue with Eric Rucheng Ren as our another accomplishment. It is her that makes this long journey of pursuing Ph.D. so enjoyable.

Finally, I would like to thank my parents for their selfless love and support. I also thank my brother Kun for sharing my experiences, his sense of humor, and being my soul mate.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Ren, Huan. Ph.D., Purdue University, December, 2002. Aggregate-flow Scheduling: Theory and Practice. Major Professor: Kihong Park.

This dissertation studies providing Quality of Service (QoS) to individual flows using aggregate-flow scheduling. Our work is carried out on both theoretical and practical sides.

We present a theoretical framework for reasoning about scalable QoS provisioning using aggregate-flow scheduling, constrained to be implementable in IP networks. Our control framework—Scalar QoS Control—generalizes per-hop and edge control achievable by setting a scalar value in packet header, e.g., the Type of Service (TOS) field of IP. We study optimal aggregate-flow scheduling problem under the framework and the properties the optimal solution exhibits which facilitate end-to-end QoS via the joint action of aggregate-flow scheduling per-hop and per-flow provisioning at the edge.

We design an optimal aggregate-flow per-hop control algorithm that achieves the induced optimal aggregate-flow scheduling solution and implement the algorithm in Cisco routers. We conduct a comprehensive performance evaluation by both simulations and experiments over Q-bahn testbed comprised of Cisco routers running the implemented optimal aggregate-flow per-hop control. The benchmarking results confirm our theoretical framework and analysis, and reveal further quantitative features of both structural and dynamical properties of the system. Our results, collectively, show that user-specified services can be efficiently and effectively achieved over networks with optimal aggregate-flow per-hop control substrate when coupled with either open-loop or closed-loop (adaptive label control) edge control.

We generalize our optimal aggregate-flow per-hop control analysis by considering stochastic input and study optimal aggregate-flow scheduling problem in general multi-class queueing systems. We introduce a stochastic framework of optimal aggregate-flow scheduling, which extends the optimal per-flow scheduling framework pioneered by Coffman and Mitrani. We show that optimal aggregate-flow scheduling in multi-class $G/G/1$ systems with work-conserving, non-preemptive and non-anticipative scheduling disciplines—for which Kleinrock's conservation law holds—is NP-hard. This stands in contrast with the quadratic time complexity of optimal per-flow scheduling and cubic time complexity of optimal aggregate-flow scheduling in static input environments subject to relative service differentiation. We show that computational hardness results from the combination of optimal aggregation and Kleinrock's conservation law.

1 INTRODUCTION

1.1 Motivation

Architecting networks capable of providing scalable, efficient, and fair services to users with diverse Quality of Service (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 [15, 16, 59, 60] rely on overprovisioning coupled with traffic shaping/policing to preserve well-behavedness properties across switches that implement a form of Generalized Processor Sharing (GPS) packet scheduling [17]. Scale-invariant burstiness associated with self-similar network traffic [46, 64] 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 achievable degree of utilization 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 an 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 [8, 12, 18, 50, 55]. The differentiated services framework [3, 12, 37, 55] 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 service degradation by flow-aggregation and aggregate-flow packet scheduling.

1.2 Key Issues

A number of works have studied the behavioral characteristics of specific instances of differentiated services networks. In previous work [8, 9, 63], the authors introduced aggregate-flow per-hop control mechanisms motivated by game theoretic considerations—a router performs class-based label switching which emulates user optimal service class selection with respect to selfish users—without considering the space of all aggregate-flow per-hop controls which is carried out in this dissertation. In [52], simplified models of Assured Service [37] and Premium (or Expedited) Service [38] are presented and analyzed with respect to their performance when compared with simulations. In [24], an adaptive 1-bit marking scheme is described, and the resulting bandwidth sharing behavior is demonstrated via simulations when the priority level is controlled end-to-end. In [18], the authors describe the 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. Other related works include [8, 12, 50, 56].

In spite of these efforts, a comprehensive understanding of the power and limitation of aggregate-flow QoS provisioning is still in its infancy. Little is known about how to select "good" aggregate-flow per-hop controls—including optimal ones—and per-flow end-to-end (or edge) controls, and what criteria to apply when designing these components. Following the divide-and-conquer approach to network design, we would like to reduce the scalable QoS provisioning problem to subproblems and solve

them individually without worrying about the details of other subsystems except through well-defined interfaces and "black box" function definitions. Although the same approach is undertaken in this work, we find that there are intimate relationships between the selection of per-hop and end-to-end controls. Thus the key problem of our study is the formulation and solution of optimal aggregate-flow scheduling for scalable QoS provisioning with regard to individual user requirements and performance.

1.3 Theoretical Contributions

Our theoretical study has three parts. First, we give a general framework of scalable QoS provisioning using aggregate-flow scheduling where packet labels can be set from a finite label set and routers provide differentiated treatment of packets based on the labels enscribed. We define the meaning of optimal per-hop control within this context and find the optimal solution for aggregate-flow control. We show that the optimal per-hop control satisfies certain properties—denoted (A1), (A2), and (B), and defined in Section 2.3—which relate to how label values impact the service a flow receives at a router. We augment the general result by presenting optimal solutions when restricting the packet scheduling disciplines to variants of GPS, and the consequences on the core properties.

Second, we expand the framework by introducing selfish users who can influence QoS provisioning behavior by regulating the label values assigned to their traffic streams. Based on the properties exported by the network control—(A1), (A2), and (B)—we show how a population of selfish users with diverse QoS requirements setting their packet labels can arrive at a global allocation of resources that is *stable* (Nash equilibrium) and *efficient* (system optimal). We show that even in situations when network resources are scarce such that no resource allocation—aggregate-flow differentiation, per-flow reservation, or otherwise—can satisfy all users' QoS requirements, the system is stable and reaches a Nash equilibrium. We show that the optimal per-

hop control is also "optimal" in the noncooperative game context in the sense that when network resources are configurable such that all users' QoS requirements can be satisfied, then there exists a Nash equilibrium that is system optimal.

Third, we generalize our optimal aggregate-flow per-hop control analysis by considering the impact of stochasticity of the input: arrival processes with general interarrival and service times. We study the stochastic modeling and optimization of aggregate-flow scheduling in general multi-class queueing systems. We introduce a stochastic framework of optimal aggregate-flow scheduling, which extends the optimal per-flow scheduling framework pioneered by Coffman and Mitrani [19, 30, 23, 22, 76]. We show that optimum scheduling in multi-class $G/G/1$ systems with work conserving, non-preemptive and non-anticipative scheduling disciplines—for which Kleinrock's conservation law holds—is NP-hard. This stands in contrast with the quadratic time complexity of optimal per-flow scheduling, and cubic time complexity in static input environments subject to relative service differentiation. We show that computational hardness results from the combination of optimal aggregation and Kleinrock's conservation law.

The chapters related to theoretical analysis are organized as follows: In Chapter 2 we present a general QoS provisioning architecture using aggregate-flow scheduling. Chapter 3 studies optimal aggregate-flow per-hop control, and system stability and efficiency under a non-cooperative game context. In Chapter 6, we discuss the stochastic framework of optimal aggregate-flow scheduling.

1.4 Implementation

In addition to the theoretical analysis, we design an efficient algorithm to achieve optimal aggregate-flow per-hop control and implement the algorithm in Cisco routers. We carry out a comprehensive performance evaluation study of QoS provisioning using optimal aggregate-flow per-hop control by both simulation and benchmarking over a real network testbed.

In Chapter 4, we investigate implementation issues of QoS provisioning using the optimal aggregate-flow per-hop control derived in Chapter 3, and carry out a performance evaluation study by ns simulation. 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 requirements. We use simulation to study both the dynamical and structural properties of aggregate-flow QoS provisioning as they relate to stability and optimality. We provide comprehensive and detailed quantitative performance results and evaluations under aggregate-flow interactions, thus extending, in addition to complementing, our theoretical results. 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, the dynamical properties of adaptive label control with respect to convergence and stability, and the QoS distribution across multiple routers on an end-to-end path in a wide area network.

In Chapter 5, we describe the design and implementation of optimal aggregate-flow per-hop control in Cisco routers. The system building experience shows that the optimal per-hop control scheme proposed is practical and implementable. The overhead brought by optimal per-hop control is small. We also conducted benchmarking over the Q-Bahn testbed which is comprised of Cisco 7206 VXR routers running the optimal per-hop control. Our benchmarking output confirms the previous results

from theoretical analysis and simulation, and demonstrates the scalability of the QoS provisioning architecture.

Collectively, our results show that optimal aggregate-flow per-hop control is implementable, and user-specified and diverse QoS requirements can be effectively facilitated over the network with optimal aggregate-flow scheduling substrate.

1.5 Related Work

Early approaches to QoS provisioning use resource reservation and admission control to provision guaranteed services—deterministic or statistical—with the help of leaky bucket regulators. Although research abounds [14, 15, 16, 20, 21, 31, 45, 54, 59, 60], 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), also known as weighted fair queuing [17, 59]. Real-time constraints of multimedia traffic and the scale-invariant burstiness associated with self-similar network traffic [46, 61, 67, 85] 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 [64]—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 [73], and second-order properties—even under bufferless queuing—adversely affect jitter and related second-order performance measures. Thus QoS and utilization stand in a trade-off relationship with each other [61, 62] 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 [8, 9, 63], the authors introduced aggregate-flow per-hop packet scheduling mecha-

nisms 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 [51], the aggregate-flow scheduling scheme is modeled as a multi-class priority queue, where users are given the freedom to choose the priorities of their traffic, but are charged accordingly. In [42, 57], 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 to, 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 an aggregate-flow QoS provisioning system. Several other papers have also addressed the issue of multi-class QoS provisioning in high-speed networks [13, 49, 66, 74, 77]. Some of the works employ a cooperative framework or place significant computing responsibilities on the part of the user [49, 74], some investigate the effect of pricing incentives [13], and others represent flow/congestion control and routing models that only partially address the quality of service problem [77].

In the differentiated services area, 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 [12, 55], with primary emphasis on scalability. Assured Service [12, 37] and Expedited (or Premium) Service [38, 55]—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 the 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) or 3-state marking and using RIO gateways [12], or through leaky bucket traffic shaping with routers implementing priority queuing [55]. The most challenging problem—how to efficiently

compute service level agreements (SLA)—is not addressed within Assured Service and Expedited Service. 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 dissertation.

Several works have addressed the performance evaluation side of differentiated services, both quantitatively and qualitatively. In [52], 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 [47], the authors investigated enhanced mechanisms that improve the throughput and fairness properties exhibited by Assured Service. Feng *et al.* [24] describe an adaptive 1-bit marking scheme and demonstrate the resulting bandwidth sharing behavior using simulations when the priority level is controlled end-to-end. In [18], 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.

Some other work has been carried out in formulating resource allocation problems spanning a number of different domains in the context of microeconomics and game theory [13, 25, 26, 36, 41, 43, 44, 49, 57, 66, 74, 77, 81, 83]. In Smart Market [50], 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 [56] provides a framework and argument for discussing the role of service differentiation through pricing, even if network resources are plentiful. In [34], 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 restricted to 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.

Prior to our stochastic optimal aggregate-flow scheduling framework, optimal per-flow scheduling with stochastic input has been intensively studied, with focus on characterizing the per-flow performance space satisfying strong conservation laws [76]—a

weaker form being Kleinrock's conservation law [39, 40, 75]—under different assumptions on the input and scheduler space. Several other aspects of optimal scheduling, in contexts relevant to network control, have also been investigated. These include computational complexity of optimal control, optimal clustering under minimum mean-square error (MMSE) criterion, and job grouping. Section 6.2 provides a more detailed review on previous work related to stochastic optimization of aggregate-flow scheduling.

2 NETWORK ARCHITECTURE

In this chapter, we present a scalable QoS provisioning architecture using aggregate-flow scheduling. The architecture, Scalar QoS Control, generalizes per-hop and edge control achievable by setting a scalar value in packet headers, e.g., the TOS field of IP. Our control framework incorporates assumptions, albeit weak, about selfish user behavior and service provider behavior. This is necessitated by the essential role they play in influencing end-to-end QoS, without which an effective evaluation of aggregate-flow QoS provisioning remains incomplete.

2.1 Overall System Structure

The network system is comprised of four principal components—*per-hop control*, *edge control*, *user control*, and *service provider control*—where the first two make up the network system proper, and the latter two are incorporated to evaluate the "goodness" of the first two components. Figure 2.1 depicts the overall system structure. A user's traffic flow, upon entering the network, is assigned a label from a set of L values, e.g., enscribed in the TOS field of IPv4. The routers provide differentiated treatment of packets based on their enscribed labels, and end-to-end QoS is determined by the treatment of a user's flow on all hops along a given path. The label values are set at the edge on a per-flow basis—either once-and-for-all (open-loop), or dynamically as a function of network state (closed-loop)—facilitating end-to-end control as part of edge control. A second component of edge control is *access control* which prevents users from arbitrarily assigning labels to their packet flows without consequences. Access control may be achieved by policing, traffic shaping, and pricing. We assume that the network (in general, service provider) exports a cost to each user which increases with service quality, or equivalently, with the resources received.

The system is completed by incorporating selfish users who can regulate the label values on their packet streams to satisfy their QoS requirements at least cost, and a selfish service provider who sets prices—which determine user cost—to maximize profit.

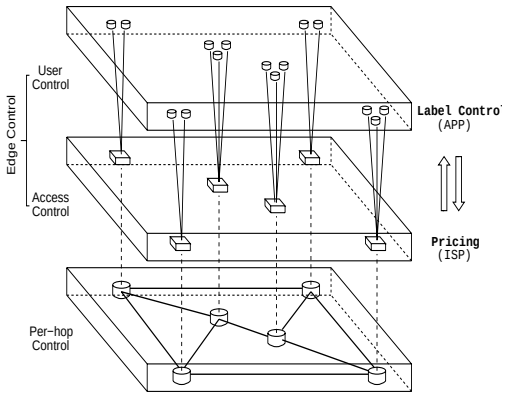

Figure 2.1. Overall QoS provisioning architecture. Network exports per-hop and edge control, user exercises scalar QoS control (η-control), and service provider exports QoS cost to user.

The job of the network system proper—per-hop control and edge-control—is to provide sufficient and efficient network mechanisms such that for a set of users or traffic flows with diverse QoS requirements, by suitable setting of the packet labels, user-specified services in the form of *target end-to-end QoS* can be provided. The setting of the label value, whether it is done by access control on behalf of a user or by a user directly, should be powerful enough so that the users' QoS requirements can be satisfied without necessitating the engagement of other traffic controls to the extent possible[1]. The network control substrate should also promote stability in a noncooperative network habited by selfish users and service providers, and facilitate efficient allocation of network resources as an outcome of selfish interactions.

[1]If an end-to-end delay of 30ms is desired but the route assigned has a propagation latency of 50ms, then clearly no amount of class-based label switching can achieve the target QoS.

2.2 Basic Definitions

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). In the following, we will assume λ_i is given and fixed ("fixed bandwidth demand"). The case when λ_i is variable ("variable bandwidth demand") is considered separately. 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 (e.g., as measured by some second-order statistic), 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 enscribed with a scalar

$$\eta_i \in \{1, 2, \ldots, L\}$$

taking on L distinct values. Unless otherwise specified, we will use $[a, b]$, for $a \leq b$, to denote the set of integers between a and b. 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—as conveyed by η_i—is lost as soon as a packet enters the network. Thus by the many-to-one mapping implied by $n > L$, *aggregate-flow QoS control* is imposed on per-hop behavior and executed per-hop at routers on an end-to-end path. In our implementation design (Chapter 4, 5) we use a number of bits in the DS field of IPv4 (and IPv6) to carry the η value, i.e., Diff-Serv Codepoint (DSCP).

2.3 Per-hop Control

2.3.1 Per-hop Control Components

Per-hop control consists of a *classifier* and a *packet scheduler*. We assume a GPS 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. It is not necessary to have GPS as the underlying packet scheduling discipline—e.g., priority queues, multiple copies of RED with different thresholds are alternatives—but we will show that GPS has certain desirable properties when

considering the problem of selecting an optimal aggregate-flow per-hop control for differentiated services. An important component is the classifier which is given by a map $\xi : [1, n] \to [1, m]$. That is, n flows—effectively L (or less) 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 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 $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$. Flow i's performance $x^i(\boldsymbol{\eta}, \boldsymbol{\lambda})$ is induced by the performance function of the service class that flow i is mapped to by ξ. When the traffic rate $\boldsymbol{\lambda}$ is fixed, we will omit it from the argument list.

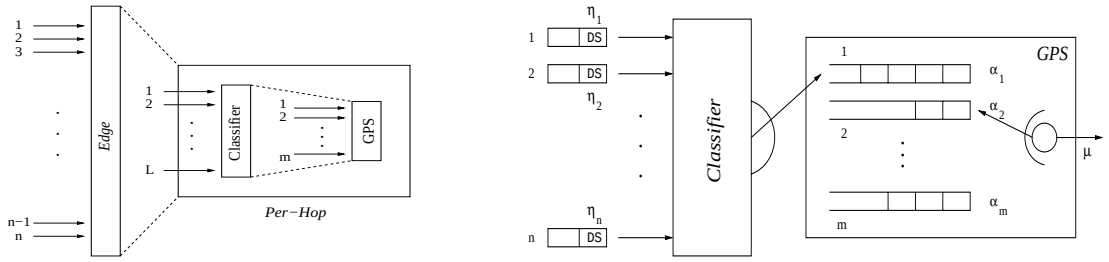

Figure 2.2. Left: Aggregate-flow QoS control affected by two stages of "information loss" via many-to-one coarsification—at edge and per-hop. Right: η value in DS field of IP datagram is used by the classifier to select service class in GPS packet scheduler.

2.3.2 Per-hop Control Properties

There are three properties of the per-hop control, listed below, which are of interest and deemed desirable 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, otherwise. In the following, $i \in [1, n]$ refers to the end user, and $x^i(\cdot)$ denotes the individual user's performance function. The properties are:

(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 $x^i(\boldsymbol{\eta}) \leq x^j(\boldsymbol{\eta})$.

In the definitions, the range of $\boldsymbol{\eta}$ is such that the perturbations remain in the n-dimensional lattice, i.e., $\boldsymbol{\eta} + \mathbf{e}_i, \boldsymbol{\eta} - \mathbf{e}_i \in [1, L]^n$. Property (A1) states that, other things being equal, increasing the label value of flow i improves the QoS received by flow i (recall that "small" means "better" QoS in our representation). 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. Note that (B) has the immediate consequence $x^i(\boldsymbol{\eta}) = x^j(\boldsymbol{\eta}) \Leftrightarrow \eta_i = \eta_j$. 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. We will show that the three properties, collectively, facilitate effective QoS differentiation and control via η control—i.e., scalar QoS control—and furthermore, allow selfish users to share resources efficiently when setting their η values commensurate with their QoS requirements.

2.4 Edge Control

2.4.1 Access 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 per-flow

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 at will—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)). This can be done by performing per-flow policing, traffic shaping, or assigning costs via pricing. Open-loop control is used in the Assured Service and Expedited Service instantiations of differentiated services—also called *absolute* differentiated services [18]—and is generally suited for short-lived flows for which feedback control, when subject to long round-trip times (RTT), is ineffective. Figure 2.3 depicts the overall structure of the end-to-end control framework.

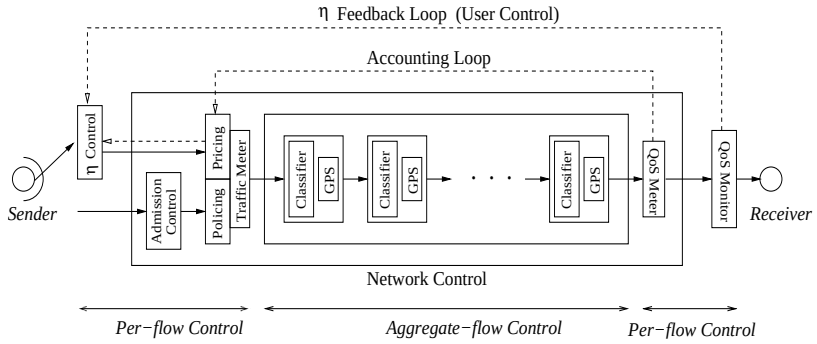

Figure 2.3. Structure of forward QoS control path. "Lower" path comprised of admission control, policing/shaping, per-hop control—open-loop control. "Upper" control path comprised of dynamic η control, pricing, receiver QoS monitoring, QoS feedback—closed-loop control.

2.4.2 End-to-end Control

Our framework (also referred to as *relative* differentiated services in [18]) allows end-to-end control to dynamically adjust the η value in accordance with a user's QoS needs. Properties (A1), (A2), and (B) admit to composability in a WAN environment where a user's traffic flow goes through several hops along an end-to-end path. That is, if a property holds for any single per-hop control, it also holds for a sequence of

per-hop controls in a network of switches when viewed as implementing a composite performance function[2]. An end-to-end control of the form

$$\eta_i(t+\tau) = \begin{cases} \eta_i(t) + 1, & \text{if } \mathbf{x}^i > \boldsymbol{\theta}^i, \\ \eta_i(t) - 1, & \text{if } \mathbf{x}^i < \boldsymbol{\theta}^i, \\ \eta_i(t), & \text{otherwise,} \end{cases} \tag{2.4.1}$$

where $\boldsymbol{\theta}^i$ represents user i's QoS requirement vector—i.e., expressed as a threshold with delay less than θ_1^i, packet loss rate less than θ_2^i—and $\tau > 0$ represents the next update, is asymptotically stable with respect to a *single* user[3]. Properties (A2) and (B) reflect the *resource-boundedness* property of a router, and come into play when considering a collection of selfish users engaged in end-to-end scalar QoS control, and the dynamics this induces as a result of interaction.

2.5 User Control

2.5.1 User Utility and Selfishness

User i's QoS requirement 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 *monotonicity* properties[4]

$$\partial U_i/\partial \lambda_i \geq 0, \ \partial U_i/\partial \mathbf{x}^i \leq 0, \text{ and } \ \partial U_i/\partial p_i \leq 0. \tag{2.5.1}$$

Other things being equal, an increase in the traffic rate is favorably received by a user, so is an improvement in QoS, but an increase in the price charged by the service provider has a detrimental effect on user satisfaction. These are minimal,

[2]In general, under flow conservation for (A1) and (A2), or certain packet loss dominance conditions.
[3]This assumes a total order on the union of reachable and required QoS vectors. See [10] for a discussion of QoS ordering.
[4]U_i need not be differentiable, nor even be continuous. We use continuous notation here for notational clarity; monotonicity is the only property required.

weak requirements on the qualitative form of user utility. If η control is allowed to be exercised by the user, then a *selfish* user i can be defined as performing the self-optimization

$$\max_{\eta_i \in [1,L]} U_i(\lambda_i, \mathbf{x}^i, p_i) \tag{2.5.2}$$

where 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 (nonincreasing) function of $\mathbf{x}^i$ which corresponds to the price function exported by the service provider. A slightly different formulation of selfish, "cost-conscious" user behavior is obtained by the constrained optimization formulation

$$\begin{aligned} &\min_{\eta_i} \ \lambda_i p_i(\mathbf{x}^i) \\ &\text{subject to} \quad \mathbf{x}^i \leq \boldsymbol{\theta}^i \end{aligned} \tag{2.5.3}$$

where $\boldsymbol{\theta}^i$ is user i's QoS requirement vector. Thus the user wants to minimize cost—i.e., achieve efficient resource allocation—while satisfying her QoS requirements. Threshold utilities expressed as bounds on the QoS received is a useful means of representing and conveying a user's QoS requirement—delay less than 33ms, packet loss rate less 10^{-4}, jitter less than 3ms, and so forth. The user is asked to convey her QoS preference as a quantifiable threshold when interacting with the network system (e.g., through a Web browser interface) which is employed in some practical systems [53].

2.5.2 Noncooperative Game

User i's QoS is influenced by the actions (η_j values) of other users $j \neq i$ via $\mathbf{x}^i = x^i(\boldsymbol{\eta})$ as captured by properties (A2) and (B). If all users engage in self-optimization, this leads to a *noncooperative game*. The first point-of-interest is *stability*. In a noncooperative game, a configuration $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ which determines the global QoS allocation is stable if no user, under (unilateral) selfish actions, can improve her

utility from that achieved at $\boldsymbol{\eta}$. More precisely, $\boldsymbol{\eta}$ is a stable configuration or *Nash equilibrium* if for all users $i \in [1, n]$,

$$U_i(\lambda_i, x^i(\boldsymbol{\eta} + c\,\mathbf{e}_i), p_i(\boldsymbol{\eta} + c\,\mathbf{e}_i)) \leq U_i(\lambda_i, x^i(\boldsymbol{\eta}), p_i(\boldsymbol{\eta})) \tag{2.5.4}$$

for all $c \in \mathbb{Z}$ such that $\eta_i + c\,\mathbf{e}_i \in [1, L]$. Since all users are stuck at $\boldsymbol{\eta}$ with respect to selfish moves, the system finds itself at an impasse, i.e., rest point. A similar characterization holds for (2.5.3). Existence of Nash equilibria and their efficiency properties are of import since they characterize the behavioral aspect of a differentiated services network when put into action in a noncooperative environment such as the Internet. We will show that the global resource allocation properties in a noncooperative network environment are intimately tied to the properties exported by the per-hop control.

2.6 Service Provider Control

For a single router *shared* by flows i and j, the only pricing constraint we impose is

$$\mathbf{x}^i \leq \mathbf{x}^j \Rightarrow p_i \geq p_j. \tag{2.6.1}$$

That is, the better the QoS received at a shared resource (i.e., router), the higher the per unit flow cost charged to the user receiving superior QoS. Since $\mathbf{x}^i \leq \mathbf{x}^j$ if, and only if, the relative resources (in the present framework, bandwidth) allocated to flow i are greater than that of flow j, relation (2.6.1) just says that the more resources a flow consumes—thus receiving superior QoS—the higher the cost it incurs vis-à-vis a flow that consumes comparatively fewer resources. Relation (2.6.1), due to its generality, leaves open the degree of freedom of setting the *magnitude* of the prices which we assume is under the control of a service provider. The service provider can be treated as yet another player in the game—assigned the index zero—and, if selfish, will try to maximize her individual utility U_0. U_0 is assumed to have the form of revenue minus cost (i.e., profit) given by $U_0(\boldsymbol{\eta}, \boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i p_i(\mathbf{x}^i) - \mathit{Cost}_0$ where Cost_0 is the total cost incurred by the service provider in delivering the services. The

service provider exports a *price function* $p = p(\mathbf{x})$ where $p(\cdot)$ is monotone decreasing in $\mathbf{x}$. Thus a selfish service provider performs the self-optimization

$$\max_{p(\cdot)} \sum_{i=1}^{n} \lambda_i p_i(\mathbf{x}^i) \tag{2.6.2}$$

assuming fixed $Cost_0$. "Closing" the system by incorporating the actions of a selfish ISP leads to a $(n+1)$-player noncooperative game.

3 OPTIMAL AGGREGATE-FLOW SCHEDULING

In this chapter, we present a theoretical framework for reasoning about aggregate-flow QoS provisioning, constrained to be implementable in IP networks. We develop a theory of optimal aggregate-flow per-hop control and the properties they exhibit which facilitate end-to-end QoS via the joint action of aggregate-flow control per-hop and per-flow control at the edge. We also study the stability and efficiency properties of the overall network system when users are allowed to influence the choice of scalar values in the DS field at the edge, and service providers export costs to users commensurate with the QoS received. The results in this chapter are also published in [68, 69].

3.1 Optimal Classifiers and Per-hop Control

We take a reductionist approach to optimal aggregate-flow per-hop control by first defining what optimal *per-flow* control is when packets are enscribed with a value from L possible choices. Aggregate-flow control can then be viewed as an *approximation* to the QoS achieved by per-flow control in a well-defined sense. Comparability between aggregate-flow and per-flow control is facilitated by the fact that, even in aggregate-flow control, an end user's QoS remains well-defined, and the loss in power due to coarseness affected by flow aggregation can be exactly quantified.

3.1.1 Optimal Per-flow Classification

Consider the per-flow control or classifier problem for n users who choose packet labels from integer set $[1, L]$. Technically, per-flow classification means $n = m$ (each flow's service can be individually configured), and L is either greater or smaller than

n. In general, the range L may be finite or unbounded, and the variable η_i discrete or continuous. The influence of *boundedness* and *discreteness* can be subtle, and its effect is shown in Section 3.1.4 with respect to system optimality where we quantify the negative performance impact of boundedness and discreteness affected by loss of resolution. When n users mark their flows with a value $\eta_i \in [1, L]$ drawn from the metric space $[1, L]$ 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. For example, η_i may represent bandwidth demand in units of Mbps. If network resources are *infinite*, then a flow's request can be satisfied based on the η_i value specified, without consideration of the needs specified by other flows (except, possibly, for pricing issues). That is, independence or decoupling holds. If, on the other hand, resources are *finite*—an OC-12 link is shared among bandwidth insensitive users—then, in general, the users' collective resource demand may exceed the available bandwidth. In the presence of such *resource contention*, a conflict resolution scheme is needed, including the criteria by which resource allocation is decided.

Assume available bandwidth is normalized such that total available bandwidth is $\mu = 1$. First, assume $\eta_i \in \mathbb{R}_+$ is a *continuous* variable over the suitable real interval $[0, 1]$, expressing user i's normalized bandwidth demand *per unit flow*. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ 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 let $\omega_i = \alpha_i / \lambda_i$ denote the fraction of resources allocated to i per unit flow. Under the above *semantics*, given $\boldsymbol{\eta}$ (and $\boldsymbol{\lambda}$), the optimization

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (\eta_i - \omega_i)^2 \tag{3.1.1}$$

measures the "goodness" of a resource allocation $\boldsymbol{\omega}$ with respect to users' codified needs $\boldsymbol{\eta}$ in the mean-square sense[1]. Since (3.1.1) penalizes by the *difference* error, the relative importance of higher η_i values is preserved, and resources are apportioned

[1]The generalization to other norms is treated separately.

accordingly. For general $\eta_i \in \mathbb{R}_+$, including the discrete and bounded case $\eta_i \in \{1, \ldots, L\}$ which is of special interest, define the normalization

$$\hat{\eta}_i = \begin{cases} \frac{\eta_i - \eta_{\min}}{\eta_{\max} - \eta_{\min}}, & \text{if } \eta_{\max} \neq \eta_{\min}, \\ 1, & \text{otherwise,} \end{cases} \tag{3.1.2}$$

where $\eta_{\min}$, $\eta_{\max}$ are the minimum and maximum values of $\{\eta_1, \eta_2, \ldots, \eta_n\}$, respectively. Note that $\hat{\eta}_i \in [0, 1]$, and unless all η_i values are equal, $\hat{\eta}_{\min} = 0$ and $\hat{\eta}_{\max} = 1$. Let $\hat{\omega}_i$ denote the normalization of ω_i via (3.1.2). Given $\boldsymbol{\eta}$, the optimization corresponding to (3.1.1) is

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (\hat{\eta}_i - \hat{\omega}_i)^2. \tag{3.1.3}$$

(3.1.3) realizes the same semantics as (3.1.1), however, generalized by the function or "code" (it is not 1-1) given by (3.1.2) to η_i values not restricted to the real unit interval $[0, 1]$. If L is bounded, then the 1-1 function $\hat{\eta}_i = \eta_i / L$ achieves a similar purpose. (3.1.3) possesses the same desirable properties as (3.1.1), which are characterized by the following two results.

Proposition 3.1.4 (Optimal Per-flow Classifier) *Given $\boldsymbol{\eta}$, $\boldsymbol{\lambda} \in \mathbb{R}_+^n$, the complete solution set to* (3.1.3) *is*

$$\alpha_i = (1 - \nu) \frac{\lambda_i \hat{\eta}_i}{\sum_{j=1}^{n} \lambda_j \hat{\eta}_j} + \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \qquad i \in [1, n], \tag{3.1.5}$$

where $0 \leq \nu \leq 1$ is a parameter which defines a continuous family of solutions.

Proof. (i) First we show that any $\boldsymbol{\alpha}$ given by (3.1.5) is an optimal solution to (3.1.3). To achieve this, we just need to show that such $\boldsymbol{\alpha}$ satisfies $\hat{\eta}_i = \hat{\omega}_i$ for $i \in [1, n]$.

(a) If $\eta_{max} = \eta_{min}$, then $\hat{\eta}_i = 1$, $i \in [1, n]$. From (3.1.5), we have

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

and

$$\omega_i = \frac{\alpha_i}{\lambda_i} = \frac{1}{\sum_{j=1}^{n} \lambda_j}, \qquad i \in [1, n].$$

Since $\omega_1 = \cdots = \omega_n$, we get $\hat{\omega}_i = 1$, $i \in [1, n]$, which means $\hat{\eta}_i = \hat{\omega}_i$, $i \in [1, n]$.

(b) If $\eta_{max} \neq \eta_{min}$, then $\hat{\eta}_{min} = 0$, and $\hat{\eta}_{max} = 1$. From (3.1.5), we have

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

and

$$\omega_{min} = \frac{\nu}{\sum_{j=1}^{n} \lambda_j}\,, \qquad \omega_{max} = \frac{1-\nu}{\sum_{j=1}^{n} \lambda_j \hat{\eta}_j} + \frac{\nu}{\sum_{j=1}^{n} \lambda_j}\,.$$

Because $\hat{\omega}_{min} \neq \hat{\omega}_{max}$, from (3.1.2),

$$\hat{\omega}_i = \frac{\omega_i - \omega_{min}}{\omega_{max} - \omega_{min}}\,, \qquad i \in [1, n]\,. \tag{3.1.6}$$

Substitute ω_i, ω_{min} and ω_{max} in (3.1.6), we will get $\hat{\omega}_i = \hat{\eta}_i$, $i \in [1, n]$.

(ii) Next we show that (3.1.5) represents a complete solution set for (3.1.3). Suppose $\boldsymbol{\alpha'} = (\alpha'_1, \cdots, \alpha'_n)$ is an optimal solution to (3.1.3). We will find some ν', $0 \leq \nu' \leq 1$ such that

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

From the first part of the proof, we have seen that some $\alpha_1, \cdots, \alpha_n$ can achieve $\hat{\eta}_i = \hat{\omega}_i$. Since $\alpha'_1, \cdots, \alpha'_n$ is an optimal solution to (3.1.3), $\hat{\eta}_i = \hat{\omega}'_i$.

(a) If $\eta_{max} = \eta_{min}$, then $\hat{\eta}_i = 1$, and $\hat{\omega}'_i = 1$, $i \in [1, n]$. We have

$$\frac{\alpha'_1}{\lambda_1} = \cdots = \frac{\alpha'_n}{\lambda_n}\,,$$

and

$$\alpha'_1 + \cdots + \alpha'_n = 1\,.$$

By solving the above two equations, we can get

$$\alpha'_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}\,.$$

In (3.1.5), let $\nu' = c$, $0 \leq c \leq 1$, then we get the same $\alpha'_1, \cdots, \alpha'_n$.

(b) If $\eta_{max} \neq \eta_{min}$, then $\hat{\omega}'_{min} = \hat{\eta}_{min} = 0$, and $\hat{\omega}'_{max} = \hat{\eta}_{max} = 1$. Thus $\omega'_{max} > \omega'_{min}$. We have

$$\frac{\omega'_i - \omega'_{min}}{\omega'_{max} - \omega'_{min}} = \hat{\eta}_i \,, \quad i \in [1, n] \,, \tag{3.1.7}$$

and

$$\omega'_1 \lambda_1 + \cdots + \omega'_n \lambda_n = 1 \,. \tag{3.1.8}$$

From (3.1.7) and (3.1.8), we can represent ω'_i, $i \in [1, n]$, as

$$\omega'_i = \omega'_{min} + \hat{\eta}_i \frac{1 - \omega'_{min} \sum_{j=1}^{n} \lambda_j}{\sum \hat{\eta}_j \lambda_j} \,.$$

Therefore,

$$\alpha'_i = \omega'_{min} \lambda_i + (1 - \omega'_{min} \sum_{j=1}^{n} \lambda_j) \frac{\hat{\eta}_i \lambda_i}{\sum \hat{\eta}_j \lambda_j} \,.$$

In (3.1.5), let $\nu' = \omega'_{min} \sum_{j=1}^{n} \lambda_j$, then we get the same $\alpha'_1, \cdots, \alpha'_n$.

We shall show $0 \leq \nu' \leq 1$. $\nu' = \omega'_{min} \sum_{j=1}^{n} \lambda_j \geq 0$ because $\omega'_{min} \geq 0$. On the other hand, since $\omega'_{min} < \omega'_{max}$,

$$\omega'_{min} \sum_{j=1}^{n} \lambda_j < \sum_{j=1}^{n} \omega'_j \lambda_j = 1.$$

■

The parameter ν, which stems from the dimension reduction associated with (3.1.2), has an appealing interpretation. The second term in (3.1.5) corresponds to the proportional share achieved by FIFO scheduling, whereas the first term corresponds to proportional share of the corresponding *virtual* flows $\lambda_i \hat{\eta}_i$, which are the original flow rates weighted by their relevancy variable $\hat{\eta}_i$ derived from η_i. Thus, if $\nu = 1$, then the per-hop control effectively ignores the label values and behaves as a FIFO queue. If $\nu = 0$, then the router acts like a GPS scheduler with service weights given by the first term. For any other value of ν, (3.1.5) represents a convex combination of the two behavioral modes.

Proposition 3.1.9 (Per-flow Classifier Properties) *The optimal per-flow classifier given in* (3.1.5) *satisfies properties* (A1), (A2), *and* (B).

Proof. Denote the normalization of $\boldsymbol{\eta}$ by $\hat{\eta}_i(\boldsymbol{\eta})$, $i \in [1, n]$, and the normalization of $\boldsymbol{\eta} + \boldsymbol{e}_i$ by $\hat{\eta}_i(\boldsymbol{\eta} + \boldsymbol{e}_i)$, $i \in [1, n]$. Similarly, we use notations $\omega_i(\boldsymbol{\eta})$ and $\omega_i(\boldsymbol{\eta} + \boldsymbol{e}_i)$, $i \in [1, n]$, to denote resources decided by $\boldsymbol{\eta}$ and $\boldsymbol{\eta} + \boldsymbol{e}_i$.

(i) *Property (A1).* Consider $\omega_i = \frac{\alpha_i}{\lambda_i}$. Rewrite (3.1.5) as

$$\omega_i = \frac{(1-\nu)}{\sum_{j=1}^{n} \lambda_j (\frac{\hat{\eta}_j}{\hat{\eta}_i})} + \frac{\nu}{\sum_{j=1}^{n} \lambda_j}.$$

Because $\hat{\eta}_i(\boldsymbol{\eta} + \boldsymbol{e}_i) \geq \hat{\eta}_i(\boldsymbol{\eta})$ and $\hat{\eta}_j(\boldsymbol{\eta} + \boldsymbol{e}_i) \leq \hat{\eta}_j(\boldsymbol{\eta})$ for $j \neq i$, $\frac{\hat{\eta}_j(\boldsymbol{\eta}+\boldsymbol{e}_i)}{\hat{\eta}_i(\boldsymbol{\eta}+\boldsymbol{e}_i)} \leq \frac{\hat{\eta}_j(\boldsymbol{\eta})}{\hat{\eta}_i(\boldsymbol{\eta})}$, $j \in [1, n]$. Hence, $\omega_i(\boldsymbol{\eta} + \boldsymbol{e}_i) \geq \omega_i(\boldsymbol{\eta})$. Furthermore, x^i is a monotone function of ω_i, so $x^i(\boldsymbol{\eta} + \boldsymbol{e}_i) \leq x^i(\boldsymbol{\eta})$. Similarly, we will get $x^i(\boldsymbol{\eta} - \boldsymbol{e}_i) \geq x^i(\boldsymbol{\eta})$.

(ii) *Property (A2).* Consider $\omega_j = \frac{\alpha_j}{\lambda_j}$, $j \neq i$.

(a) If $\eta_i \neq \eta_{max}$, rewrite (3.1.5) as

$$\omega_j = \frac{(1-\nu)\hat{\eta}_j}{\lambda_i \hat{\eta}_i + \sum_{k \neq i} \lambda_k \hat{\eta}_k} + \frac{\nu}{\sum_{k=1}^{n} \lambda_k}.$$

Because $\eta_i \neq \eta_{max}$, $\hat{\eta}_i(\boldsymbol{\eta} + \boldsymbol{e}_i) \geq \hat{\eta}_i(\boldsymbol{\eta})$, and $\hat{\eta}_k(\boldsymbol{\eta} + \boldsymbol{e}_i) = \hat{\eta}_k(\boldsymbol{\eta})$ for $k \neq i$. Hence, $\omega_j(\boldsymbol{\eta} + \boldsymbol{e}_i) \leq \omega_j(\boldsymbol{\eta})$;

(b) If $\eta_i = \eta_{max}$, then rewrite (3.1.5) as

$$\omega_j = \frac{(1-\nu)}{\lambda_i (\frac{\hat{\eta}_i}{\hat{\eta}_j}) + \sum_{k \neq i} \lambda_k (\frac{\hat{\eta}_k}{\hat{\eta}_j})} + \frac{\nu}{\sum_{k=1}^{n} \lambda_k}.$$

Because $\eta_i = \eta_{max}$ for configuration $\boldsymbol{\eta}$, $\eta_i + 1 = \eta_{max}$ for configuration $\boldsymbol{\eta} + \boldsymbol{e}_i$. By (3.1.2), $\frac{\hat{\eta}_i(\boldsymbol{\eta}+\boldsymbol{e}_i)}{\hat{\eta}_j(\boldsymbol{\eta}+\boldsymbol{e}_i)} \geq \frac{\hat{\eta}_i(\boldsymbol{\eta})}{\hat{\eta}_j(\boldsymbol{\eta})}$, and $\frac{\hat{\eta}_k(\boldsymbol{\eta}+\boldsymbol{e}_i)}{\hat{\eta}_j(\boldsymbol{\eta}+\boldsymbol{e}_i)} = \frac{\hat{\eta}_k(\boldsymbol{\eta})}{\hat{\eta}_j(\boldsymbol{\eta})}$, $k \neq i$. Hence, $\omega_j(\boldsymbol{\eta} + \boldsymbol{e}_i) \leq \omega_j(\boldsymbol{\eta})$.

Therefore, in both cases, $\omega_j(\boldsymbol{\eta} + \boldsymbol{e}_i) \leq \omega_j(\boldsymbol{\eta})$. From the monotonicity of x^i, we get $x^j(\boldsymbol{\eta} + \boldsymbol{e}_i) \geq x^j(\boldsymbol{\eta})$. Similarly, we can also get $x^j(\boldsymbol{\eta} - \boldsymbol{e}_i) \leq x^j(\boldsymbol{\eta})$.

(iii) *Property (B).* For $i \neq j$, $\eta_i \geq \eta_j$ implies $\hat{\eta}_i \geq \hat{\eta}_j$. By (3.1.5), $\omega_i \geq \omega_j$ if $\hat{\eta}_i \geq \hat{\eta}_j$. Hence $x^i(\boldsymbol{\eta}) \leq x^j(\boldsymbol{\eta})$. ■

3.1.2 Optimal Aggregate-flow Classification

With the semantic set-up of optimal per-flow classification, let us consider the aggregate-flow classifier problem where $n > m$. The original aggregate-flow classifier problem, $n > L = m$, is subsumed by the more general set-up where L can take on any value. From a QoS provisioning perspective, the ultimate goal of a differentiated services network comprised of aggregate-flow per-hop controls is the provisioning of *end-to-end QoS* commensurate with each user's needs. Aggregate-flow control, whether it has many or few labels, must service n flows using $m < n$ service classes which results in a reduced ability to effectively shape end-to-end QoS with respect to the performance criterion (3.1.3) when compared to per-flow control. That is, the minimum value of (3.1.3) achieved by optimal per-hop control is smaller than that of optimal aggregate-flow control. This is a consequence of a more general result given by Proposition 3.1.12.

We give a formal definition of aggregate-flow per-hop control. An *aggregate-flow per-hop control with parameter* (m, n) is a function

$$\Phi_{m,n} : (\boldsymbol{\eta}, \boldsymbol{\lambda}) \mapsto (\xi, \tilde{\boldsymbol{\alpha}}), \tag{3.1.10}$$

where $\xi : [1, n] \to [1, m]$ is the *classifier* and $\tilde{\boldsymbol{\alpha}} = (\alpha^1, \ldots, \alpha^m)$ is the vector of service weights assigned to the m service classes. With respect to end users, $\Phi_{m,n}$ induces—explicitly or implicitly—a performance function $\varphi^i_{m,n}$ for each user $i \in [1, n]$

$$\varphi^i_{m,n} : (\boldsymbol{\eta}, \boldsymbol{\lambda}) \mapsto \alpha_i, \tag{3.1.11}$$

where $\alpha_i = \varphi^i_{m,n}(\boldsymbol{\eta}, \boldsymbol{\lambda}) \geq 0$ is user i's share of the bandwidth allocated by $\Phi_{m,n}$. Since the traffic rate $\boldsymbol{\lambda}$ is fixed, we will omit it from the argument list. The two-stage interpretation of aggregate-flow per-hop control is depicted in Figure 2.2.

The resource share received by individual flows under $\Phi_{m,n}$ can be computed as follows. Let

$$U_k = \{i \in [1, n] : \ \xi(i) = k\}, \qquad k \in [1, m],$$

be the partition of $[1, n]$ induced by ξ. For $i \in U_k$, set α_i such that $\sum_{i \in U_k} \alpha_i = \alpha^k$, and $\alpha_i/\lambda_i =$ constant. This is the share received by user $i \in [1, n]$.

Proposition 3.1.12 (Service Class Monotonicity) *Let $\Phi_{m,n}$ be an aggregate-flow per-hop control, and let $S_m = \{\boldsymbol{\alpha} : \varphi_{m,n}(\boldsymbol{\eta}) = \boldsymbol{\alpha}$ for some $\boldsymbol{\eta}\}$. Then* (3.1.3) *achieves a smaller value with more service classes, i.e., for all m' satisfying $m + 1 \leq m' \leq n$,*

$$\Big\{ \min_{\boldsymbol{\alpha} \in S_{m'}} \sum_{i=1}^{n} (\hat{\eta}_i - \hat{\omega}_i)^2 \Big\} \;\leq\; \Big\{ \min_{\boldsymbol{\alpha} \in S_m} \sum_{i=1}^{n} (\hat{\eta}_i - \hat{\omega}_i)^2 \Big\}.$$

Proof. Let $\Phi_{m,n}$ be an arbitrary but fixed m-class aggregate-flow per-hop control. Because $m + 1 \leq n$, under $\Phi_{m,n}$ there exists a service class k that receives more than one flows. Construct an $(m + 1)$-class aggregate-flow per-hop control $\Phi_{m+1,n}$ as follows: $\Phi_{m+1,n}$ takes same class mapping and weight assignment as $\Phi_{m,n}$ except dividing class k into two classes k_1, k_2 with weight α^{k_1}, α^{k_2} satisfying

$$\alpha^{k_1} + \alpha^{k_2} = \alpha^k, \qquad \frac{\alpha^{k_1}}{\sum_{i \in U_{k_1}} \lambda_i} = \frac{\alpha^{k_2}}{\sum_{i \in U_{k_2}} \lambda_i}.$$

We have

$$\varphi^i_{m,n}(\boldsymbol{\eta}) = \varphi^i_{m+1,n}(\boldsymbol{\eta}), \qquad i \in [1, n].$$

Thus any $\boldsymbol{\alpha} \in S_m$ can be achieved by some $\Phi_{m+1,n}$. Therefore, $S_m \subseteq S_{m+1} \subseteq \cdots \subseteq S_n$, and the proposition follows. ■

Consider a special type of aggregate-flow per-hop control $\Phi_{m,n}$—called *Reduction Classifier*—whose behavior is completely determined by its classifier $\xi : [1, n] \to [1, m]$, in the following sense. On input $(\boldsymbol{\eta}, \boldsymbol{\lambda})$, $\Phi_{m,n}$ behaves as shown in Figure 3.1: It reduces the n user problem to an m user per-flow classification problem by aggregation of component flows and centroid computation, then solves the reduced problem by applying the optimal per-flow classification solution.

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

1. Compute $\lambda^k = \sum_{i \in U_k} \lambda_i$ for each $k \in [1, m]$.
2. Compute $\hat{\eta}^k$ for $k \in [1, m]$ as follows,

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

3. Use per-flow optimal solution (Proposition 3.1.4) with new input $\tilde{\boldsymbol{\eta}} = (\hat{\eta}^1, \ldots, \hat{\eta}^m)$, $\tilde{\boldsymbol{\lambda}} = (\lambda^1, \ldots, \lambda^m)$, to solve the reduced per-flow classifier problem consisting of m superusers.

Figure 3.1. Behavior of reduction classifier

Theorem 3.1.13 (Reduction Classifier) *Let $\Phi_{m,n}$ be a reduction classifier represented by its classifier ξ. Then $\Phi_{m,n}$ is an* optimal aggregate-flow per-hop control*, i.e., satisfies* (3.1.3) *if, and only if, ξ is a solution to*

$$\min_{\xi'} \sum_{k \in [1,m]} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\eta}^k)^2 \tag{3.1.14}$$

where the minimum ranges over all reduction classifiers ξ'.

Proof. Let $\Phi_{m,n}$ be an aggregate-flow per-hop control. Let $\tilde{\boldsymbol{\omega}} = (\omega^1, \cdots, \omega^m)$ where $\omega^k = \alpha^k / \lambda^k$, $k \in [1, m]$. We have $\omega_i = \omega^{\xi(i)}$ and $\hat{\omega}_i = \hat{\omega}^{\xi(i)}$. Thus $\Phi_{m,n}$ is the optimal solution to (3.1.3) if and only if $\Phi_{m,n}$ satisfies

$$\min_{(\xi, \tilde{\boldsymbol{\alpha}})} \sum_{k \in [1,m]} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^k)^2.$$

For a reduction classifier, $\tilde{\boldsymbol{\alpha}} = (\alpha^1, \ldots, \alpha^m)$ is computed by optimal per-flow classification with input $\tilde{\boldsymbol{\eta}} = (\hat{\eta}^1, \ldots, \hat{\eta}^m)$, thus $\hat{\eta}^k = \hat{\omega}^k$, $k \in [1, m]$. Hence the

theorem holds if we can show that in an optimal aggregate-flow per-hop control, The weight α^k of service class k, $k \in [1, m]$ is computed as

$$\hat{\omega}^k = \begin{cases} 0, & \text{if } \exists i \in U_k,\ \hat{\eta}_i = 0; \\ 1, & \text{if } \exists i \in U_k,\ \hat{\eta}_i = 1; \\ \sum_{i \in U_k} \hat{\eta}_i / |U_k|, & \text{otherwise.} \end{cases} \tag{3.1.15}$$

We will prove the above by contradiction.

(i) For class k, if $\exists i^* \in U_k$, $\hat{\eta}_{i^*} = 0$, then (3.1.15) indicates that $\hat{\omega}^k = 0$. Suppose $\hat{\omega}^k \neq 0$. Then $\exists j \neq k$, $\hat{\omega}^j = 0$. We will construct another aggregate-flow per-hop control $\Phi'_{m,n}$ with same $(\hat{\omega}^1, \cdots, \hat{\omega}^m)$ but different ξ' as follows: For flow i^*, $\xi'(i^*) = j$; For any other flow $i \neq i^*$, $\xi'(i) = \xi(i)$. We have $(\hat{\eta}_{i^*} - \hat{\omega}^k)^2 > 0$, and $(\hat{\eta}_{i^*} - \hat{\omega}^j)^2 = 0$. Therefore,

$$\sum_{j=1}^{m} \sum_{i \in U'_j} (\hat{\eta}_i - \hat{\omega}^j)^2 < \sum_{k=1}^{m} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^k)^2,$$

which is contradictory to the optimality of $\Phi_{m,n}$.

(ii) For class k, if $\exists i^* \in U_k$, $\hat{\eta}_{i^*} = 1$, then (3.1.15) indicates that $\hat{\omega}^k = 1$. This case can be proved by a similar argument as in (i).

(iii) For class k, if $\forall i \in U_k, \hat{\eta}_i \neq 0$ and $\hat{\eta}_i \neq 1$, then(3.1.15) indicates that $\hat{\omega}^k = \sum_{i \in U_k} \hat{\eta}_i / |U_k|$. Suppose $\hat{\omega}^k \neq \sum_{i \in U_k} \hat{\eta}_i / |U_k|$. By (i), $\exists k_1 \neq k, \hat{\omega}^{k_1} = 0$ since $\forall i \in U_k, \hat{\eta}_i \neq 0$; similarly, we know $\exists k_2 \neq k, \hat{\omega}^{k_2} = 1$. Therefore, we can construct another aggregate-flow per-hop control $\Phi'_{m,n}$ with same ξ but different $\tilde{\boldsymbol{\omega}}' = (\omega^{(1)}, \cdots, \omega^{(m)})$ as follows: For $j \neq k$, $\hat{\omega}^{(j)} = \hat{\omega}^j$; For $j = k$, $\hat{\omega}^{(j)} = \sum_{i \in U_k} \hat{\eta}_i / |U_k|$. By the property of mean square, we have

$$\sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^{(k)})^2 < \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}_k)^2,$$

and hence

$$\sum_{k=1}^{m} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^{(k)})^2 < \sum_{k=1}^{m} \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^k)^2,$$

which is contradictory to the optimality of $\Phi_{m,n}$. ■

Theorem 3.1.13 shows that 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 optimal aggregate-flow per-hop control is a clustering or classification problem in the statistical classification sense.

A classifier ξ is *well-formed* (also called a *grouping*) if the three conditions $\eta_i < \eta_j$, $\xi(i) = \xi(j)$, and $\eta_i \leq \eta_k \leq \eta_j$ jointly imply $\xi(k) = \xi(i)$. Thus if two different label values are mapped to the same service class, then all η values "sandwiched" in-between must be mapped to the same service class. ξ can be represented by well-formed parentheses on the totally ordered set $\eta_1 \leq \eta_2 \leq \ldots \leq \eta_n$, where adjacent values are grouped into the same partition except, possibly, at boundaries.

Theorem 3.1.16 (Grouping) *An optimal aggregate-flow classifier is well-formed.*

Proof. Let $\Phi_{m,n}$ be an optimal aggregate-flow per-hop control with classifier ξ and weight vector $\tilde{\boldsymbol{\alpha}} = (\alpha^1, \cdots, \alpha^m)$. We will first prove that ξ is well-formed in $\hat{\eta}$-space by contradiction.

Suppose $\hat{\eta}_{l_1} < \hat{\eta}_{l_2}$, $\xi(l_1) = \xi(l_2) = k$ and $\exists l$, $\hat{\eta}_{l_1} \leq \hat{\eta}_l \leq \hat{\eta}_{l_2}$ and $\xi(l) = k'$, $k' \neq k$. For class k, let $\hat{\eta}^k_{min}$ and $\hat{\eta}^k_{max}$ be the minimum and maximum value of $\hat{\eta}_i$ for all $i \in U_k$, and let i^k_{min} and i^k_{max} be the indices of $\hat{\eta}^k_{min}$ and $\hat{\eta}^k_{max}$. We also use similar notations for class k'. Clearly $\hat{\eta}^{k'}_{min} \leq \hat{\eta}_l \leq \hat{\eta}_{l_2} \leq \hat{\eta}^k_{max}$. We then construct another aggregate-flow per-hop control $\Phi'_{m,n}$ with classifier ξ' and corresponding $\hat{\boldsymbol{\omega}}' = (\hat{\omega}^{(1)}, \cdots, \hat{\omega}^{(m)})$ as follows:

(i) Suppose $\hat{\omega}^k = \hat{\omega}^{k'}$. Because $\hat{\eta}_{l_1} < \hat{\eta}_{l_2}$, either $\hat{\eta}_{l_1} \neq \hat{\omega}^k$ or $\hat{\eta}_{l_2} \neq \hat{\omega}^k$. Suppose $\hat{\eta}_{l_1} \neq \hat{\omega}^k$. Let $U'_k = U_k \cup U_{k'} - \{l_1\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ and $U'_{k'} = \{l_1\}$, $\hat{\omega}^{(k')} = \hat{\eta}_{l_1}$.

(ii) Suppose $\hat{\omega}^k < \hat{\omega}^{k'}$. For class k and k', do the following:

(a) If $\hat{\eta}^{k'}_{min} \leq \hat{\omega}^k$, then $U'_k = U_k \cup \{i^{k'}_{min}\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ and $U'_{k'} = U_{k'} - \{i^{k'}_{min}\}$, $\hat{\omega}^{(k')} = \hat{\omega}^{k'}$.

(b) If $\hat{\omega}^{k'} \leq \hat{\eta}^k_{max}$, then $U'_k = U_k - \{i^k_{max}\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ and $U'_{k'} = U_{k'} \cup \{i^k_{max}\}$, $\hat{\omega}^{(k')} = \hat{\omega}^{k'}$.

(c) If $\hat{\omega}^k < \hat{\eta}^{k'}_{min}$, $\hat{\eta}^k_{max} < \hat{\omega}^{k'}$, and $\hat{\eta}^{k'}_{min} < \hat{\eta}^k_{max}$, then $U'_k = U_k \cup \{i^{k'}_{min}\} - \{i^k_{max}\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ and $U'_{k'} = U_{k'} \cup \{i^k_{max}\} - \{i^{k'}_{min}\}$, $\hat{\omega}^{(k')} = \hat{\omega}^{k'}$.

(d) If $\hat{\omega}^k < \hat{\eta}^{k'}_{min}$, $\hat{\eta}^k_{max} < \hat{\omega}^{k'}$, and $\hat{\eta}^{k'}_{min} = \hat{\eta}^k_{max}$, then compare $|\hat{\eta}^k_{max} - \hat{\omega}^k|$ and $|\hat{\eta}^{k'}_{min} - \hat{\omega}^{k'}|$. If $|\hat{\eta}^k_{max} - \hat{\omega}^k| \leq |\hat{\eta}^{k'}_{min} - \hat{\omega}^{k'}|$, $U'_k = U_k \cup \{i^{k'}_{min}\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ when $\hat{\omega}^k = 0$, $\hat{\omega}^{(k)} = \sum_{i \in U'_k} \hat{\eta}_i / |U'_k|$ when $\hat{\omega}^k \neq 0$, and $U'_{k'} = U_{k'} - \{i^{k'}_{min}\}$, $\hat{\omega}^{(k')} = \hat{\omega}^{k'}$ when $\hat{\omega}^{k'} = 1$, $\hat{\omega}^{(k')} = \sum_{i \in U'_{k'}} \hat{\eta}_i / |U'_{k'}|$ when $\hat{\omega}^{k'} \neq 1$; If $|\hat{\eta}^k_{max} - \hat{\omega}^k| \geq |\hat{\eta}^{k'}_{min} - \hat{\omega}^{k'}|$, $U'_k = U_k - \{i^k_{max}\}$, $\hat{\omega}^{(k)} = \hat{\omega}^k$ when $\hat{\omega}^k = 0$, $\hat{\omega}^{(k)} = \sum_{i \in U'_k} \hat{\eta}_i / |U'_k|$ when $\hat{\omega}^k \neq 0$, and $U'_{k'} = U_{k'} \cup \{i^k_{max}\}$, $\hat{\omega}^{(k')} = \hat{\omega}^{k'}$ when $\hat{\omega}^{k'} = 1$, $\hat{\omega}^{(k')} = \sum_{i \in U'_{k'}} \hat{\eta}_i / |U'_{k'}|$ when $\hat{\omega}^{k'} \neq 1$.

(iii) $\hat{\omega}^k > \hat{\omega}^{k'}$. Similar to (ii).

In each of the above cases, $U'_j = U_j$ and $\hat{\omega}^{(j)} = \hat{\omega}^j$ for all class j with $j \neq k$ and $j \neq k'$. We have

$$\sum_{i \in U'_k} (\hat{\eta}_i - \hat{\omega}^{(k)})^2 + \sum_{i \in U'_{k'}} (\hat{\eta}_i - \hat{\omega}^{(k')})^2 < \sum_{i \in U_k} (\hat{\eta}_i - \hat{\omega}^k)^2 + \sum_{i \in U_{k'}} (\hat{\eta}_i - \hat{\omega}^{k'})^2. \tag{3.1.17}$$

Hence

$$\sum_{j=1}^{m} \sum_{i \in U'_j} (\hat{\eta}_i - \hat{\omega}^{(j)})^2 < \sum_{j=1}^{m} \sum_{i \in U_j} (\hat{\eta}_i - \hat{\omega}^j)^2, \tag{3.1.18}$$

which is contradictory to the hypothesis that $\Phi_{m,n}$ is optimal. Therefore ξ is well-formed in $\hat{\eta}$-space. Because $\eta_i \leq \eta_j \Leftrightarrow \hat{\eta}_i \leq \hat{\eta}_j$, ξ is also well-formed in η-space. ■

Thus, aggregate-flow per-hop control is, mathematically, an optimal clustering problem. Unlike its many brethren in higher dimensions that are, with few exceptions, NP-complete [28], the clustering problem given by (3.1.14) in Theorem 3.1.13 has a poly-time algorithm; e.g., it can be solved by dynamic programming. When $L = m$—the practically relevant case where there are as many labels as service classes—optimal aggregate-flow classification has a linear time algorithm.

3.1.3 Properties of Optimal Aggregate-flow Classifiers

Although optimal per-flow classifiers satisfy properties (A1), (A2), and (B), the same is not necessarily true of optimal aggregate-flow classifiers.

Theorem 3.1.19 (Aggregate-flow Classifier Properties) *An optimal aggregate-flow per-hop control satisfies property* (B)*, but need not satisfy properties* (A1) *and* (A2)*.*

Proof. Let $\Phi_{m,n}$ be an optimal aggregate-flow per-hop control.

(i) *Property (B).* Consider two flows $i \neq j$ and $\eta_i \geq \eta_j$. Let $k_1 = \xi(\eta_i)$ and $k_2 = \xi(\eta_j)$. Since an optimal aggregate-flow classifier is well-formed. $\eta_i \geq \eta_j$ implies $\hat{\eta}^{k_1} \geq \hat{\eta}^{k_2}$. Thus $\hat{\omega}^{k_1} \geq \hat{\omega}^{k_2}$. Therefore, $\omega_i \geq \omega_j$ and property (B) follows.

(ii) *Property (A1).* Property (A1) does not necessarily hold. Given $\boldsymbol{\eta'} = \boldsymbol{\eta} + \boldsymbol{e}_i$, let ξ' be the classifier and $\tilde{\boldsymbol{\alpha}}' = (\alpha^{(1)}, \cdots, \alpha^{(m)})$ be the weight vector computed by $\Phi_{m,n}$. Property (A1) requires $\omega'_i \geq \omega_i$. If $\xi = \xi'$, then $\hat{\eta}^{\xi'(i)} \geq \hat{\eta}^{\xi(i)}$. Thus $\omega'_i \geq \omega_i$. However, if $\xi \neq \xi'$, i.e. the change of η_i leads to "regrouping", then it is possible that $\omega'_i < \omega_i$. Here is an example.

Let $n = 5$, $m = 3$. Let $\boldsymbol{\lambda} = (1, 10, 1, 1, 1)$, $\boldsymbol{\eta} = (0, 2, 4, 6, 9)$, and $\boldsymbol{\eta'} = \boldsymbol{\eta} + \boldsymbol{e}_4 = (0, 2, 4, 7, 9)$. The results of $\Phi_{m,n}$ for $(\boldsymbol{\lambda}, \boldsymbol{\eta})$ are: $\hat{\boldsymbol{\eta}} = (0, \frac{2}{9}, \frac{4}{9}, \frac{6}{9}, 1)$, $U_1 = \{1, 2\}$, $U_2 = \{3, 4\}$, $U_3 = \{5\}$, and $\tilde{\boldsymbol{\omega}} = (0, \frac{5}{19}, \frac{9}{19})$; The results of $\Phi_{m,n}$ for $(\boldsymbol{\lambda}, \boldsymbol{\eta'})$ are: $\hat{\boldsymbol{\eta}}' = (0, \frac{2}{9}, \frac{4}{9}, \frac{7}{9}, 1)$, $U'_1 = \{1\}$, $U'_2 = \{2, 3\}$, $U'_3 = \{4, 5\}$, and $\tilde{\boldsymbol{\omega}}' = (0, \frac{1}{17}, \frac{3}{17})$. For user 4, $\xi(4) = 2$, $\xi'(4) = 3$, and $\omega_4 = \omega^2 = \frac{5}{19}$, $\omega'_4 = \omega^{(3)} = \frac{3}{17}$, Thus $\omega'_4 < \omega_4$.

(iii) *Property (A2).* Property (A2) does not necessarily hold. Given $\boldsymbol{\eta'} = \boldsymbol{\eta} + \boldsymbol{e}_i$, let ξ' be the classifier and $\tilde{\boldsymbol{\alpha}}' = (\alpha^{(1)}, \cdots, \alpha^{(m)})$ be the weight vector computed by $\Phi_{m,n}$. Property (A2) requires $\forall j \neq i$, $\omega'_j \leq \omega_j$. Following is a counterexample.

Suppose $\xi(i) = k$ and $\forall j \in U_k$, $\hat{\eta}_j \neq 0$ and $\hat{\eta}_j \neq 1$. As shown in the proof of Theorem 3.1.13, $\hat{\omega}^k = \frac{\sum_{j \in U_k} \hat{\eta}_j}{|U_k|}$. If $\xi' = \xi$, and $\hat{\eta}'_i \neq 1$, then $\xi'(i) = k$ and $\hat{\omega}^{(k)} = \frac{\sum_{j \in U'_k} \hat{\eta}'_j}{|U'_k|} = \hat{\omega}_k + \frac{\hat{\eta}'_i - \hat{\eta}_i}{|U_k|}$. Because $\hat{\eta}'_i > \hat{\eta}_i$, $\hat{\omega}^{(k)} > \hat{\omega}^k$. Therefore $\forall j \in U_k$, $\omega'_j > \omega_j$. ■

Property (A2) is more subtle than (A1) and (B), but of import in influencing the stability and dynamical structure of noncooperative networks built on top of aggregate-flow scheduling.

Proposition 3.1.20 (Classifier Properties with $m = L$) *An optimal aggregate-flow per-hop control with parameters $m = L$ satisfies properties* (A1)*,* (A2)*, and* (B)*.*

Proof. Let $\Phi_{L,n}$ be an optimal aggregate-flow per-hop control with parameter $m = L$. Given $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$, because η_i, $i \in [1, n]$, ranges over $[1, L]$, ξ is computed as:

$$\xi(i) = \eta_i, \qquad \hat{\eta}^{\xi(i)} = \hat{\eta}_i, \qquad i \in [1, n], \tag{3.1.21}$$

and $\tilde{\boldsymbol{\alpha}}$ is computed as:

$$\alpha^k = (1 - \nu) \frac{\lambda^k \hat{\eta}^k}{\sum_{l=1}^{L} \lambda^l \hat{\eta}^l} + \nu \frac{\lambda^k}{\sum_{l=1}^{L} \lambda^l}, \qquad k \in [1, L]. \tag{3.1.22}$$

From (3.1.21) and (3.1.22), we can derive user's share of bandwidth allocated by $\Phi_{L,n}$:

$$\alpha_i = \alpha^{\xi(i)} \frac{\lambda_i}{\lambda^{\xi(i)}} = (1 - \nu) \frac{\lambda_i \hat{\eta}_i}{\sum_{j=1}^{n} \lambda_j \hat{\eta}_j} + \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}, \qquad i \in [1, n]. \tag{3.1.23}$$

Compare (3.1.23) with (3.1.5), we can see that $\Phi_{L,n}$ is equal to an optimal per-flow classifier. According to proposition (3.1.9), $\Phi_{L,n}$ satisfies properties (A1), (A2), and (B). ■

The next result shows that the optimal aggregate-flow per-hop control with $m = L$ not only differentiates services among classes, i.e. satisfying property (B), but also preserves an even, uniform QoS separation: the performance difference (in terms of bandwidth per unit flow) between flows is proportional to their η value difference.

Proposition 3.1.24 (QoS Separation with $m = L$) *Let $\Phi_{L,n}$ be an optimal aggregate-flow per-hop control with parameters $m = L$. Given input $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$, for all $i,\ j \in [1, n]$,*

$$\omega_i - \omega_j = c\,(\eta_i - \eta_j) \tag{3.1.25}$$

where c is a constant only depending on $\boldsymbol{\eta}$ and $\boldsymbol{\lambda}$.

Proof. (i) If $\eta_{max} = \eta_{min}$, then $\forall\, i,\ j \in [1, n]$, $\eta_i = \eta_j$ and $\omega_i = \omega_j$. Thus (3.1.25) is satisfied. (ii) If $\eta_{max} \neq \eta_{min}$, we have

$$\omega_i = (1 - \nu) \frac{\hat{\eta}_i}{\sum_{k=1}^{n} \lambda_k \hat{\eta}_k} + \frac{\nu}{\sum_{k=1}^{n} \lambda_k}, \qquad i \in [1, n].$$

Hence,

$$\begin{aligned}\omega_i - \omega_j &= (1-\nu)\frac{\hat{\eta}_i - \hat{\eta}_j}{\sum_{k=1}^{n}\lambda_k\hat{\eta}_k} \\ &= (1-\nu)\frac{\frac{\eta_i-\eta_{min}}{\eta_{max}-\eta_{min}} - \frac{\eta_j-\eta_{min}}{\eta_{max}-\eta_{min}}}{\sum_{k=1}^{n}\lambda_k\frac{\eta_k-\eta_{min}}{\eta_{max}-\eta_{min}}} \\ &= \frac{1-\nu}{\sum_{k=1}^{n}\lambda_k(\eta_k-\eta_{min})}(\eta_i-\eta_j), \qquad i,j\in[1,n].\end{aligned}$$

■

The $L = m$ constraint advanced by Proposition 3.1.20 and Proposition 3.1.24 coincides with practical considerations that derive from an implementation perspective. For example, assuming four bits from the TOS field in IPv4 are used to encode the label set $\{a, a+1, \ldots, a+15\}$ for some $a \geq 0$, then we may configure 16 service classes at routers, one for each of the 16 possible label values. The classifier results and properties for fixed service weights are treated separately.

3.1.4 System Optimality and Structural Properties

To satisfy user i's QoS requirement $\boldsymbol{\theta}^i$, the per-hop control—whatever its specific form—must apportion a fraction $\alpha_i^* \geq 0$ of the available bandwidth. Let α_i^* denote the minimal such bandwidth. We find it more convenient to work in the service weight space $\{\boldsymbol{\alpha} : \boldsymbol{\alpha} \geq 0 \text{ and } \sum_{i=1}^{n}\alpha_i \leq 1\}$. We use $\varphi^i(\cdot)$ to denote the performance function corresponding to $x^i(\cdot)$ which allocates—explicitly or implicitly—a service weight to user i for a given input $\boldsymbol{\eta}$.

Given per-hop control $\Phi_{m,n}$, let $\mathcal{A}$ represents the set of configurations where all users' QoS requirements are satisfied, i.e.

$$\mathcal{A}^* = \{\boldsymbol{\eta} : \varphi^i(\boldsymbol{\eta}) \geq \alpha_i^*, i \in [1,n]\}.$$

A configuration $\boldsymbol{\eta}$ is *system optimal* if $\boldsymbol{\eta} \in \mathcal{A}^*$, and for all $\boldsymbol{\eta}' \neq \boldsymbol{\eta}$, $\boldsymbol{\varphi}(\boldsymbol{\eta}') > \boldsymbol{\varphi}(\boldsymbol{\eta})$ does not hold. In a system optimal configuration, the users' QoS requirements are met while expending the minimal amount of resources. In an *overloaded* system, i.e.,

$\sum_{i=1}^{n} \alpha_i^* > 1$, by definition, there cannot exist a way of allocating network resources such that all users' QoS requirements are satisfied.

We turn our focus to characterizing the case when $\mathcal{A}^*$ is nonempty under optimal aggregate-flow per-hop control. The next result is the only general result that holds from property (B) without exploiting further features of optimal aggregate-flow classifier solutions.

Proposition 3.1.26 (Diagonal Inclusion) *Let $\mathcal{D} = \{\boldsymbol{\eta} : \eta_i = \eta_j$ for all $i, j \in [1,n]\}$. For all per-hop control $\Phi_{m,n}$ satisfying (B), if $\alpha_i^* \leq \lambda_i / \sum_{j=1}^{n} \lambda_j$ for all users $i \in [1,n]$, then $\mathcal{D} \subseteq \mathcal{A}^*$.*

Proof. For any $\boldsymbol{\eta} \in \mathcal{D}$, $\eta_i = \eta_j$, for all $i,\ j \in [1,n]$. By property (B), $\eta_i = \eta_j$ implies $\omega_i = \omega_j$. We have $\frac{\alpha_1}{\lambda_1} = \frac{\alpha_2}{\lambda_2} = \cdots = \frac{\alpha_n}{\lambda_n}$, and $\sum_{i=1}^{n} \alpha_i = 1$. Thus $\alpha_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$, which means $\alpha_i \geq \alpha_i^*$, $i \in [1,n]$. Therefore, $\eta \in \mathcal{A}$ and $\mathcal{D} \subseteq \mathcal{A}$. ■

Note that $\alpha_i^* \leq \lambda_i / \sum_{j=1}^{n} \lambda_j$ for all $i \in [1,n]$ implies that $\sum_{i \in [1,n]} \alpha_i^* \leq 1$. Next, we find weaker conditions for $\mathcal{A}^* \neq \emptyset$, and characterize the loss of power resulting from having a bounded, discrete label set $\{1, 2, \ldots, L\}$. To achieve this, we utilize the properties of the optimal aggregate-flow classifier solution for $L = m$. First, consider the case when $\eta_i \in \mathbb{R}_+$ for all $i \in [1,n]$, and $n = m$. The case of interest, $\boldsymbol{\eta} \in [1,L]^n$ in the aggregate-flow case can be analyzed by relating it to the unrestricted case.

Theorem 3.1.27 (Unrestricted Intersection) *Assume $\eta_i \in \mathbb{R}_+$ for all $i \in [1,n]$. Let $n = m$, and let ξ be the optimal per-flow classifier. Then $\mathcal{A}^* \neq \emptyset$ if, and only if,*

(a) *$\exists\, i \in [1,n]$ such that $\alpha_i^* \leq \nu \lambda_i / \sum_{j=1}^{n} \lambda_j$, and*

(b) *$\sum_{j=1}^{n} \max\{\alpha_i^*,\ \nu \lambda_i / \sum_{j=1}^{n} \lambda_j\} \leq 1$.*

Proof. (i) First we show that $\mathcal{A} \neq \emptyset$ when both (a) and (b) are satisfied. Given $\boldsymbol{\alpha}^* = (\alpha_1^*, \cdots, \alpha_n^*)$ satisfying (a) and (b), we will construct an $\boldsymbol{\eta}$ such that $\varphi^i(\boldsymbol{\eta}) \geq \alpha_i^*$, $i \in [1,n]$. Let $\alpha_i' = \max\{\alpha_i^*,\ \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}\}$, $i \in [1,n]$. Because of (b), $\sum_{i=1}^{n} \alpha_i' \leq 1$. We construct $\boldsymbol{\eta}$ as $\eta_i = \frac{\alpha_i'}{\lambda_i} - \frac{\nu}{\sum_{j=1}^{n} \lambda_j}$, $i \in [1,n]$. $\eta_i \geq 0$ since $\alpha_i' \geq \nu \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$, $i \in [1,n]$.

We need to check whether $\varphi^i(\boldsymbol{\eta}) \geq \alpha_i^*$, $i \in [1,n]$. Because of (a), we get $\eta_{min} = 0$. Hence $\hat{\eta}_i = \frac{\eta_i}{\eta_{max}}$. Using (3.1.5), we have:

$$\begin{aligned}
\alpha_i &= (1-\nu)\frac{\lambda_i\hat{\eta}_i}{\sum_{j=1}^n \lambda_j\hat{\eta}_j} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \\
&= (1-\nu)\frac{\frac{1}{\eta_{max}}\lambda_i(\frac{\alpha_i'}{\lambda_i} - \frac{\nu}{\sum_{j=1}^n \lambda_j})}{\frac{1}{\eta_{max}}\sum_{j=1}^n \lambda_j(\frac{\alpha_j'}{\lambda_j} - \frac{\nu}{\sum_{l=1}^n \lambda_l})} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \\
&= (1-\nu)\frac{\alpha_i' - \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}}{\sum_{j=1}^n \alpha_j' - \nu} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \\
&\geq (1-\nu)\frac{\alpha_i' - \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}}{1-\nu} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \\
&= \alpha_i'.
\end{aligned}$$

(ii) Second we show that $\mathcal{A} = \emptyset$ when either (a) or (b) are violated.

(a) Suppose $\forall i \in [1,n]$, $\alpha_i^* > \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$. Pick any $\boldsymbol{\eta} \in \mathbb{R}_+^n$, we have $\hat{\eta}_{min} = 0$. Suppose $\hat{\eta}_k = \hat{\eta}_{min}$, then according to (3.1.5), $\alpha_k = \nu\frac{\lambda_k}{\sum_{j=1}^n \lambda_j}$ and $\alpha_k < \alpha_k^*$. Therefore $\mathcal{A} = \emptyset$.

(b) Suppose $\sum_{i=1}^n \max\{\alpha_i^*,\ \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}\} > 1$. We can show $\mathcal{A} = \emptyset$ by contradiction. Suppose $\mathcal{A} \neq \emptyset$, then $\exists\boldsymbol{\eta}$, $\alpha_i = \varphi^i(\boldsymbol{\eta}) \geq \alpha_i^*$ for $i \in [1,n]$. By (3.1.5), $\alpha_i \geq \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$ for $i \in [1,n]$. Therefore, $\alpha_i \geq \max\{\alpha_i^*,\ \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}\}$. Because $\sum_{i=1}^n \alpha_i = 1$, we get $\sum_{i=1}^n \max\{\alpha_i^*,\ \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j}\} \leq 1$, which is contradictory to our hypothesis. ■

Here $\nu \geq 0$ is the solution parameter of the optimal per-flow classifier which determines how much proportional sharing to inject in the service weight allocation ($\nu = 1$ degenerates per-hop control to FIFO). Theorem 3.1.27 is a tight characterization of $\mathcal{A}^*$'s nonemptiness in the unrestricted case where properties (a) and (b) stem from the particular form of the optimal per-flow classifier solution given by Proposition 3.1.4. Note that as $\nu \to 0$, (b) becomes $\sum_{j=1}^n \alpha_i^* \leq 1$ which is the weakest possible condition for nonemptiness of $\mathcal{A}^*$. The next result is an immediate consequence of Theorem 3.1.27.

Corollary 3.1.28 (Empty Restricted Intersection) *If $\mathcal{A}^* = \emptyset$ in the unrestricted case, then $\mathcal{A}^* = \emptyset$ in the restricted case where $\eta_i \in \{1, 2, \ldots, L\}$ for all $i \in [1,n]$, and $L < \infty$.*

The aggregate-flow and per-flow cases with respect to nonemptiness of $\mathcal{A}^*$ can be related by the next result which is a consequence of Theorem 3.1.20.

Proposition 3.1.29 (Per-flow and Aggregate-flow Relation) *Let* $\eta_i \in \{1, 2, \ldots, L\}$ *for all* $i \in [1, n]$, *and* $L < \infty$. $\mathcal{A}^* \neq \emptyset$ *for optimal per-flow per-hop control (i.e.,* $n = m$*) if, and only if,* $\mathcal{A}^* \neq \emptyset$ *for optimal aggregate-flow per-hop control with* $m = L$.

Proof. By the proof of Proposition 3.1.9, the optimal aggregate-flow per-hop control with $m = L$ is equivalent to the optimal per-flow per-hop control. ■

Given the relationship of nonemptiness of $\mathcal{A}^*$ between the per-flow and aggregate-flow case under $\eta_i \in \{1, 2, \ldots, L\}$, what remains is a quantitative characterization of the loss of power due to discreteness and boundedness of the label set $[1, L]$ in the aggregate-flow case.

Theorem 3.1.30 (Loss of Power due to Restriction) *Let* $L = m < n$. *For optimal aggregate-flow per-hop control* $\Phi_{L,n}$, *if there exists* $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ *with* $\gamma_{min} = 0$, $\gamma_{max} = 1$, $0 \leq \gamma_i \leq 1$, *such that*

$$(1-\nu)\,\frac{\lambda_i\gamma_i}{\sum_{j=1}^n \lambda_j\gamma_j} + \nu\,\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \geq \alpha_i^* + \frac{1-\nu}{L-1}\frac{\lambda_i}{\sum_{j=1}^n \lambda_j\gamma_j} \tag{3.1.31}$$

for all $i \in [1, n]$, *then* $\mathcal{A}^* \neq \emptyset$.

Proof. Given user's resource requirement $\boldsymbol{\alpha^*} = (\alpha_1^*, \cdots, \alpha_n^*)$, Let $\Phi_{L,n}$ be an $m = L$ optimal aggregate-flow per-hop control with classifier ξ and weight vector $\tilde{\boldsymbol{\alpha}} = (\alpha^1, \cdots, \alpha^L)$. If there exists $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_n)$ with $\gamma_{\min} = 0$, $\gamma_{\max} = 1$, $0 \leq \gamma_i \leq 1$ that satisfies (3.1.31), then we have

$$(1-\nu)\,\frac{\lambda_i(\gamma_i - \frac{1}{L-1})}{\sum_{j=1}^n \lambda_j\gamma_j} + \nu\,\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \geq \alpha_i^*\,, \qquad i \in [1, n].$$

Without loss of generality, suppose $\gamma_1 = \gamma_{min} = 0$, $\gamma_n = \gamma_{max} = 1$. Define

$$D = \{\hat{\boldsymbol{\eta}} : \hat{\eta}_1 = 0,\ \gamma_i - \frac{1}{L-1} \leq \hat{\eta}_i \leq \gamma_i \text{ for } i \in [2, n-1],\ \hat{\eta}_n = 1\}.$$

Then for all $\hat{\boldsymbol{\eta}} \in D$, we have

$$(1-\nu)\frac{\lambda_i\hat{\eta}_i}{\sum_{j=1}^n \lambda_j\hat{\eta}_j} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \geq (1-\nu)\frac{\lambda_i(\gamma_i - \frac{1}{L-1})}{\sum_{j=1}^n \lambda_j\gamma_j} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \geq \alpha_i^*.$$

Construct $\boldsymbol{\eta}^* = (\eta_1^*, \cdots, \eta_n^*)$ as follows:

$$\eta_i^* = \lfloor (L-1)\gamma_i + 1 \rfloor, \qquad i \in [1, n].$$

Then $\hat{\boldsymbol{\eta}}^* \in D$. Hence $\varphi^i(\boldsymbol{\eta^*}) \geq \alpha_i^*$, $i \in [1, n]$. Therefore, $\mathcal{A} \neq \emptyset$. ■

The left-hand-side of inequality (3.1.31) just denotes a valid service weight vector with respect to the optimal aggregate-flow classifier. The second term in the right-hand-side of (3.1.31) of Theorem 3.1.30 quantifies the loss of power due to coarseness. If $L \to \infty$, then the loss-of-power term drops out. In practice, L is a small finite value (e.g., using 4 bits in the precedence field of IP, $L = 16$). The next result shows that $n \gg L$—the raison d'etre of aggregate-flow control—facilitates tightness of the bound.

Corollary 3.1.32 (Nonempty Discrete Intersection) *Under the same conditions as* Theorem 3.1.30, *let* $d_i = \lfloor (L-1)\gamma_i \rfloor$, $i \in [1, n]$. *Then,* $\mathcal{A}^* \neq \emptyset$ *if for all* $i \in [1, n]$,

$$(1-\nu)\frac{\lambda_i\gamma_i}{\sum_{j=1}^n \lambda_j\gamma_j} + \nu\frac{\lambda_i}{\sum_{j=1}^n \lambda_j} \geq \alpha_i^* + (1-\nu)\frac{\lambda_i}{\sum_{k=1}^{L-1} k\sum_{j:d_j=k}\lambda_j}. \tag{3.1.33}$$

Proof. We have

$$\sum_{k=1}^{L-1} k\sum_{j:d_j=k}\lambda_j = \sum_{j=1}^n d_j\lambda_j \leq (L-1)\sum_{j=1}^n \lambda_j\gamma_j.$$

Hence,

$$(1-\nu)\frac{\lambda_i}{\sum_{k=1}^{L-1} k\sum_{j:d_j=k}\lambda_j} \geq \frac{1-\nu}{L-1}\frac{\lambda_i}{\sum_{j=1}^n \lambda_j\gamma_j}.$$

Therefore, if (3.1.33) holds, then (3.1.31) holds and $\mathcal{A}^* \neq \emptyset$. ■

For $n \gg L$, we can expect $\frac{\lambda_i}{\sum_{k=1}^{L-1} k\sum_{j:d_j=k}\lambda_j} \ll 1$, and (3.1.31) gives a tight bound on the existence condition of system optimal configurations.

3.2 Game Theoretic Structure

The roadmap of the game theoretic results is as follows. First, we derive stability properties—existence of Nash equilibria and their structure—and dynamics of the noncooperative QoS provision game when users are allowed to set their η values end-to-end. Second, we show efficiency properties with respect to system optimality, in particular, when Nash equilibria are system optimal.

3.2.1 Basic Definitions

In Section 3.1.4, we use α_i^* to denote the minimal bandwidth needed to satisfy user i's QoS requirement. We call the pair $(\boldsymbol{\eta}, \boldsymbol{\eta}')$ of control vectors a *selfish move* of user $i \in [1, n]$ with respect to α_i^* if $\boldsymbol{\eta}' = \boldsymbol{\eta} \pm \mathbf{e}_i$, and the following two conditions are satisfied:

(i) $\varphi^i(\boldsymbol{\eta}) < \alpha_i^*$ implies $\boldsymbol{\eta}' = \boldsymbol{\eta} + \mathbf{e}_i$ and $\varphi^i(\boldsymbol{\eta}') > \varphi^i(\boldsymbol{\eta})$;

(ii) $\varphi^i(\boldsymbol{\eta}) > \alpha_i^*$ implies $\boldsymbol{\eta}' = \boldsymbol{\eta} - \mathbf{e}_i$ and $\alpha_i^* \leq \varphi^i(\boldsymbol{\eta}') < \varphi^i(\boldsymbol{\eta})$.

Thus an "unhappy" user tries to improve his happiness by increasing η_i, while an "overly" satisfied user tries to reduce the satisfaction level to match his actual needs.

We call a pair of control vectors $(\boldsymbol{\eta}, \boldsymbol{\eta}')$ a *concurrent selfish move* (in the negative direction) if for some $J \subseteq [1, n]$, $\boldsymbol{\eta} = \boldsymbol{\eta} - \sum_{i \in J} \mathbf{e}_i$, and $(\boldsymbol{\eta}, \boldsymbol{\eta} - \mathbf{e}_i)$ is a selfish move for all $i \in J$. An analogous definitions holds for concurrent selfish moves in the *positive direction*. We will sometimes refer to selfish moves as *sequential* selfish moves to distinguish from concurrent ones. The definition of selfish move describes an efficient or cost conscious user who only consumes just enough resources to satisfy her QoS needs.

For user i, let $\mathcal{A}_i = \{\boldsymbol{\eta} : \varphi^i(\boldsymbol{\eta}) \geq \alpha_i^*\}$. Thus $\mathcal{A}_i$ represents the set of configuration where user i's QoS requirement is satisfied. We have

$$\mathcal{A}^* = \bigcap_{i=1}^{n} \mathcal{A}_i,$$

since all users' QoS requirements are satisfied at $\boldsymbol{\eta} \in \mathcal{A}^*$. $\boldsymbol{\eta} \in \mathcal{A}^*$ is a *corner point* of $\mathcal{A}^*$ if the set of selfish moves from $\boldsymbol{\eta}$ is empty.

3.2.2 Nash Equilibria and Stability Properties

We present the dynamical properties of the noncooperative QoS provision game when $\mathcal{A}^*$ exists (i.e., is nonempty) and $\boldsymbol{\eta} \in \mathcal{A}^*$.

Proposition 3.2.1 (Projection) *For user i and configuration $\boldsymbol{\eta} \in \mathcal{A}_i$, let $\mathcal{M}_i(\boldsymbol{\eta}) = \{\boldsymbol{\eta}' : \eta'_i = \eta_i, \text{ and } \eta'_j \leq \eta_j \text{ for } j \neq i\}$. Then $\mathcal{M}_i(\boldsymbol{\eta}) \subseteq \mathcal{A}_i$.*

Proof. Let $\boldsymbol{\eta}' \in \mathcal{M}_i(\boldsymbol{\eta})$. By property (A2), $\boldsymbol{\eta}^1 = (\eta'_1, \eta_2, \ldots, \eta_i, \ldots, \eta_n) \in \mathcal{A}_i$. By a second application of (A2) to $\boldsymbol{\eta}^1$, $\boldsymbol{\eta}^2 = (\eta'_1, \eta'_2, \eta_3, \ldots, \eta_i, \ldots, \eta_n) \in \mathcal{A}_i$. By a repeated application of (A2), the procedure eventually yields $\boldsymbol{\eta}'$ where membership in $\mathcal{A}_i$ is preserved at every step. ■

Proposition 3.2.1 is a consequence of property (A2) of the per-hop control. We can use Proposition 3.2.1 and property (A1) to show a closure property of $\mathcal{A}^*$.

Lemma 3.2.2 (Closure) *$\mathcal{A}^*$ is closed under selfish moves, sequential and concurrent. That is, for $\boldsymbol{\eta} \in \mathcal{A}^*$ and any subset of users $J \subseteq [1, n]$ such that $(\boldsymbol{\eta}, \boldsymbol{\eta} - \mathbf{e}_i)$ is a selfish move for all $i \in J$,*

$$\boldsymbol{\eta} - \sum_{i \in J} \mathbf{e}_i \in \mathcal{A}^*.$$

Proof. First, note that closure with respect to sequential selfish moves is a special case of closure with respect to concurrent selfish moves. We will show the lemma for the sequential case and then use it to prove the general case.

(i) *Sequential case.* Let i be such that $\boldsymbol{\eta} - \mathbf{e}_i$ or $\boldsymbol{\eta} + \mathbf{e}_i \notin \mathcal{A}^*$. By property (A1),

$$\varphi^i(\boldsymbol{\eta} - \mathbf{e}_i) \leq \varphi^i(\boldsymbol{\eta}),$$

hence this contracts $(\boldsymbol{\eta}, \boldsymbol{\eta} - \mathbf{e}_i)$ being a selfish move. A similar argument holds for $\boldsymbol{\eta} + \mathbf{e}_i$.

(ii) *Concurrent case*. Let $\boldsymbol{\eta}' = \boldsymbol{\eta} - \sum_{i \in J} \mathbf{e}_i \in \mathcal{A}^*$. For any $i \in [1, n]$, we will show that $\boldsymbol{\eta}' \in \mathcal{A}_i$. First, assume $i \notin J$. By $\boldsymbol{\eta} \in \mathcal{A}_i$ and Proposition 3.2.1, $\mathcal{M}_i(\boldsymbol{\eta}) \subseteq \mathcal{A}_i$. But by the definition of concurrent selfish move $\boldsymbol{\eta}' \in \mathcal{M}_i(\boldsymbol{\eta})$. which proves the first case. Assume $i \in J$. By Proposition 3.2.1, $\mathcal{M}_i(\boldsymbol{\eta} - \mathbf{e}_i) \subseteq \mathcal{A}_i$. Since $\boldsymbol{\eta}' \in \mathcal{A}_i$, this completes the proof. ■

Thus selfish users, even when making simultaneous selfish changes to their η values, cannot escape from the set $\mathcal{A}^*$ where their QoS requirements are all satisfied, some more than necessary.

A concurrent selfish move, with respect to users in $J \subseteq [1, n]$ and intersection set $\bigcap_{i \in J} \mathcal{A}_i$, can be represented by a subset of $J' \subseteq J$ that shows the users making a move since selfish moves within $\bigcap_{i \in J} \mathcal{A}_i$ can only occur in the downward direction.

Theorem 3.2.3 (Monotone Convergence) *Any initial configuration $\boldsymbol{\eta} \in \mathcal{A}^*$ converges to a corner point of $\mathcal{A}^*$ under selfish moves, sequential or concurrent.*

Proof. Consider $\boldsymbol{\eta}' = \boldsymbol{\eta} - \sum_{i \in J} \mathbf{e}_i$ where $J \subseteq [1, n]$ represents a concurrent selfish move. By definition of selfish move, we have $\eta'_i < \eta_i$ for all $i \in J$. By closure of $\mathcal{A}^*$ (Lemma 3.2.2) and boundedness of η'_i by α^*_i, after a finite number of applications of concurrent selfish moves, no selfish move will be possible. The configuration reached, by definition, is a corner point of $\mathcal{A}^*$. ■

Thus a corner point of $\mathcal{A}^*$ is a fixed point under the dynamics of selfish moves within $\mathcal{A}^*$, from which users cannot escape by selfish actions due to closure. Theorem 3.2.3 also shows that $\mathcal{A}^*$ always possesses a corner point, not necessarily unique.

A corner point $\boldsymbol{\eta}$ represents an *efficient* allocation of resources for all users in the sense that each user i's QoS requirement is satisfied by $\boldsymbol{\eta}$, i.e., $\alpha^*_i = \varphi^i(\boldsymbol{\eta}) \geq \alpha^*_i$. Furthermore, any incremental action by i will either violate his QoS requirement or increase the apportioned resources beyond what is needed to satisfy the user's QoS requirement. We will show that a nonincremental action by user i will have the same consequences (Theorem 3.2.4). If $\varphi^i(\boldsymbol{\eta}) = \alpha^*_i$ then $\boldsymbol{\eta}$ is efficient in an absolute sense.

Theorem 3.2.4 (Corner Point and Nash) *Let $\boldsymbol{\eta}$ be a corner point of $\mathcal{A}^*$. Then $\boldsymbol{\eta}$ is a Nash equilibrium.*

Proof. Pick any user $i \in [1, n]$. Since $\boldsymbol{\eta}$ is a corner point, $\varphi^i(\boldsymbol{\eta} - \mathbf{e}_i) < \alpha_i^*$. By property (A1), $\varphi^i(\boldsymbol{\eta} - c\,\mathbf{e}_i) \leq \varphi^i(\boldsymbol{\eta} - \mathbf{e}_i)$ for $c > 1$, and part two of condition (i) is satisfied. Part one of the same condition is satisfied by (A1). ■

We remark that a corner point of $\mathcal{A}^*$ must be Nash equilibrium, but the converse need not be true. Indeed, there are Nash equilibria that need not be in $\mathcal{A}^*$, even when it is nonempty.

Theorem 3.2.5 (Nash and System Optimality) *A configuration $\boldsymbol{\eta}$ is Nash and system optimal if, and only if, $\boldsymbol{\eta}$ is a corner point of $\mathcal{A}^*$.*

3.3 Conclusion

In this chapter, we provide a general theoretical framework of scalable QoS provisioning using aggregate-flow scheduling where packet labels can be set from a finite label set and routers provide differentiated treatment of packets based on the labels enscribed. We define the meaning of optimal per-hop control within this context and find the optimal solution for aggregate-flow control. We show that the optimal per-hop control satisfies certain properties—denoted (A1), (A2), and (B)—which relate how label values impact the service a flow receives at a router. We augment the general result by presenting optimal solutions when restricting the packet scheduling disciplines to variants of GPS, and the consequences on the core properties.

In Section 3.2, we expand the framework by considering selfish users who can influence QoS provisioning behavior by regulating the label values assigned to their traffic streams. Based on the properties exported by the network control—(A1), (A2), and (B)—we show how a population of selfish users with diverse QoS requirements setting their packet labels can arrive at a global allocation of resources that is stable (Nash equilibrium) and efficient (system optimal). We show that even in situations when

network resources are scarce such that no resource allocation—aggregate-flow differentiation, per-flow reservation, or otherwise—can satisfy all users' QoS requirements, the system reaches a Nash equilibrium. We show that the optimal per-hop control is also "optimal" in the noncooperative game context in the sense that when network resources are configurable such that all users' QoS requirements can be satisfied, then there exists a Nash equilibrium that is system optimal.

4 PERFORMANCE EVALUATION

In this chapter, we extend the theoretical work in Chapter 3 by investigating implementation and performance evaluation issues associated with the induced optimal aggregate-flow scheduling. 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 carry out a comprehensive performance evaluation study of QoS provisioning using optimal aggregate-flow scheduling. The results in this chapter are also published in [70, 72].

4.1 QoS Provisioning Architecture Design

4.1.1 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 2.3) where L is the total number of label values. For each incoming packet, the optimal aggregate-flow per-hop control performs 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 4.2):

  - 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. [1]

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

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 $k \in \{1, \ldots, L\}$.

In Section 4.2.2, 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.

[1] 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.

4.1.2 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)).

As described in Chapter 2, 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

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

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

$$\frac{d\eta_i}{dt} = (-1)^a \varepsilon \, ||\mathbf{x}^i(\boldsymbol{\eta}) - \boldsymbol{\theta}^i|| \tag{4.1.1}$$

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 4.2), 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 (4.1.1).

4.1.3 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 4.1.1 with $m = 16$ service classes as link bandwidth is varied[2]. 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.

[2]The performance results are taken from Section 4.2.

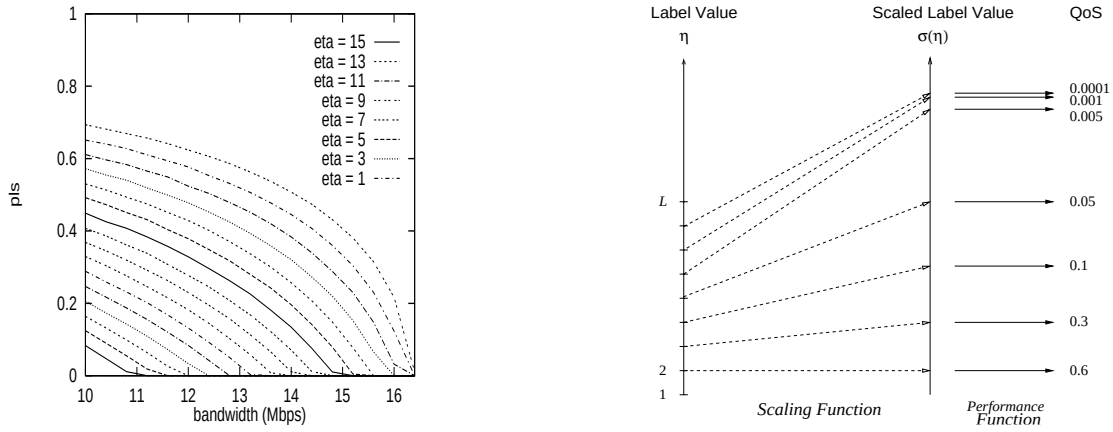

Figure 4.2. Left: "Equal spacing" QoS separation achieved by optimal aggregate-flow classifier when $L = 16$. Right: Scaling function σ affecting nonuniform stretching and contraction.

To solve the potential inefficiency and structural limitation pointed out above, we introduce a 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 depicted in Figure 4.3.

The performance gain brought by the scaling function is demonstrated and further discussed in Section 4.2.3.

4.1.4 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

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

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 $k \in \{1, \ldots, L\}$.

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

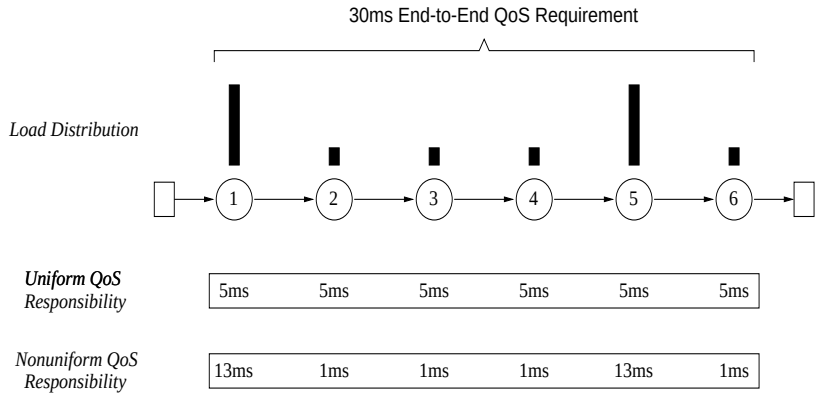

Figure 4.4. Uniform vs. nonuniform local QoS responsibility distribution to satisfy 30ms end-to-end delay requirement for a given load imbalance.

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, in general, efficient—allocation is one

where heavily loaded routers are given less responsibility than lightly loaded ones[3]. 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 4.2.6, we show that both properties are satisfied by the optimal aggregate-flow scheduler.

4.2 Performance Results

4.2.1 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* [11]—which was used in our earlier work on WAN QoS performance evaluation [8, 9]. 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 4.1.

[3]One can formally attribute this property by using congestion pricing and utilization as cost measures [9].

Network Configuration

We use two benchmark network topologies, a 2-switch dumbbell topology shown in Figure 4.5 (left) which isolates a bottleneck link, and a 4-switch caterpillar topology shown in Figure 4.5 (right) which is comprised of multiple bottleneck links.

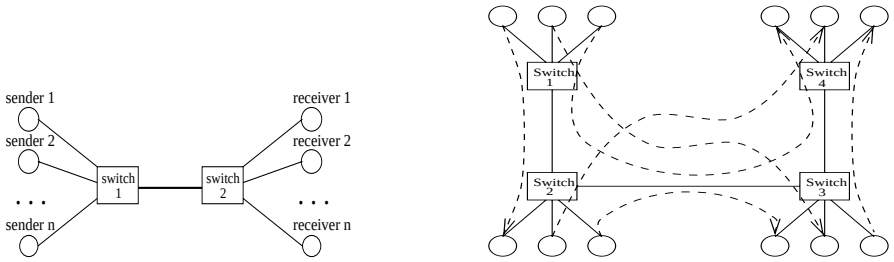

Figure 4.5. Benchmark network topology. Left: 2-switch single bottleneck link shared by n flows. Right: 4-switch multiple bottleneck link caterpillar topology.

Most of the performance results reported in this chapter are from the bottleneck topology shown in Figure 4.5 (left) whose results are extended by those carried out for the topology shown in Figure 4.5 (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 (3.1.2), the η value we used in simulation ranges from 0 to $L-1$.

4.2.2 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.

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 4.6 shows the QoS separation achieved

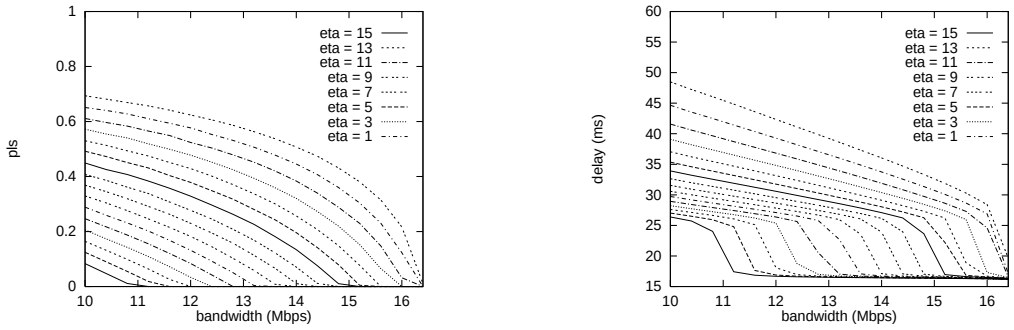

Figure 4.6. QoS separation achieved by optimal aggregate-flow classifier when $L = 16$. Left: Packet loss rate. Right: End-to-end delay.

in the $L = 16$ service classes when the scheduler executes the optimal aggregate-flow service weight assignment. 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 4.6 (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 its 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.

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 4.7 (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 by service class separation which shows that the

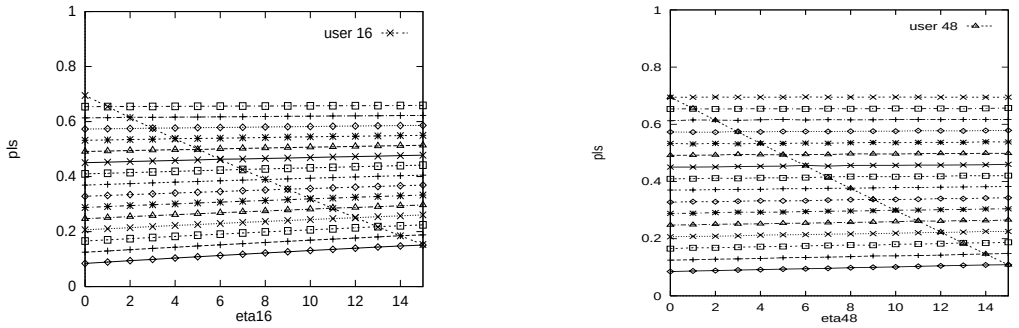

Figure 4.7. Manifestation of properties (A1) and (A2). End-to-end QoS shaping as a function of label value η_{16} of singular user flow. Left: 16 users (originally each group has one user). Right: 48 users (average group population size of 3).

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 4.7 (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.

Impact of Weighting Parameter ν

Recall that the service weight assignment, given by equation (3.1.5), 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 3.1, $\nu \to 1$ diminishes the contribution of the differentiation component (the first term of (3.1.5)) 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 4.8.

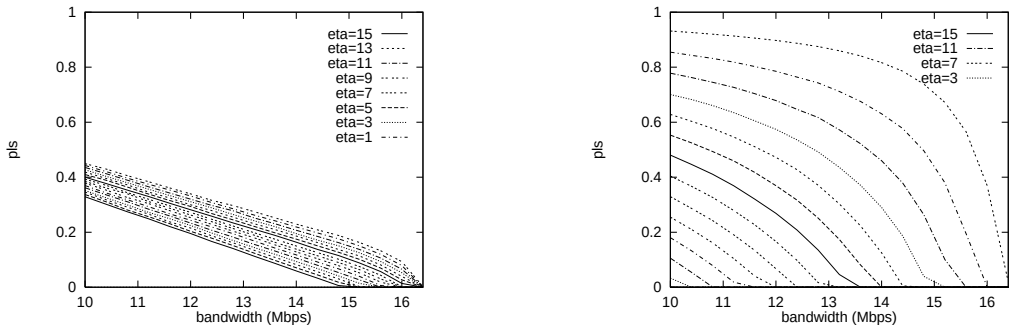

Figure 4.8. Impact of ν on QoS separation for $L = 16$. Left: QoS exported by service classes as a function of bottleneck bandwidth when $\nu = 0.9$. Right: Corresponding plots when $\nu = 0.1$.

The left figure shows QoS separation as a function of bottleneck bandwidth for $L = 16$ when $\nu = 0.9$ which makes the system resemble a FIFO scheduler. Figure 4.8 (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. From a network management per-

spective, ν is a control parameter that the service provider may engage to reflect the overall needs of its customer base and their QoS profile.

4.2.3 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 $\mathcal{A}^*$ (cf. Section 3.2), 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.

Efficiency and Optimality

In this section, we study the structure of $\mathcal{A}^*$. Recall in Section 3.2, we define $\mathcal{A}^*$ as the set of $\boldsymbol{\eta}$ configurations where all user requirements are satisfied. Given a configuration $\boldsymbol{\eta}$, we 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 4.9 (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

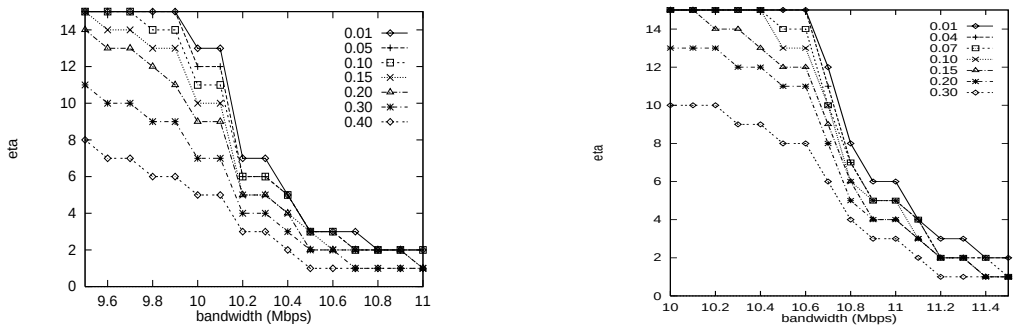

Figure 4.9. Structure of $\mathcal{A}^*$. Left: The change in Nash equilibria as we increase bottleneck bandwidth for user population with QoS requirement profile $(0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 1.0)$. At 10 Mbps, Nash equilibria become corner points of $\mathcal{A}^*$. Right: Corresponding landscape for more stringent user population QoS profile shown in the legend.

10 Mbps, $\mathcal{A}^*$ is empty but Nash equilibria nonetheless exist (cf. Section 3.2). 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 4.9 (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.

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 4.10 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 (3.1.5)) is set to 0.5.

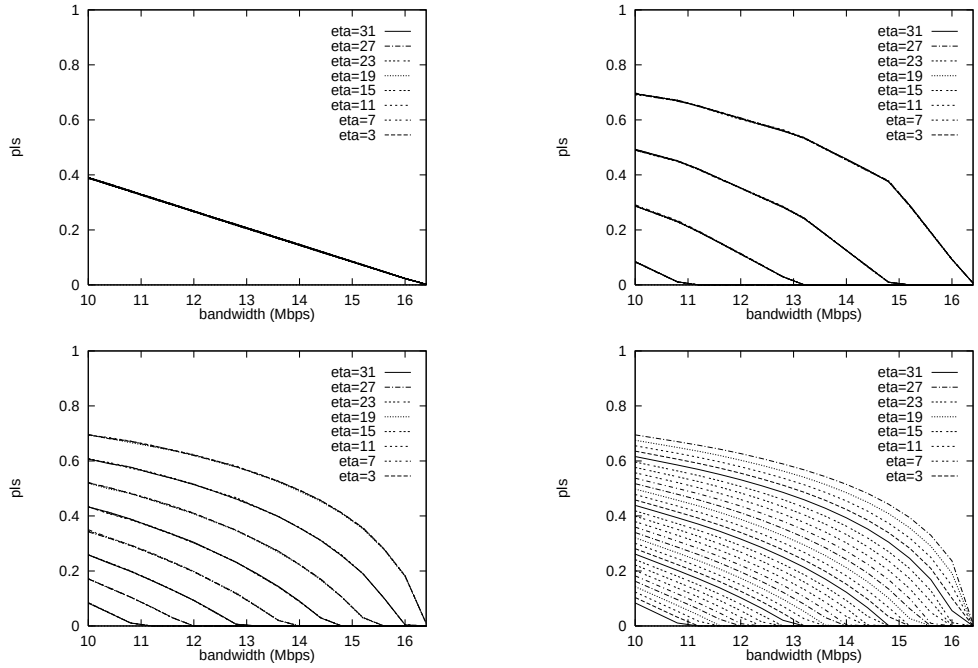

Figure 4.10. Impact of bounded label set size L on QoS exported by the service classes as a function of bottleneck bandwidth for L=1, 4, 8, 32.

Figure 4.10 (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 (3.1.2) (see Section 3.1). Since for all finite $L > 0$, $\eta \in \{1, 2, \ldots, L\}$ is mapped by (3.1.2) 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 (3.1.2), and $\mathbb{Z}_+$ is mapped to a dense subset of $[0, 1]$. Thus L translates to "QoS resolution" in a straightforward fashion.

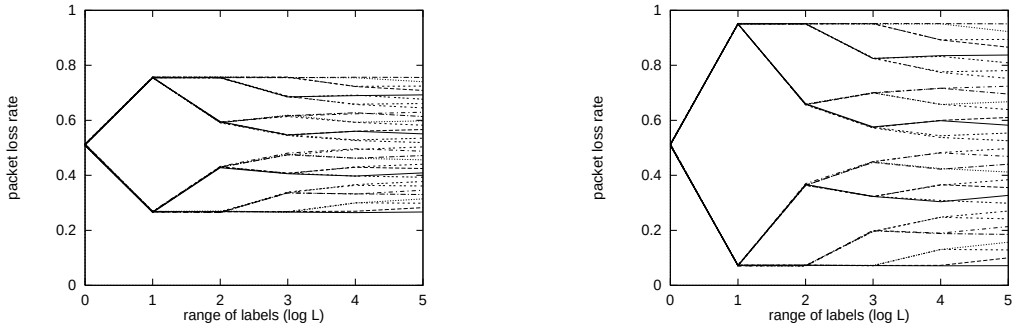

Figure 4.11. The combined impact of L and ν on QoS shaping. Left: QoS exported in L service classes as a function of L for $\nu = 0.5$. Right: Corresponding plot for $\nu = 0.1$.

Figure 4.11 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 4.11 (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, 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 (3.1.2)—can be observed in Figure 4.11.

The size of the label set L also impacts the system efficiency, the existence or nonemptiness of $\mathcal{A}^*$ which is shown in Figure 4.12. The plot shows the minimum bottleneck bandwidth needed to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L. We observe that,

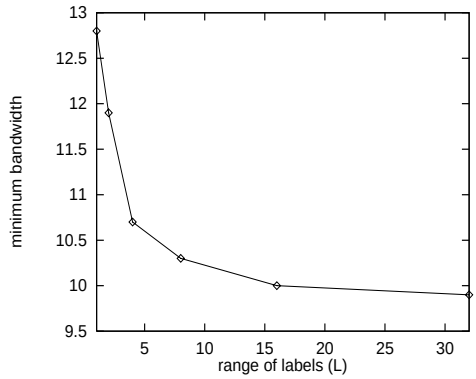

Figure 4.12. Impact of L on existence of $\mathcal{A}^*$: minimum bottleneck bandwidth required to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L.

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.

4.2.4 The Role of Scaling Function

In this section, we study the performance impact of the scaling function. Figure 4.13 shows how the scaling function can affect the QoS differentiation among service classes. Comparing it with Figure 4.6, we see that the scaling function can achieve non-uniform QoS differentiation.

Next we show that the scaling function can improve system efficiency. As in Section 4.2.3, 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 4.14 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 4.12. The bottom

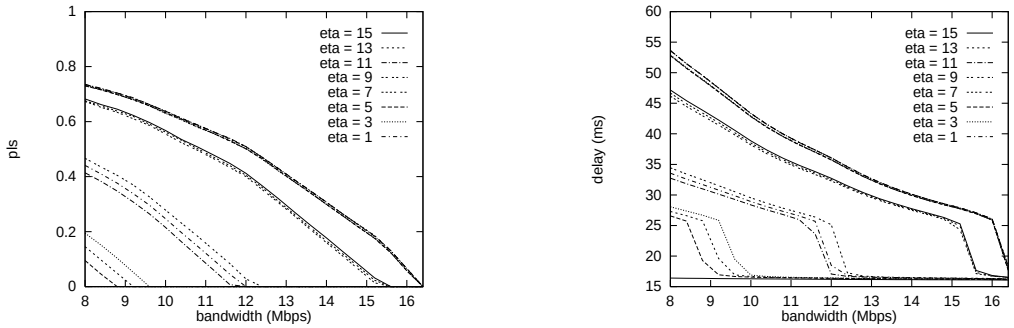

Figure 4.13. QoS differentiation achieved by optimal aggregate-flow classifier with scaling function. $\sigma(\eta)$ for $\eta \in [0, 15]$: 1.0, 1.1, 1.2, 10, 11, 12, 100, 110, 120, 500, 550, 600, 1000, 1100, 1200, 2000. Left: Packet loss rate. Right: End-to-end delay.

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.

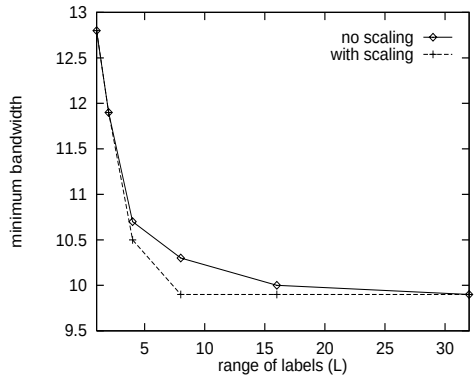

Figure 4.14. Impact of scaling function on system efficiency: minimum bottleneck bandwidth required to achieve $\mathcal{A}^* \neq \emptyset$ as a function of L.

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 because the system is FIFO when $L = 1$, and we always get $\widehat{\sigma(\eta)} = 0, 1$ after the transformation (3.1.2) 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 $L > 8$.

4.2.5 Impact of Burstiness

The previous sections have shown the structural QoS provisioning 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 traffic control and QoS provisioning—and show that the qualitative behavior of the aggregate-flow QoS provisioning remains the same. Figure 4.15 shows QoS separation results across $m = 16$ aggregate-flow service classes for the same network configuration as Figure 4.6 (see Section 4.2.2) except for the difference that the traffic sources are VBR—Poisson—with the same average data rates.

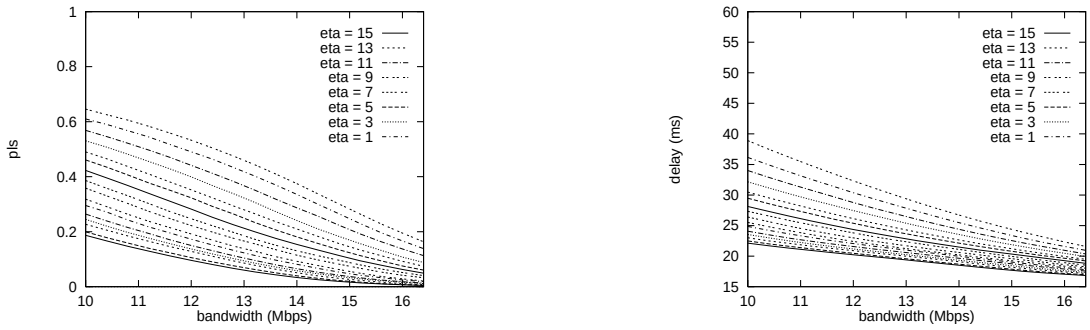

Figure 4.15. QoS separation achieved by optimal aggregate-flow classifier when $L = 16$ under VBR traffic. Left: packet loss rate. Right: end-to-end delay.

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.

Figure 4.16 shows the impact of bounded, discrete label set size L under VBR traffic which corresponds to the set-up in Figure 4.10 (Section 4.2.3) for CBR traffic. As with the CBR traffic case, we observe an increased power of QoS resolution as L is

increased (only shown for $L = 4$ and 16 due to space constraints). The principal qualitative difference, as with Figure 4.15, is the more gradual change of the performance curve which is affected by burstiness. In essence, burstiness does not add additional complications to QoS provisioning above and beyond its "usual" impact with respect to queuing, multiplexing gain, and so forth, which can be immensely subtle—in their own right—depending on the traffic properties.

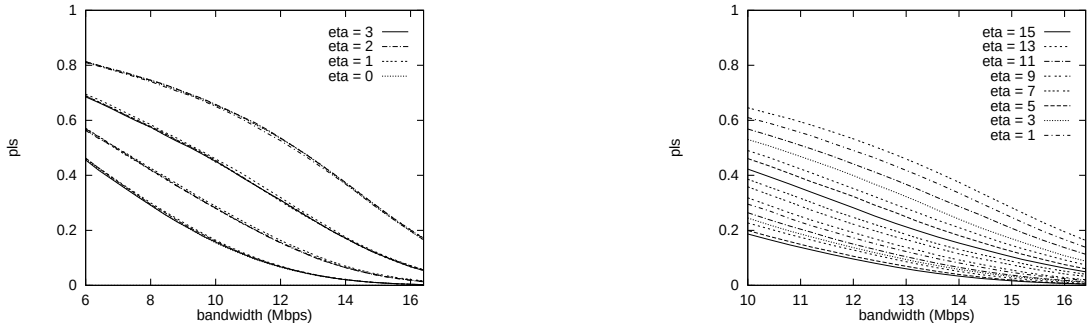

Figure 4.16. Impact of finite label set size L on QoS for VBR traffic. Left: L=4; Right: L=16.

Figure 4.17 shows QoS separation results across $m = 16$ aggregate-flow service classes with scaling function under VBR traffic. The network configuration in Figure 4.17 is same as in Figure 4.13 except for the difference that the traffic sources are VBR—Poisson—with the same average data rates. Both the packet loss (left) and delay (right) 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.

4.2.6 Dynamics and Convergence

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

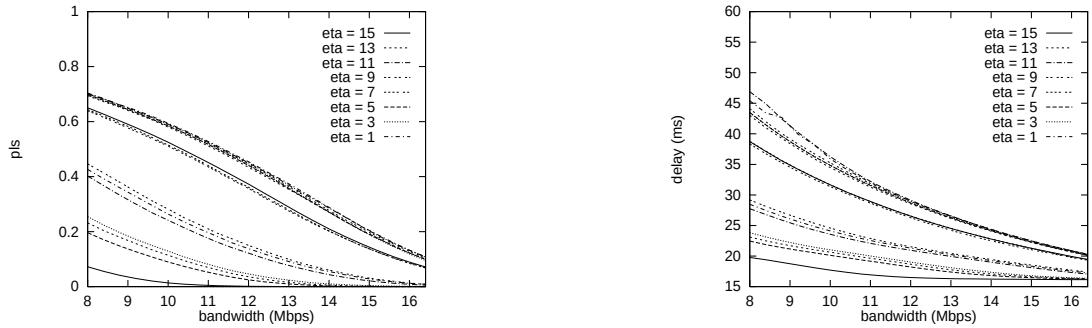

Figure 4.17. QoS separation achieved by optimal aggregate-flow classifier with scaling function when $L = 16$ under VBR traffic. $\sigma(\eta)$, $\eta \in [0, 15]$: 1.0, 1.1, 1.2, 10, 11, 12, 100, 110, 120, 500, 550, 600, 1000, 1100, 1200, 2000. Left: packet loss rate. Right: end-to-end delay.

time evolutions. The QoS exported by the service classes and the end-to-end QoS received by individual user flows is commensurately stable.

Figure 4.18 (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 4.18 (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.

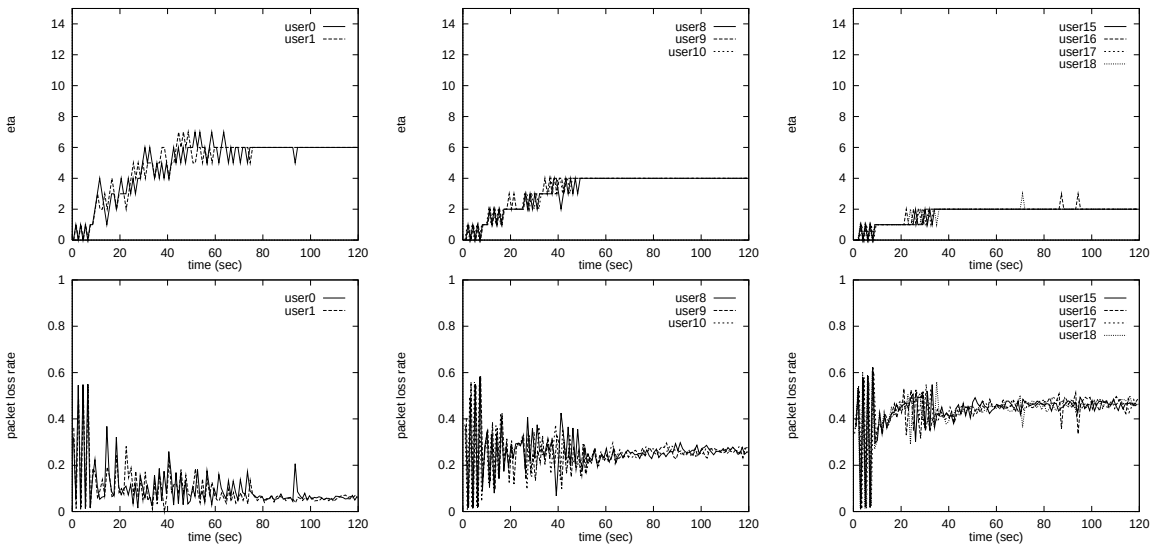

Figure 4.18. Time evolution of adaptive label control and end-to-end QoS. Top row: Evolution of label values shown for three user groups with common QoS requirements (0.1, 0.3, and 0.5). Bottom row: Corresponding trace of measured end-to-end QoS for user flows belonging to the three QoS groups.

Stringency of QoS Requirement

Figure 4.19 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).

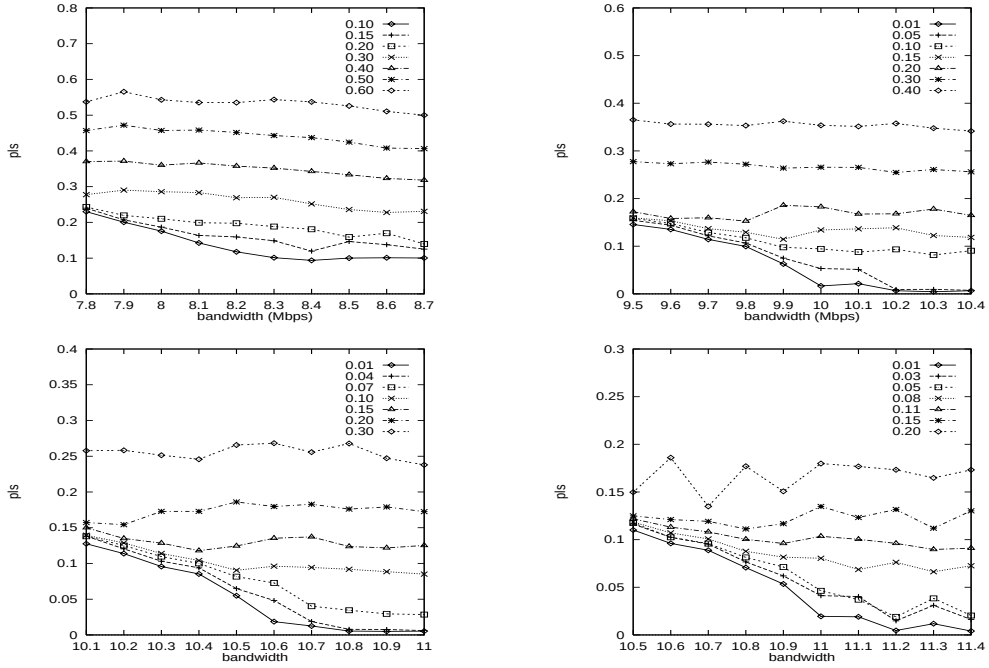

Figure 4.19. End-to-end QoS achieved under adaptive label control as a function of bottleneck bandwidth for successively more stringent QoS requirement profiles (shown in the legends).

Time Evolution in Many-switch Topology

We also have performance results for the many-switch system (shown in Figure 4.5) when adaptive label control is used. Figure 4.20 shows the QoS provisioning dynamics of the 4-router benchmark internetwork shown in Figure 4.5 (right). The number of service classes is set at $L = 16$, and there are a total of 175 user flows possessing 8 sets of QoS requirements including one best-effort application type (the trivial QoS upper bound). The QoS requirements are: 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, and best-effort. Figure 4.20 (top) shows the label value dynamics and convergence for a subset of three user groups, and Figure 4.20 (bottom) shows the corresponding end-to-end QoS achieved. The qualitative dynamics are analogous to the previous

benchmark results with one noticeable *quantitative* performance difference being the longer transient period required for convergence. This is, in part, due to the larger number of flows—more inter-flow interactions which can impede convergence—and the increased round-trip time (RTT) of the feedback loop.

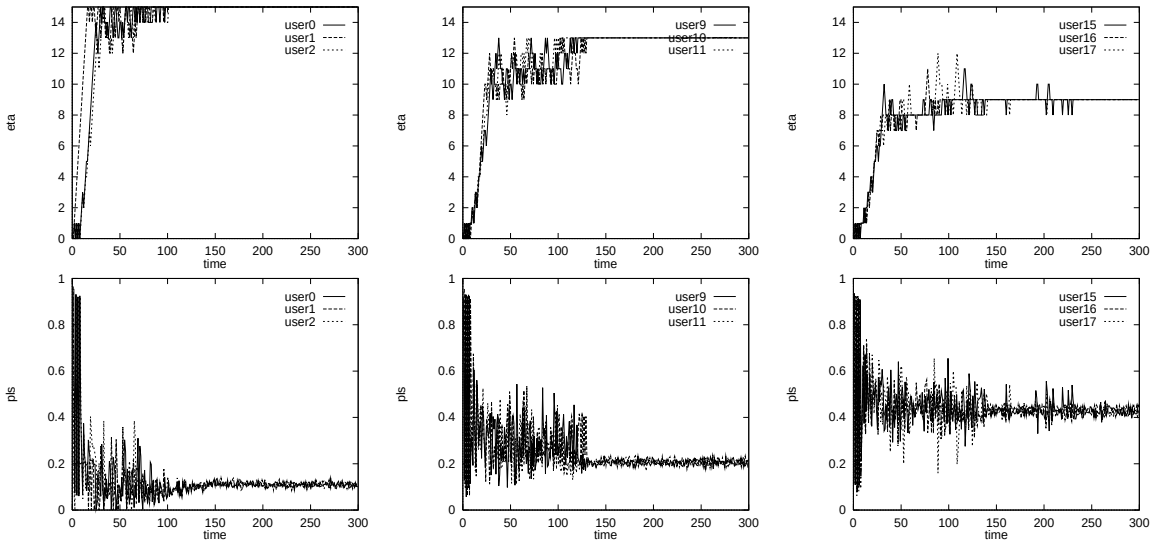

Figure 4.20. Time evolution of adaptive label control and end-to-end QoS in many-switch topology (Figure 4.5 (right)) with many flows. Top row: Evolution of label values shown for three user groups with common QoS requirements (0.1, 0.3, and 0.5). Bottom row: Corresponding trace of measured end-to-end QoS for user flows belonging to the three QoS groups.

In addition to the increased noise factor and time lag influence (in general, functional or delay-differential equations used in the analysis of feedback congestion control can cause oscillatory behavior [65]), another pertinent factor is the inter-flow interaction due to property (A2). The latter can make one user flow's label control action adversely affect another flow's end-to-end QoS thus increasing the time needed to quiescence.

Load Imbalance

In Section 4.1.4, 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.

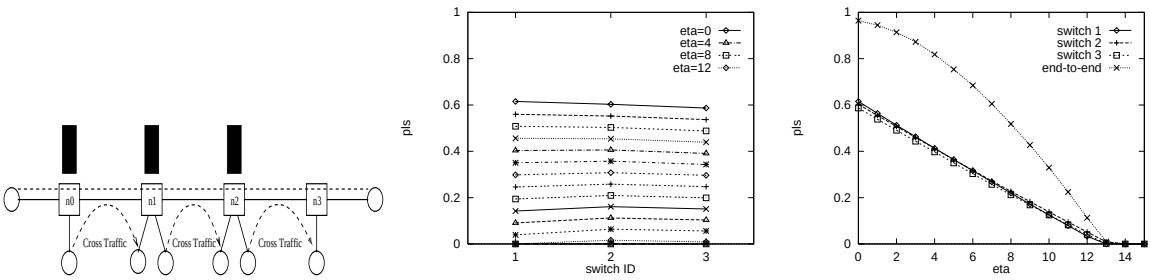

Figure 4.21. QoS distribution on multi-hop path with three medium-loaded switches. Left: Switch loads. Middle: Static QoS distribution. Right: Dynamical QoS distribution.

We first consider the scenario when all the switches are evenly loaded. Figure 4.21 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 comparable. 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.

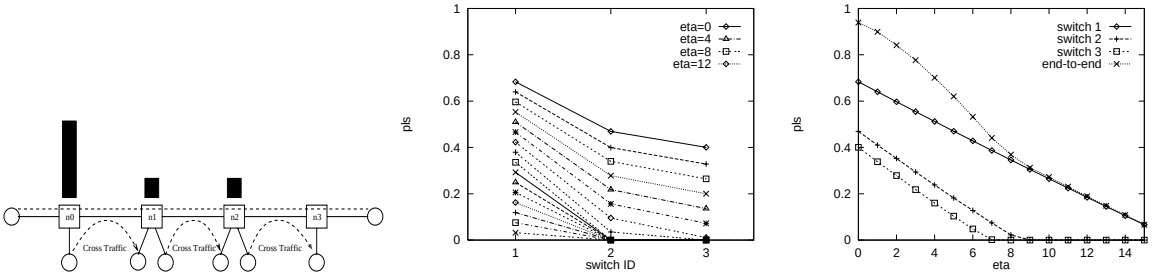

Figure 4.22. QoS distribution on multi-hop path with one heavy-loaded switch and two light-loaded switches. Left: switch loads. Middle: static QoS distribution. Right: dynamical QoS distribution.

Next we study the QoS distribution among switches when they are under different load conditions as depicted in Figure 4.22 (left). Figure 4.22 (middle) shows the static QoS distribution among the switches by plotting the QoS of all the classes at each switch. As stated in Section 4.1.4, the classes in the lightly-loaded switches experience better QoS than the corresponding classes in the heavily-loaded switch. Figure 4.22 (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 4.1.4, 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 4.22 (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.

The QoS distribution over load imbalance system is further demonstrated by Figure 4.23 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 each switches under these four different load imbalance 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.

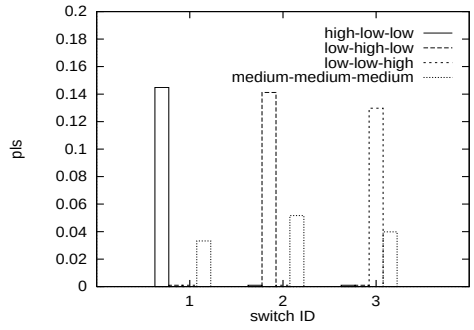

Figure 4.23. To satisfy given QoS target, the actual QoS achieved at each switch with respect to different load imbalance patterns.

4.3 Conclusion

In this chapter, we advance previous theoretical work on 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 proposed practical enhancements by introducing scaling function and discussing load imbalance issues. We have 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 theoretical results in Chapter 3. The performance results, overall, show that user-specified services can be efficiently and effectively achieved over the network with optimal aggregate-flow per-hop control substrate when coupled with adaptive label control.

5 SYSTEM BUILDING AND BENCHMARKING

Collaborating with Cisco Systems, we have implemented the optimal aggregate-flow per-hop control in commercial routers. The system building experience shows that the optimal per-hop control scheme purposed is practical and implementable. The overhead brought by optimal per-hop control is small. We also conducted benchmarking over Q-Bahn testbed which is comprised of Cisco 7206 VXR routers running developed optimal per-hop control. Our benchmarking output confirms the previous results from theoretical analysis and simulation study, and demonstrates the scalability of the QOS provisioning architecture.

In Section 5.1, we discuss the overall design and the key components in optimal per-hop control implementation. Section 5.2 describes system building procedure. In Section 5.3, we present our benchmarking results.

5.1 System Design

5.1.1 Key issues

In Section 4.1.1, we described an algorithm for optimal aggregate-flow per-hop control based on weighted fair queue. The basic idea is to online measure the traffic to each class and dynamically adjust the weights of classes commensurate to the current load distribution among them (cf. Section 4.1.1 for more details). Our implementation of optimal per-hop control in Cisco routers follows the same algorithm with focus on system efficiency. Two key issues related to efficiency are:

- How to keep the operation in packet forwarding path as minimal as possible, including packet classification, class load measurement, and enqueueing?
- How to reduce the overhead of periodical weight recalculation?

We also expect the optimal per-hop control may coexist with other QoS solutions in practice. Thus our design reuses many existing QoS mechanisms provided in Cisco routers and provides interface for flexible user configuration.

The platforms we use are Cisco 7200 series routers with Cisco Internetwork Operating System (IOS) version 12.2. The optimal per-hop control implementation can also be transplanted to other platforms with minor change.

5.1.2 Overall Structure

In our design, the optimal per-hop control is built upon class based weight fair queue (CBWFQ) mechanism in Cisco routers (cf. [80] for an overview) and its functionality can take effects on individual output interfaces. If enabled, the optimal per-hop control intercepts and processes packets in Cisco express forwarding (cf. [78] for more information) which is the default packet process mode in Cisco 7200 series routers.

In express forwarding mode, packets are handled by interrupt when received. Following is a general description of the packet process procedure during express forwarding when CBWFQ is configured on the corresponding output interface [80, 79].

1. The interface processor first detects there is a packet on the network media, and transfers this packet to the input/output memory on the router.

2. The interface processor generates a receive interrupt. During this interrupt, the central processor determines what the type of the packet (IP), and then begins to switch the packet.

3. The processor searches the route cache to determine if the packet's destination is reachable, what the output interface should be, what the next hop towards this destination is, and finally, what MAC header the packet should have to successfully reach the next hop. The processor uses this information to rewrite the packet's MAC header.

4. When CBWFQ is enable on the outbound interface, the processor decides which class to put the packet and copies the packet to the queue of the class. The receive interrupt then returns, and the process that was running on the processor before the interrupt occurred continues running.

5. After the transmission of the current outgoing packet is completed, the output interface processor generates an interrupt. During the interrupt, the central processor determines which packet to send next based on the CBWFQ algorithm, copies the packet to the transmit queue of the interface, then returns.

6. The output interface processor detects the packet on its transmit queue, and transfers the packet onto the network media.

In the above procedure, after determining the output interface of the packet and before putting the packet into the output queue of the interface, the IOS detects that the optimal per-hop control is enabled, and then passes control to the optimal per-hop control module (referred as OPC module from now on).

Figure 5.1 depicts the overall structure of optimal per-hop control module.

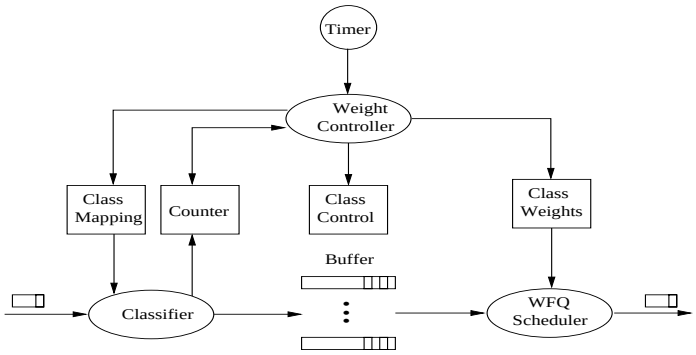

Figure 5.1. Structure of optimal aggregate-flow per-hop control module implemented in Cisco routers.

The OPC module has two components: classifier, which is packet-driven, and weight controller, which is timer-driven. The data abstraction used and/or updated by OPC module are:

- Class mapping table: determines the mapping from DSCP values to class indices.
- Counters: records the traffic characteristics of classes during the last time period, including number of packets received and number of packets dropped.
- Class weights: the weight assignment (bandwidth sharing percentage) of classes in CBWFQ.
- Class control table: the most important data structure in OPC module. It contains the control parameters used to calculate the class weights (cf. Figure 4.1.1). Our design supports the scaling function enhancement (cf. Section 4.1.3), thus the scaling function results of DSCP values are contained. The parameters in class control table can be flexiblely configured through Command Line Interface (CLI) of Cisco routers. For this purpose, link list is a preferred implementation.

When a packet is passed to OPC module, the classifier first looks up the DSCP value at the packet header and decides which class this packet should go to based on the information in the class mapping table, then updates the corresponding counters and puts the packet into the buffer of the target class. In the simplified case—the number of DSCP used is equal to the number of class configured at CBWFQ—the mapping from DSCP to class index is one to one (cf. Section 4.1.1). To reduce overhead, we use simple counters, together with existing traffic measurement mechanism provided by IOS, to record the traffic characteristics of classes.

The weight controller is a timer-driven module whose time interval can be set through CLI. When triggered, the weight controller first computes the weights of the classes based on the counter values and the information stored in class control table following the algorithm described in Figure 4.1.1 (or Figure 4.1.3 if scaling function is used), then updates the weights for CBWFQ, and finally resets the counters. To reduce overhead, we use a back-end process to act as weight controller. The timer inaccuracy is handled accordingly. Section 5.1.3 provides further details.

5.1.3 Dynamic Weight Computation

We further simplified the algorithm described in Figure 4.1.1 (or Figure 4.1.3 when scaling function is used) to improve system efficiency. First, rewrite the weight computation equation in Figure 4.1.1 as follows:

$$\begin{aligned}
\alpha_k &= (1-\nu)\frac{\lambda^k\hat{\eta}^k}{\sum_{j=1}^{L}\lambda^j\hat{\eta}^j} + \nu\frac{\lambda^k}{\sum_{j=1}^{L}\lambda^j} \\
&= (1-\nu)\frac{\lambda^k\frac{\eta^k-\eta_{min}}{\eta_{max}-\eta_{min}}}{\sum_{j=1}^{L}\lambda^j\frac{\eta^j-\eta_{min}}{\eta_{max}-\eta_{min}}} + \nu\frac{\lambda^k}{\sum_{j=1}^{L}\lambda^j} \\
&= (1-\nu)\frac{\lambda^k(\eta^k-\eta_{min})}{\sum_{j=1}^{L}\lambda^j(\eta^j-\eta_{min})} + \nu\frac{\lambda^k}{\sum_{j=1}^{L}\lambda^j}, \qquad k\in[1,L].
\end{aligned}$$

In practice, best-effort traffic exists and $\eta_{min} = 0$. Thus, we don't need to keep track of η_{min} which takes $O(n)$ time and the above equation can be simplified as

$$\alpha_k = (1-\nu)\frac{\lambda^k\eta^k}{\sum_{j=1}^{L}\lambda^j\eta^j} + \nu\frac{\lambda^k}{\sum_{j=1}^{L}\lambda^j}, \qquad k\in[1,L]. \tag{5.1.1}$$

Next, we need to relate λ^k to the counter values. Note that

$$\lambda^k = \frac{number\_of\_packet\_arrived \cdot average\_packet\_length}{last\_timer\_interval}.$$

A valid assumption is that the average lengths of packets in different classes within the last timer period are same. Let A^k denote the number of packet arrived at class k, $k \in [1, \cdots, L]$. Then equation (5.1.1) becomes

$$\alpha_k = (1-\nu)\frac{A^k\eta^k}{\sum_{j=1}^{L}A^j\eta^j} + \nu\frac{A^k}{\sum_{j=1}^{L}A^j}, \qquad k\in[1,L]. \tag{5.1.2}$$

Equation (5.1.2) is used by weight controller module. Note that the timer inaccuracy due to weight controller being back-end process does not influence the final weight calculation results. When scaling function is used, the corresponding equation will be

$$\alpha_k = (1-\nu)\frac{A^k\sigma(\eta^k)}{\sum_{j=1}^{L}A^j\sigma(\eta^j)} + \nu\frac{A^k}{\sum_{j=1}^{L}A^j}, \qquad k\in[1,L]. \tag{5.1.3}$$

In our implementation, we use integer operation when calculating equation (5.1.2) and (5.1.3) to improve efficiency. To represent weight α_k, $0 \le \alpha_k \le 1$ by an integer

β_k, let B be the integer representing the maximum range of β_k, $k \in [1, L]$, then $\beta_k = \alpha_k B$. Our simulation results show that $B \geq 512$ suffices in terms of accuracy—it achieves the same performance results as float weights are used in CBWFQ. The order of operation in Equation (5.1.3) and (5.1.2) is arranged in the way such that round-off errors are minimized.

When updating CBWFQ, β_k, $k \in [1, L]$ needs to be converted into bandwidth share, the data format used by CBWFQ.

5.1.4 User Configuration Interface

The parameters of OPC module can be configured through CLI using existing Cisco QoS configuration framework class-map and policy-map. Class-map defines the recognized DSCP values or DSCP sets. Policy-map defines the behavior of per-hop control including number of classes used, the mapping from DSCP values (or sets) to classes, and the scaling function values for classes. Policy-map is set on interface basis, thus an ISP can enforce different per-hop control policies on different interfaces.

There are also some parameters global to all interfaces, including the timer interval of weight controller. This simplifies the weight controller design—addition data structure (delta list) is not needed to keep track of timer and the recalculation of the class weights can be done uniformly for all interfaces.

5.2 System Building Procedure

Two independent organizations, Cisco IOS developing team and Network System Lab at Purdue University collectively built optimal per-hop control in Cisco routers. The collaboration protocol is as follows: First both parties jointly design the overall structure of optimal per-hop control (OPC module). Next, Network System Lab provides the pseudo code of OPC module, and Cisco team incorporates the module with the rest of IOS and compiles a prototype IOS image. The prototype images is then downloaded and tested by Network System Lab over Q-bahn testbed at Purdue.

The testing results are sent back to Cisco team for debugging. The modified images will be tested and debugged in the same way repeatedly until a final version is reached.

The network configuration for testing follows the ones used in simulation (Figure 5.2)to verify testing results. Some software tools are developed to automate the tests. The whole system development process took 6 months with most of the time devoted to testing and debugging.

The benchmarking and performance evaluation was carried out over Q-bahn testbed by Network System Lab independently at Purdue.

5.3 Benchmarking Results

This section presents the benchmarking and performance evaluation results obtained from Q-bahn testbed—an 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. Figure 5.2 shows the 11-switch Abilene-like topology of Q-bahn testbed.

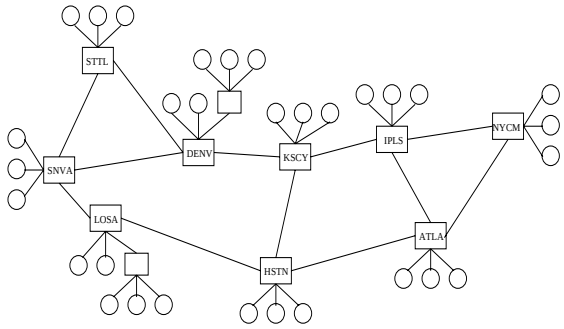

Figure 5.2. Network topology of Q-bahn testbed

5.3.1 QoS Differentiation

In this section, we examine the QoS differentiation behavior, the most basic property, of optimal per-hop control.

The first benchmarking setup is as follows: we configure the router with 8 classes, and send different traffic volume to different classes while keeping the total arriving traffic constant. We measure the average QoS experienced at classes 1–8 and examine the QoS ordering among these classes under different load distribution.

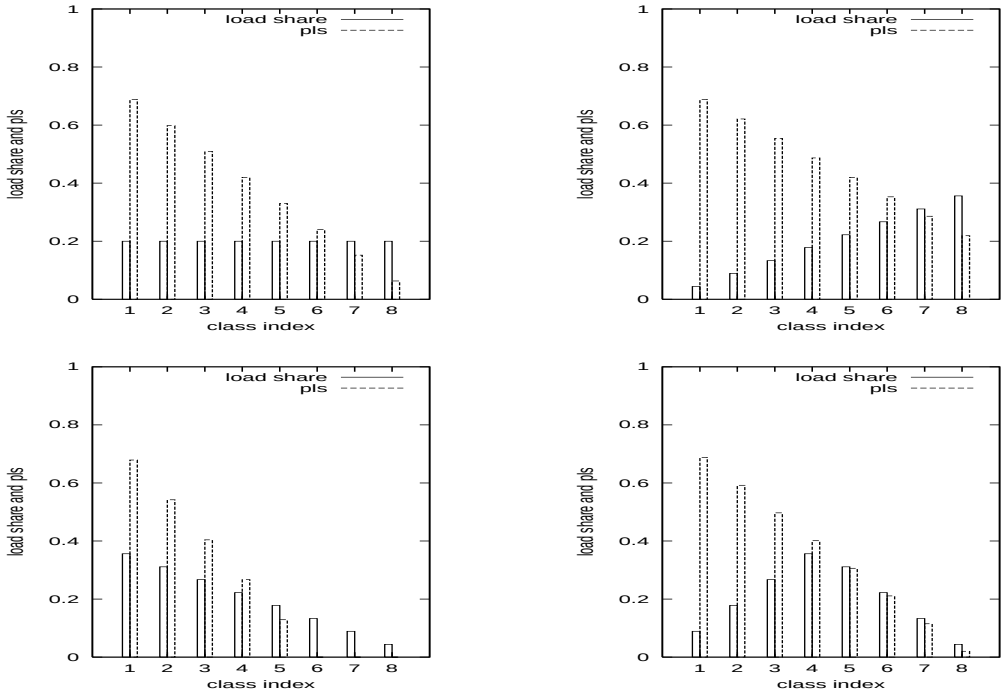

Figure 5.3. QoS separation among classes under different load distribution

Figure 5.3 shows load distribution and the average QoS achieved at different classes. The load distribution is depicted by the percentage of traffic received at each classes out of the total traffic arrived at the router. The QoS is measured as the average packet loss rate at each class during the period. Each figure of 5.3 represents a different load distribution. Class indices are marked on the X-axis. For each class, the left box represents the load percentage and the right box represents the packet loss rate. In top left figure, all classes receive same traffic volume thus the load percentage is 1/8 for all classes. In top right figure, higher classes get more traffic. In particular, the ratio of load at class 1–8 are $1 : 2 : 3 : 4 : 5 : 6 : 7 : 8$. In bottom left figure, higher classes get less traffic where ratio of load at class 1–8 are in reverse

order: 8 : 7 : 6 : 5 : 4 : 3 : 2 : 1. Bottom right figure shows another pattern of load distribution where ratio of load at class 1–8 are 2 : 4 : 6 : 8 : 7 : 5 : 3 : 1. In all the figures corresponding to different load distributions, we observe the semantics "higher class gets better QoS" is preserved. Note that the average packet loss rate over all classes are same for all the four figures since the total arriving traffic at the router is constant.

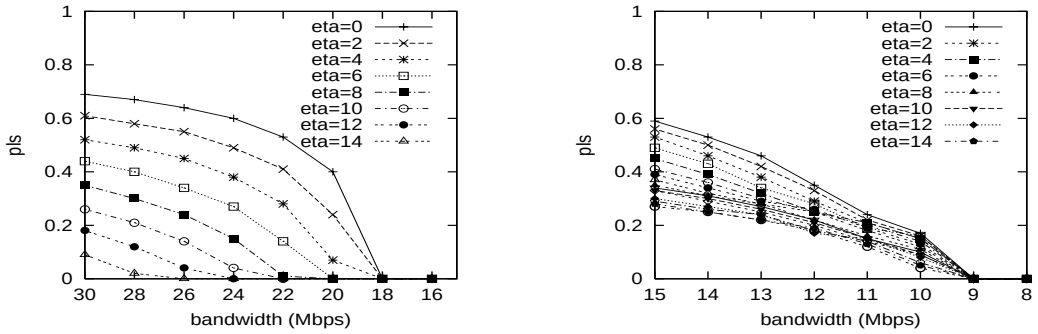

Figure 5.4. QoS separation among classes under different congestion level

Next, as a complement to the above setup, we vary the total amount of arriving traffic at the router, which determines the congestion level and overall QoS at the router, while keeping the load distribution among the classes constant. Figure 5.4 shows the QoS achieved at different classes under different congestion level. The traffic load is evenly distributed among the classes, i.e. all classes receive same percentages of total traffic, and the congestion level is increased/decreased by increasing/decreasing the traffic sending to individual classes simultaneously. Figure 5.4 (left) demonstrates the result when 8 classes are used and Figure 5.4 (right) demonstrates the result when 16 classes are used. The X-axis represents the traffic rate (Mbps) received per class and the Y-positions of the points mark the QoS (packet loss rate) achieved at the classes. We observe the following: although the overall QoS is getting better/worse as the congestion level (traffic rate per class) is decreased/increased, the semantics "higher class gets better QoS" is preserved.

5.3.2 Dynamic Environment

We also conducted more realistic experiments over Q-bahn testbed. In each experiment, we generate a large number of processes on the hosts connected to Q-bahn testbed (Figure 5.2). These processes setup connections (sessions) between each other and send traffic over Q-bahn testbed. For each session, the source and destination hosts, interarrival time, lifetime, sending rate, and QoS(throughput) requirement are randomly selected following pre-defined distribution. Our goal is to study the behavior of the optimal per-hop control in large-scale dynamic environment.

Figure 5.5 shows the result of a two-hour experiment. In the experiment, we generated 1378 sessions with average interarrival time 10 seconds and average session lifetime 10 minutes. The average sending rate of sessions is 6Mbps. The traffic are sent to 5 different classes. The number of sessions selecting different classes has the ratio $9 : 7 : 5 : 3 : 1$, i.e. the higher classes are selected by less sessions.

The left figures in 5.5 shows the workload configuration. The left top figure depicts the total number of active sessions over the testbed as time evolves. The left middle figure depicts the percentage of sessions generated on each end host. The left bottom figure depicts the sessions distribution among the end host as time evolves. At any given time, the average number of sessions over all hosts and the corresponding standard deviation are displayed.

The right figures in 5.5 demonstrates the class assignment and QoS achieved. The right top figure depicts the percentage of sessions selecting different classes. The right middle figure depicts the percentage of sessions that achieve different levels of QoS satisfaction. The right bottom figure, which is the most important figure, demonstrates the relation between QoS satisfaction level and the class selected. The average QoS satisfaction ratios for sessions selecting different classes are shown in the right bottom figure.

The experiment results confirms that the optimal per-hop control achieve “higher classes get better QoS” semantics in any scenarios including dynamic environments.

5.4 Conclusion

This chapter describes our experience on building the optimal per-hop control in Cisco routers and benchmarking over Q-bahn testbed. Our results shows that the optimal per-hop control is a practical and implementable scheme which can provide predictable services in realistic and dynamic network environments.

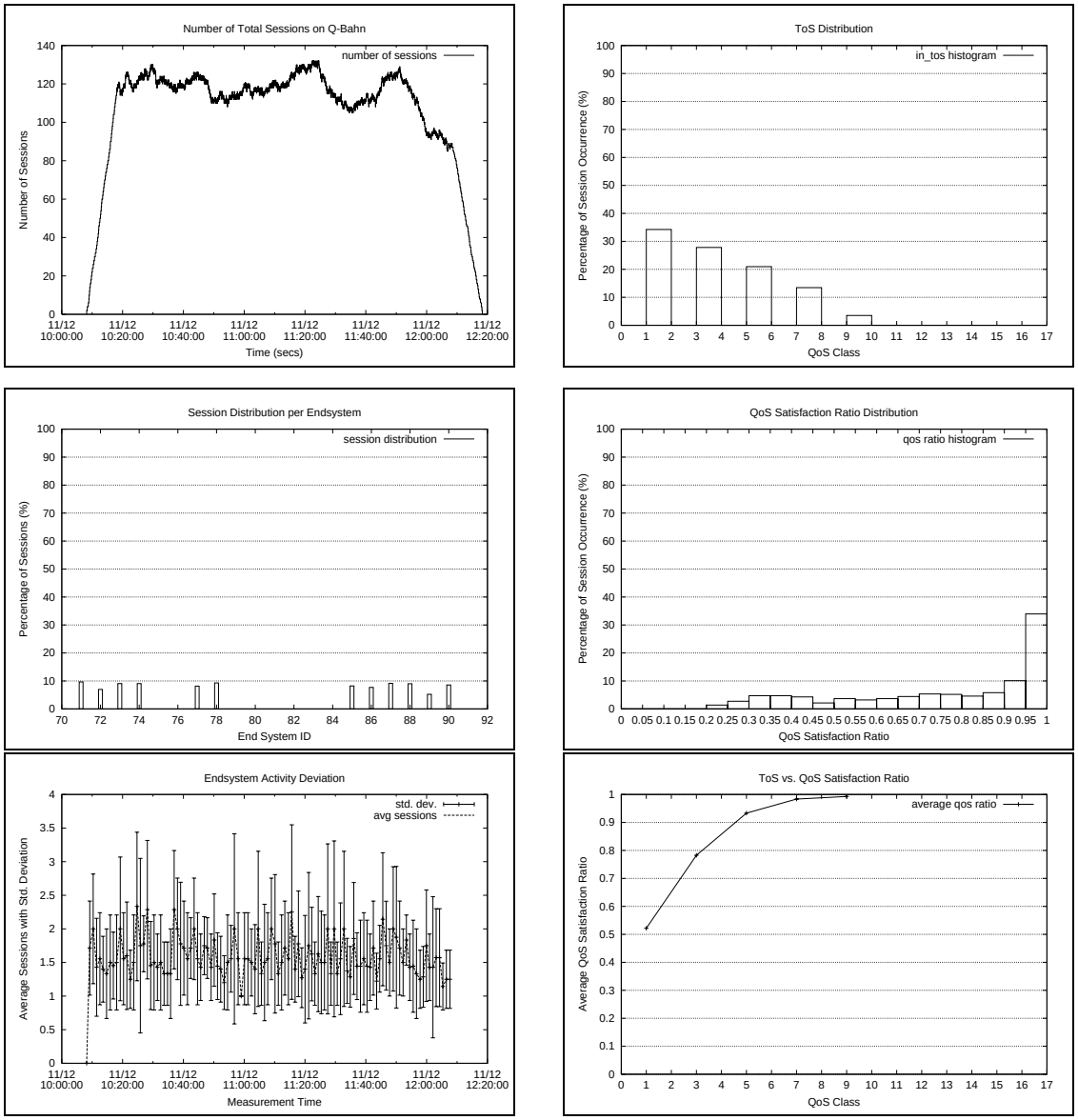

Figure 5.5. Q-Bahn experiment report

6 STOCHASTIC MODELING AND OPTIMIZATION

6.1 Introduction

In this chapter, we study the stochastic modeling and optimization of aggregate-flow scheduling in multi-class $G/G/1$ queueing systems, where each user flow has a QoS requirement and schedulers allocate resources such that user satisfaction for the whole system is maximized. This work is motivated by our results in Chapter 3 on optimal per-hop control design which did not consider the impact of stochasticity of the input: arrival processes with general interarrival and service times. For certain scheduler spaces—in particular, work conserving, non-preemptive and non-anticipative schedulers—various forms of Kleinrock's conservation law [40] can be shown to hold. The optimum scheduling problem then asks: Given a target performance or quality of service (QoS) requirement associated with the input, find a scheduler that comes closest, in a suitable sense, to achieving the desired performance. Kleinrock's conservation law has the effect of making the optimal scheduling problem constrained, as the performance space of the input flows must lie on a hyperplane defined by the conservation law.

The optimal per-flow scheduling problem in multi-class queueing systems with n flows or types was pioneered by Coffman and Mitrani [19], with a string of works (cf. Section 6.2 for related work) that have focused on characterizing the structure of the performance space. Our work can be viewed as extending Coffman and Mitrani's per-flow scheduling framework by introducing aggregate-flow schedulers that capture the constraint that a router may only have m, $m \leq n$, service classes at its disposal.

In Chapter 2, we specify aggregate-flow schedulers by assuming a many-to-one function, $\xi : \{1, \ldots, n\} \to \{1, \ldots, m\}$, called a classifier, and requiring that the scheduler maps each flow $i \in \{1, \ldots, n\}$ to one of m service classes as specified by

$\xi(i)$. The resultant m (or less) aggregated super-flows are then treated as input to an m-flow per-flow scheduler, which, in conjunction with the classifier ξ, defines the aggregate-flow scheduler. The "goodness" of an aggregate-flow scheduler is evaluated with respect to the QoS received by individual flows, hence the criterion of optimality remains unchanged. Since $m = n$ and $\xi =$ identity map corresponds to per-flow schedulers, the optimal aggregate-flow scheduling framework can be viewed as a generalization of Coffman and Mitrani's per-flow scheduling framework.

6.1.1 Features of Optimal Aggregate-flow Scheduling

Aggregate-flow scheduling—and per-flow scheduling as a special case—has the following features that are relevant to optimal scheduling:

Kleinrock's conservation law. The stochastic input along with properties of the scheduler space induce conservation laws—Kleinrock's conservation law [39, 40, 75] and more restricted forms called strong conservation laws [76, 23]—which capture the trade-off relation that to improve service to one flow, the performance of one or more flows must be sacrificed. Kleinrock's conservation law can be viewed as an application of Little's conservation law [48, 84]. Both hold for general $G/G/1$ systems and scheduler spaces, but details of the underlying stochastic framework must be specified as there is no unique form.

Performance space. Previous works in per-flow scheduling have focused on properties of the performance space. For a given input and scheduler space, the performance space is the set of n-dimensional performance vectors that are achievable by some schedulers in the scheduler space. Kleinrock's conservation law constrains the performance space to lie on a $(n-1)$-dimensional hyperplane. Strong conservation laws additionally impose a polymatroidal structure. For example, for $M/G/1$ systems with work conserving, non-preemptive and non-anticipative scheduling disciplines, the performance space is convex and spanned by random weighted combinations of static priority schedulers [30].

Optimum performance. The process of finding an optimum scheduler can be divided into two steps: (i) find a performance vector in the performance space closest to the desired target QoS, and (ii) identify a scheduler in the scheduler space that achieves the performance of the first step. When the notion of "closest" is captured by the minimum mean-square error (MMSE) under the L_2 norm—a commonly used criterion in estimation, control and optimization—then step (i) can be further subdivided into two steps: first, find the orthogonal projection of the target QoS vector on the conservation law hyperplane, then find a vector in the performance space closest to the projection point. When the performance space is suitably nice—e.g., convex—the orthogonal projection may lie inside the performance space; in general, this needs not be the case.

Complexity of optimum scheduling. The 3-step decomposition property of finding an optimum scheduler facilitates studying the algorithmic aspect of computing an optimum scheduler for a given stochastic input and target performance, when Kleinrock's conservation law holds, without necessitating a detailed characterization of the structure of the performance space. The complexity of optimal per-flow scheduling for a given input and target performance vector involves finding an optimal point in the performance space with respect to a given performance criterion. Optimal aggregate-flow scheduling has the added effect of limiting the feasible region to subspaces induced by the scheduling constraint that only $m \leq n$ service classes are available to an aggregate-flow scheduler.

Figure 6.1 illustrates of the conservation law hyperplane, performance space, and aggregate-flow subspace for $n = 3$ and $m = 2$.

6.1.2 New Contribution

A conceptual contribution of the work in this chapter is the introduction of stochastic framework of optimal aggregate-flow scheduling, which extends Coffman and Mitrani's optimal per-flow scheduling framework [19]. The generalized framework

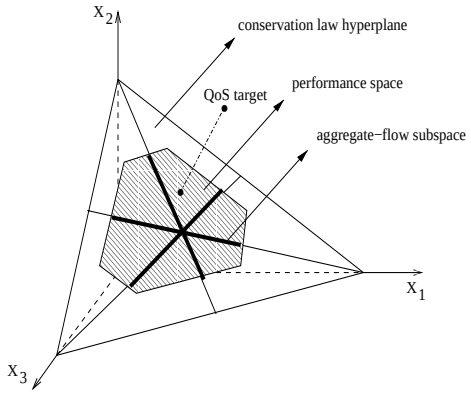

Figure 6.1. Depiction of conservation law hyperplane, per-flow performance space, and aggregate-flow subspace for an $n = 3$ and $m = 2$ optimum scheduling example.

provides, in part, a theoretical foundation for QoS provisioning using aggregate-flow scheduling under stochastic input. The technical contribution is a computational complexity characterization of aggregate-flow scheduling where we prove that optimal aggregate-flow scheduling for multi-class $G/G/1$ systems is NP-hard. This stands in contrast with optimal per-flow scheduling studied in previous contexts, including certain polymatroid optimization under strong conservation laws and optimal aggregate-flow scheduling without conservation laws, which are poly-time solvable. The latter arises in relative service differentiation under static input, and has cubic time complexity (Chapter 3). Kleinrock's conservation law in combination with optimal flow aggregation imparts computational hardness. In settings where only one of the conditions holds, optimal aggregate-flow scheduling is polynomially solvable.

The complexity result applies to a broad range of multi-class $G/G/1$ queueing systems under work conserving, non-preemptive and non-anticipative scheduling disciplines, when Kleinrock's conservation law holds, without requiring detailed knowledge of the associated performance space. This is achieved by a sufficiency condition which states that (i) as long as the performance space contains an open ball—however small—centered around the performance vector achieved by the FIFO scheduler, and (ii) within the ball performance inside a service class is the same, the optimal aggregate-flow scheduling problem is NP-hard. While the aforementioned conditions

point toward the applicability of the hardness result to general queueing systems, even for very simple queueing systems such as $M/M/1$ whose performance space satisfies (i) and (ii)—i.e., they are not assumptions—optimal aggregate-flow scheduling remains NP-hard.

The rest of the chapter is organized as follows. In the next section we discuss related works. In Section 6.3, we introduce a multi-class queueing framework including assumptions on the stochastic input and scheduler space. We then define the optimal aggregate-flow scheduling problem for multi-class $G/G/1$ systems. In Section 6.4, we describe the structure of optimal aggregate-flow scheduling and state the main result. Section 6.5 proves hardness of one-dimensional clustering with linear constraints, a key component in the proof of the main result. The results in this chapter are also presented in [71].

6.2 Related Work

The problem of finding an optimal scheduler for a given performance target under stochastic input in per-flow multi-class queueing systems was introduced by Coffman and Mitrani [19]. Since then, the area has been intensively investigated, with focus on characterizing the per-flow performance space satisfying strong conservation laws [76]—a weaker form being Kleinrock's conservation law—under different assumptions on the input and scheduler space.

Coffman and Mitrani's seminal paper [19] studied the per-flow multi-class $M/M/1$ system for preemptive schedulers. They showed that the performance space is convex and spanned by static priority schedulers (i.e., priority queues). Gelenbe and Mitrani [30] studied the $M/G/1$ queue for non-preemptive schedulers and proved analogous results. In [23], Federgruen and Groenevelt extended [30] to $M/G/c$, and they established a corresponding result for $G/M/1$ under preemptive schedulers in [22]. Federgruen and Groenevelt [23, 22] used the polymatroidal structure of the performance space to show that convex separable objective functions can be optimized

in polynomial time. Georgiadis and Viniotis [32] considered $GI/G/1$ systems under general work conserving schedulers, including time-varying adaptive policies, without a priori assuming ergodicity. Shanthikumar and Yao [76] established the equivalence between strong conservation laws and the polymatroidal structure of the performance space, showing that the latter is not only sufficient but also necessary. Bertsimas [2] provides a survey of optimal control in multi-class queueing systems following Coffman and Mitrani's framework [19], where relaxation is used to find approximations of the performance space and corresponding performance bounds. Chen and Yao [7] (Chapter 11) provide a comprehensive exposition of related works, including generalized conservation laws and their relationship to extended polymatroids with application to closed queueing systems.

The aforementioned works in optimal per-flow scheduling considered computational complexity. Polymatroid optimization, enabled by strong conversation laws, affected poly-time solvability of convex separable functions for queueing systems where the $2^n - 2$ inequalities of the strong conservation law can be represented in space polynomial in n. For linear objective functions where the optimal solution must be a vertex of the base polytope of the polymatroid, this dependence of the input size on a compact representation of the constraints is removed. This is due to the fact that an optimal solution can be recovered from the n coefficients of the objective function. Poly-time solvability of linear objective functions extends to generalized conservation laws. An important negative result was proved by Papadimitriou and Tsitsiklis [58] for optimal control of closed multi-class queueing networks which they showed to be EXP-complete. The queueing system considered, however, is significantly different in that it is a closed queueing network where stochasticity only arises from the (exponential) service time distribution, and combinatorial hardness is built in at multiple levels including routing and scheduling. Thus, it is not surprising that the stochastic control problem considered is computationally hard, although the degree of its hardness (EXP-complete) is interesting.

In Chapter 3, we studied optimal aggregate-flow multi-class scheduling in a shared resource environment where bandwidth is directly provisioned. That is, the semantics of resource sharing is with respect to the underlying resource—i.e., bandwidth—and not the performance induced by the apportioned resources. Under the minimum mean-square error (MMSE) criterion and relativization of user bandwidth requirements, optimal aggregate-flow scheduling is shown to be poly-time solvable, in particular, has cubic time complexity. This chapter generalizes the optimal aggregate-flow multi-class scheduling framework by considering stochastic input for which Kleinrock's conservation law holds. The study of optimal per-flow multi-class scheduling showed that per-flow scheduling combined with strong conservation laws leads to poly-time complexity of convex objective functions. In Chapter 3, it was shown that aggregate-flow scheduling without conservation laws is also computationally tractable. This chapter shows that aggregate-flow scheduling combined with conservation laws is NP-hard.

A key component to proving NP-hardness of optimal aggregate-flow scheduling involves reduction from a one-dimensional constrained clustering problem with MMSE objective function. Surprisingly, little is known about the complexity of one-dimensional optimal clustering with linear constraints and MMSE objective function. We prove it is NP-complete. Brucker [6] showed that one-dimensional unconstrained clustering under MMSE is poly-time solvable. The complexity of k-dimensional, $k \geq 2$, unconstrained clustering under MMSE remains an open problem [6, 35, 28]. Gal and Klots [27] solved a form of one-dimensional unconstrained clustering where the objective is to maximize the sum of average group weights over all groups.

Statistical multiplexing—a form of flow aggregation—has been studied with the aim of characterizing the multiplexing gain via envelop curves and effective bandwidth [4, 21, 33, 86]. From a scheduling perspective, these works fall under the category of per-flow multi-class scheduling. They showed that bandwidth is more efficiently utilized the more flows are multiplexed or aggregated. In contrast with Coffman and Mitrani's per-flow scheduling framework where the performance space—

given by strong conservation laws—captures the trade-off across different schedulers, in statistical multiplexing an effective characterization of achievable performance for specific schedulers (e.g., FIFO, EDF, SP, GPS) is sought with the aid of the law of large numbers assuming independence among flows.

A notion similar to flow aggregation, known as job grouping [82], has been studied in the scheduling literature. The job grouping model studies optimal scheduling for deterministic sequences of jobs. Aggregation is defined at the individual job level, and optimal scheduling studies the grouping of jobs so as to minimize processing time. In essence, optimal scheduling in the job grouping context involves unconstrained clustering which, in the absence of conservation laws, is polynomially solvable by dynamic programming.

Although optimal aggregate-flow scheduling was originally motivated by the practical context of designing efficient routers for achieving scalable QoS provisioning, it has led to interesting linkages between stochastic scheduling, clustering, job grouping, and scalable network design.

6.3 System Model

The system model is comprised of four parts. The first part describes the stochastic framework of multi-class queueing which, in addition to notational set-up, is needed to define aggregate-flow scheduling. The second part defines the optimal aggregate-flow scheduling problem. The third part discusses Kleinrock's conservation law with respect to the underlying input and scheduler space, and how it impacts scheduling. The fourth part specifies an interface between the stochastic and algorithmic components of optimal aggregate-flow scheduling which is used in Section 6.4.

6.3.1 Multi-class Queueing Model

Consider an n-flow (or type) $G/G/1$ system with general interarrival and service times. Let $U = (U_k)$, $U_k = [T_k, S_k, C_k]$, $k \in \mathbb{Z}$, be a sequence of random variables rep-

resenting the input, where $T_k \in \mathbb{R}_+$ (the positive reals excluding 0) is the interarrival time between the k-th and $(k+1)$-th arriving packets, $S_k \in \mathbb{R}_+$ is the service time of the k-th packet, and $C_k \in \{1, \cdots, n\}$ is the flow index of the k-th arriving packet. Following [5], the evolution of the system at arrival instants may be described by a recursive stochastic equation

$$X_{k+1} = f(X_k, U_k) = f(X_k, [T_k, S_k, C_k]), \quad k \in \mathbb{Z} \tag{6.3.1}$$

where X_k denotes the state of the system seen by the k-th arriving packet, and f is a measurable function on the joint probability space and captures the scheduling discipline. For stochastic schedulers, an additional random input is needed to represent the behavior of f. For strictly stationary input, (6.3.1) can be shown to have well-defined solutions [5]. For our purposes, it suffices to have a stochastic framework wherein Kleinrock's conservation law holds, which need not be restricted to strictly stationary input. For example, this applies to second-order self-similar processes which are used to model long-range dependence in Internet traffic.

Recall in Chapter 2, we specify aggregate-flow schedulers by assuming a many-to-one function, $\xi : \{1, \ldots, n\} \to \{1, \ldots, m\}$, $m \leq n$, called a classifier, and requiring that a scheduler maps each flow $i \in \{1, \ldots, n\}$ to one of m service classes as specified by $\xi(i)$. The resultant m (or less) aggregated super-flows are then treated as input to an m-flow per-flow scheduler, which, in conjunction with the classifier ξ, defines the aggregate-flow scheduler.

Definition 6.3.2 An n-to-m *aggregate-flow scheduler* g is a pair (ξ, f), where $\xi : \{1, \ldots, n\} \to \{1, \ldots, m\}$ is a classifier, f is a state transition function of an m-flow multi-class queueing system, and the action of g is given by

$$\begin{aligned} X_{k+1} &= g(X_k, U_k) \\ &= f(X_k, [T_k, S_k, \xi(C_k)]), \quad k \in \mathbb{Z}. \end{aligned}$$

Since ξ and f are measurable, g is measurable. An n-to-m aggregate-flow scheduler is just a special case of an n-flow per-flow scheduler, hence the stochastic framework

of per-flow scheduling can be inherited. On the other hand, since $m = n$ and $\xi =$ identity map corresponds to per-flow schedulers, aggregate-flow schedulers can be viewed as a generalization of per-flow schedulers, where the classifier ξ allows various forms of information loss due to $m < n$ to be imposed on the system, including the special case of no loss.

An alternative definition of aggregate-flow scheduler is: g is an n-to-m aggregate-flow scheduler if there exists a classifier ξ such that given an arbitrary system state x_k seen by the k-th packet and two instances of the $(k+1)$-th packet $u_{k+1} = [t_{k+1}, s_{k+1}, c_{k+1}]$ and $u'_{k+1} = [t_{k+1}, s_{k+1}, c'_{k+1}]$, $g(x_k, u_{k+1}) = g(x_k, u'_{k+1})$ if $\xi(c_{k+1}) = \xi(c'_{k+1})$. The second definition is more behavioral and does not make use of a state transition function f of an m-flow multi-class queueing system. The two definitions can be shown to be equivalent.

6.3.2 Optimal Aggregate-flow Scheduling

Let $\boldsymbol{w} = (w_1, \cdots, w_n)$ denote the performance vector of an n-flow multi-class queueing system where $w_i > 0$ is a performance measure associated with flow i. In general, $\boldsymbol{w}$ may depend on (X_k) which is determined by the state transition function f and input $U = (U_k)$. This is denoted by $\boldsymbol{w} = \phi_f(U)$ where ϕ_f is assumed measurable. Given input U and a set of schedulers $\mathcal{S}$, the (per-flow) *performance space* with respect to U and $\mathcal{S}$ is defined as

$$\mathcal{H} = \{\boldsymbol{w} : \exists f \in \mathcal{S}, \boldsymbol{w} = \phi_f(U)\}.$$

Thus, $\mathcal{H}$ is the set of performance vectors that are achievable by some schedulers in the scheduler space. We use $\mathcal{H}_a$ to denote the corresponding aggregate-flow performance space when f is restricted to be an n-to-m aggregate-flow scheduler.

Let $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_n)$ be a desired target performance vector where $\theta_i > 0$ represents the performance or QoS requirement of flow i. Under the minimum mean-square

error criterion of "goodness," the per-flow optimal scheduling problem in n-flow multi-class $G/G/1$ systems given input U and scheduler space $\mathcal{S}$ is defined as

$$\inf_{\boldsymbol{w}\in\mathcal{H}} \|\boldsymbol{w}-\boldsymbol{\theta}\|^2 \tag{6.3.3}$$

where $\|\cdot\|$ is the L_2 norm. Note that finding a scheduler $f \in \mathcal{S}$ that achieves an optimal w is treated as a separate problem. This is justified by the negative nature of the main result. The corresponding optimal aggregate-flow scheduling problem when the scheduler is restricted to belong to the aggregate-flow scheduler space $\mathcal{S}_a$ is given by

$$\inf_{\boldsymbol{w}\in\mathcal{H}_a} \|\boldsymbol{w}-\boldsymbol{\theta}\|^2 \tag{6.3.4}$$

where $\mathcal{H}_a$ is the aggregate-flow performance space under input U and scheduler space $\mathcal{S}_a$. For $m = n$, $\mathcal{H}_a = \mathcal{H}$, i.e., per-flow scheduling is a special case. Hence, the case of interest in aggregate-flow scheduling is $m < n$ for which $\mathcal{H}_a \subseteq \mathcal{H}$.

6.3.3 Conservation Law

Kleinrock's conservation law takes on the form of an inner product functional. The following shows an instance under strictly stationary input. Let $1/\lambda = E\{T_0\}$, $1/\mu = E\{S_0\}$, $\rho = \lambda/\mu$, and $\lambda_i = P\{C_0 = i\}\lambda$, $1/\mu_i = E\{S_0 \,|\, C_0 = i\}$, $\rho_i = \lambda_i/\mu_i$, for $i = 1, \cdots, n$. Let X_k be the waiting time W_k, and let $w_i = E\{W_0 \,|\, C_0 = i\}$.

Proposition 6.3.5 *Consider an n-flow $G/G/1$ queueing system in steady state, i.e., a stationary solution of (6.3.1) exists. Given input U, for any non-preemptive, work-conserving and non-anticipative scheduler,*

$$\sum_{i=1}^{n} \rho_i w_i = \bar{c}$$

where $\bar{c} > 0$ is a constant that depends on the input.

Proposition 6.3.5 is a straightforward consequence of Theorem 6.3.2 in [5] (Chapter 6, pp. 202). In general, Kleinrock's conservation law specifies an $(n-1)$-dimensional hyperplane

$$\mathcal{H}^* = \{\boldsymbol{w} \,:\, \boldsymbol{\rho} \circ \boldsymbol{w} = \bar{c}\} \tag{6.3.6}$$

where $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$. We have $\mathcal{H}_a \subseteq \mathcal{H} \subseteq \mathcal{H}^*$. Strict stationarity in not necessary for Kleinrock's conservation law to hold. For example, for second-order stationary processes which have been used to model long-range dependent traffic [64], Kleinrock's conservation law can be shown to hold under certain steady state assumptions.

Coffman and Mitrani's per-flow scheduling framework [19] uses strong conservation laws [76] where, in addition to (6.3.6), $2^n - 2$ inequalities over the subsets of $J \subseteq \{1, \ldots, n\}$ are imposed

$$\sum_{i \in J} \rho_i w_i \geq \bar{c}_J.$$

Jointly with (6.3.6), the inequalities lead to a polytope whose vertices are static priority schedulers. In our work, the weaker form (6.3.6) suffices.

6.3.4 Aggregate-flow Performance Space and Open Ball Containment

Given $\xi : \{1, \ldots, n\} \to \{1, \ldots, m\}$, let $\mathcal{G}_\xi = \{\boldsymbol{v} \in \mathbb{R}^n : v_i = v_j \text{ if } \xi(i) = \xi(j)\}$. Let $\mathcal{G}_m = \bigcup_\xi \mathcal{G}_\xi$ where the union is over all n-to-m classifiers ξ. Thus, $\mathcal{G}_m$ denotes the set of all vectors that have at most m distinct elements. Under the assumption

$$w_i = \overline{w}_{\xi(i)}, \quad i = 1, \cdots, n \tag{6.3.7}$$

where flows belonging to the same service class experience the same performance, it holds that

$$\mathcal{H}_a \subseteq \mathcal{H} \cap \mathcal{G}_m. \tag{6.3.8}$$

Let $\boldsymbol{q} > \mathbf{0}$ denote the performance vector achieved by the FIFO scheduler. Note that in the definition of classifier, ξ need not be surjective. Hence, for $1 \leq m \leq n$, FIFO is an n-to-m aggregate-flow scheduler and $\boldsymbol{q} \in \mathcal{H}_a$. By work conservation and assumption (6.3.7), we have $q_1 = q_2 = \cdots = q_n = \bar{c}/\rho$. The following assumption ("open ball containment property") provides an interface between the stochastic and algorithmic components of optimal aggregate-flow scheduling:

$$\exists\, r > 0 \text{ such that } \mathcal{B}_r(\boldsymbol{q}) \cap \mathcal{H}_a = \mathcal{B}_r(\boldsymbol{q}) \cap \mathcal{H}^* \cap \mathcal{G}_m \tag{6.3.9}$$

where $\mathcal{B}_r(\boldsymbol{q}) \subseteq \mathbb{R}^n$ is the open ball of radius r centered around $\boldsymbol{q}$. We will show that assumption (6.3.9) suffices to make the main complexity result hold for any multi-class $G/G/1$ system when Kleinrock's conservation law holds.

6.4 Structure of Optimal Aggregate-flow Scheduling

In this section, we state the main result and show the structure of optimal aggregate-flow scheduling which can be related to one-dimensional optimal clustering with linear constraint. The latter is shown to be NP-complete in Section 6.5.

6.4.1 Main Result

Theorem 6.4.1 *Given an n-flow m-class $G/G/1$ system with user QoS requirement vector $\boldsymbol{\theta} \in \mathbb{R}_+^n$ and work-conserving, non-preemptive and non-anticipative schedulers, assume Kleinrock's conservation law (6.3.6) holds and the open ball containment property (6.3.9) on $\mathcal{H}_a$ is satisfied. Then optimal aggregate-flow scheduling under the MMSE criterion (6.3.4)*

$$\inf_{\boldsymbol{w} \in \mathcal{H}_a} \|\boldsymbol{w} - \boldsymbol{\theta}\|^2$$

is NP-hard.

Theorem 6.4.1 says that for multi-class queueing systems with general input where Kleinrock's conservation law holds and containment of a well-structured ball in $\mathcal{H}_a$ is satisfied, the problem of optimally grouping n input flows into m service classes such that the realized performance comes closest to users' target performance in the mean-square sense is computationally hard. In fact, our proof shows that a special case of the main result with $m = 2$ is already NP-hard. For strong conservation laws, $\mathcal{H}$ is convex and (6.3.9) is more easily satisfied.

The decision problem corresponding to Theorem 6.4.1 is given by the following.

Problem 6.4.2 [OPT-AGG] Under the assumptions of Theorem 6.4.1, for given $K \in \mathbb{R}_+$ and $\boldsymbol{\theta} \in \mathbb{R}_+^n$ decide if there exists $\boldsymbol{w} \in \mathcal{H}_a$ such that

$$\|\boldsymbol{w} - \boldsymbol{\theta}\|^2 \leq K.$$

We remark that the parameters of the input instance are actually defined over the rationals $\mathbb{Q}$, consistent with convention of computational complexity for capturing input length. For notational simplicity, we will continue to use reals assuming this is understood.

6.4.2 Decomposition

In the definition of OPT-AGG, it suffices for the performance requirement vector to lie on the conservation law hyperplane, i.e., $\boldsymbol{\theta} \in \mathcal{H}^*$. The restricted problem is called OPT-AGG$'$. Both problems are *computationally equivalent*: a poly-time algorithm for OPT-AGG$'$ gives a poly-time algorithm for OPT-AGG, and vice versa. This is enabled by the next proposition.

Proposition 6.4.3 *Given $\boldsymbol{\theta} \in \mathbb{R}^n$, let $\hat{\boldsymbol{\theta}}$ be the orthogonal projection of $\boldsymbol{\theta}$ on $\mathcal{H}^*$; i.e., $\hat{\boldsymbol{\theta}}$ is the unique solution to $\min_{\boldsymbol{w} \in \mathcal{H}^*} \|\boldsymbol{w} - \boldsymbol{\theta}\|^2$. For any $\mathcal{A} \subseteq \mathcal{H}^*$, $\boldsymbol{w}^*$ is a solution to*

$$\inf_{\boldsymbol{w} \in \mathcal{A}} \|\boldsymbol{w} - \boldsymbol{\theta}\|^2$$

if and only if $\boldsymbol{w}^*$ is a solution to

$$\inf_{\boldsymbol{w} \in \mathcal{A}} \|\boldsymbol{w} - \hat{\boldsymbol{\theta}}\|^2.$$

Proof. The orthogonal projection is given by

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} + \frac{\bar{c} - \boldsymbol{\rho} \circ \boldsymbol{\theta}}{\|\boldsymbol{\rho}\|^2} \boldsymbol{\rho} \tag{6.4.4}$$

where $\boldsymbol{\rho}$ and $\bar{c}$ are the parameters of $\mathcal{H}^*$. Since $\mathcal{A} \subseteq \mathcal{H}^*$, for all $\boldsymbol{w} \in \mathcal{A}$ we have

$$\|\boldsymbol{w} - \boldsymbol{\theta}\|^2 = \|\boldsymbol{w} - \hat{\boldsymbol{\theta}}\|^2 + \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2.$$

$\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2$ is a constant from which the proposition follows. ■

Proposition 6.4.3 states that finding a closest point in a subset $\mathcal{A}$ of the conservation law hyperplane with respect to an arbitrary performance requirement $\boldsymbol{\theta}$ can be decomposed into two steps: first, finding the orthogonal projection of $\boldsymbol{\theta}$ on $\mathcal{H}^*$, then finding a point in $\mathcal{A}$ closest to the projection point. When $\mathcal{A} = \mathcal{H}$ and the per-flow performance space $\mathcal{H}$ is suitably "nice"—e.g., convex polytope under strong conservation law—then the optimal per-flow solution is either the projection point itself, or it lies on the boundary of the polytope $\mathcal{H}$. In either case, the optimal solution can be found in polynomial time. The next proposition relates the optimal per-flow solution to a fairness property.

Proposition 6.4.5 *$\boldsymbol{w} \in \mathcal{H}^*$ is the solution to $\min_{\boldsymbol{w}\in\mathcal{H}^*} \|\boldsymbol{w} - \boldsymbol{\theta}\|^2$, if and only if $\boldsymbol{w}$ satisfies*

$$\frac{w_1 - \theta_1}{\rho_1} = \cdots = \frac{w_n - \theta_n}{\rho_n}. \tag{6.4.6}$$

Proof. $\boldsymbol{w} \in \mathcal{H}^*$ if and only if $\boldsymbol{\rho} \circ \boldsymbol{w} = \bar{c}$, and $\boldsymbol{w}$ satisfies (6.4.6) if and only if $\exists\, c \in \mathbb{R}$ such that $\boldsymbol{w} = \boldsymbol{\theta} + c\boldsymbol{\rho}$. Jointly we get $\boldsymbol{w} = \hat{\boldsymbol{\theta}}$, where $\hat{\boldsymbol{\theta}}$ is the orthogonal projection of $\boldsymbol{\theta}$ on $\mathcal{H}^*$, as specified by (6.4.4). ■

(6.4.6) can be viewed as a fairness criterion in the following sense: $\rho_i = \lambda_i/\mu_i$ represents the traffic intensity of flow i; $w_i - \theta_i$ measures the deviation from the performance target. By setting $w_i - \theta_i$ proportional to ρ_i, we enforce the trade-off: "The excess happiness (or misery) is inversely proportional to how much one sends."

Let OPT-AGG″ be a further restriction of OPT-AGG′ where the QoS requirement vector $\boldsymbol{\theta}$ is restricted to lie in an open ball on the hyperplane

$$\boldsymbol{\theta} \in \mathcal{B}_{r/2}(\boldsymbol{q}) \cap \mathcal{H}^*$$

where $r > 0$ is the radius of the open ball in Theorem 6.4.1. Since OPT-AGG″ is a special case of OPT-AGG′, OPT-AGG′ is computationally as hard as OPT-AGG″. Thus, to show hardness of OPT-AGG, it suffices to show that OPT-AGG″ is NP-hard.

6.4.3 Clustering

We show that optimal aggregate-flow scheduling is equivalent to an optimal clustering problem. Given $\xi : \{1, \cdots, n\} \to \{1, \cdots, m\}$ and $\boldsymbol{d} = [d_1, \cdots, d_m] \in \mathbb{R}^m$, define $\pi(\xi, \boldsymbol{d}) \in \mathbb{R}^n$

$$\pi_i(\xi, \boldsymbol{d}) = d_{\xi(i)}, \quad i = 1, \cdots, n.$$

Thus π is an n-dimensional vector whose components, as dictated by ξ, take on values from the m-dimensional vector $\boldsymbol{d}$. Following is a definition of a clustering problem which we show is equivalent to optimal aggregate-flow scheduling.

Problem 6.4.7 [OPT-CLUST] Under the assumptions of Theorem 6.4.1, for given $K \in \mathbb{R}_+$ and $\boldsymbol{\theta} \in \mathcal{B}_{r/2}(\boldsymbol{q}) \cap \mathcal{H}^*$ decide if there exist ξ and $\boldsymbol{d} \in \mathbb{R}_+^m$ such that

$$\|\pi(\xi, \boldsymbol{d}) - \boldsymbol{\theta}\|^2 \leq K$$

subject to $\pi(\xi, \boldsymbol{d}) \in \mathcal{H}_a$.

Proposition 6.4.8 *OPT-AGG″ has a solution if and only if OPT-CLUST has a solution.*

Proof. We can consider two cases: (i) $K \geq (r/2)^2$ and (ii) $K < (r/2)^2$. In case (i), $\boldsymbol{w} = \boldsymbol{q}$, and $\xi(i) = 1$ for $i = 1, \ldots, n$ and $d_1 = q_1$ are solutions to OPT-AGG″ and OPT-CLUST, respectively.

Consider case (ii). Assume OPT-AGG″ has a solution $\boldsymbol{w}$. The radius of the open ball in the definition of OPT-AGG″ implies that $\boldsymbol{w} \in \mathcal{B}_r(\boldsymbol{q})$. Since $\boldsymbol{w} \in \mathcal{H}_a$, by (6.3.9), $\boldsymbol{w} \in \mathcal{G}_m$. Thus there exist ξ and $\boldsymbol{d}$ such that $\boldsymbol{w} = \pi(\xi, \boldsymbol{d})$, and ξ, $\boldsymbol{d}$ is a solution to OPT-CLUST. In the reverse direction, $\boldsymbol{w} = \pi(\xi, \boldsymbol{d})$ is a solution to OPT-AGG″. ■

Proposition 6.4.8 implies that OPT-CLUST and OPT-AGG″ are computationally equivalent. Thus to prove Theorem 6.4.1, it suffices to prove that OPT-CLUST is NP-complete.

6.4.4 Open Ball Scaling

To prove NP-completeness of OPT-CLUST, we first put OPT-CLUST into a polynomially equivalent form OPT-CLUST′.

Problem 6.4.9 [OPT-CLUST′] Under the assumptions of Theorem 6.4.1, for given $K \in \mathbb{R}_+$ and $\boldsymbol{\theta} \in \mathcal{H}^*$ decide if there exist ξ and $\boldsymbol{d} \in \mathbb{R}^m$ such that

$$\|\pi(\xi, \boldsymbol{d}) - \boldsymbol{\theta}\|^2 \leq K$$

subject to $\pi(\xi, \boldsymbol{d}) \in \mathcal{H}^*$.

OPT-CLUST′ is a relaxation of OPT-CLUST—as opposed to specializations carried out in previous transformations—where $\boldsymbol{\theta}$ and $\pi(\xi, \boldsymbol{d})$ are allowed to be in $\mathcal{H}^*$. Therefor the equivalence reduction between OPT-CLUST and OPT-CLUST′ is more complex.

Lemma 6.4.10 *OPT-CLUST and OPT-CLUST′ are computationally equivalent.*

Before proving Lemma 6.4.10, we give Proposition 6.4.11 which is needed in the proof of the lemma. The proposition says: performing optimal clustering, given a target performance vector on $\mathcal{H}^*$ that is far away from $\boldsymbol{q}$, is equivalent to performing optimal clustering with respect to a target performance vector—actually, a continuum of vectors on the line connecting the original target vector and $\boldsymbol{q}$—arbitrarily close to $\boldsymbol{q}$. The scaling is determined by the parameter $c > 0$ in the proposition.

Proposition 6.4.11 *Given $c \in \mathbb{R}_+$, $K \in \mathbb{R}_+$, $\boldsymbol{\theta} \in \mathbb{R}^n_+$, and $\boldsymbol{\rho} \in \mathbb{R}^n_+$, there exist ξ' and $\boldsymbol{d}' \in \mathbb{R}^m$ such that*

$$\|\pi(\xi', \boldsymbol{d}') - \boldsymbol{\theta}\|^2 \leq K, \quad \pi(\xi', \boldsymbol{d}') \circ \boldsymbol{\rho} = \boldsymbol{\theta} \circ \boldsymbol{\rho}$$

if and only if there exist ξ'' and $\boldsymbol{d}'' \in \mathbb{R}^m$ such that

$$\|\pi(\xi'', \boldsymbol{d}'') - (\boldsymbol{q} + \frac{\boldsymbol{\theta} - \boldsymbol{q}}{c})\|^2 \leq \frac{K}{c^2}, \quad \pi(\xi'', \boldsymbol{d}'') \circ \boldsymbol{\rho} = (\boldsymbol{q} + \frac{\boldsymbol{\theta} - \boldsymbol{q}}{c}) \circ \boldsymbol{\rho}$$

where $q_i = \frac{\boldsymbol{\theta} \circ \boldsymbol{\rho}}{\sum_{j=1}^n \rho_j}$ for $i = 1, \cdots, n$.

Proof. ($\Rightarrow$) Let $\xi'' = \xi'$ and $d_i'' = q + \frac{d_i' - q}{c}$ for $i = 1, \cdots, m$ where $q = \frac{\boldsymbol{\theta} \circ \boldsymbol{\rho}}{\sum_{j=1}^{n} \rho_j}$. Then

$$\pi(\xi'', \boldsymbol{d}'') = \boldsymbol{q} + \frac{\pi(\xi', \boldsymbol{d}') - \boldsymbol{q}}{c}.$$

We have

$$\begin{aligned} \|\pi(\xi'', \boldsymbol{d}'') - (\boldsymbol{q} + \frac{\boldsymbol{\theta} - \boldsymbol{q}}{c})\|^2 &= \|\boldsymbol{q} + \frac{\pi(\xi', \boldsymbol{d}') - \boldsymbol{q}}{c} - (\boldsymbol{q} + \frac{\boldsymbol{\theta} - \boldsymbol{q}}{c})\|^2 \\ &= \|\frac{\pi(\xi', \boldsymbol{d}') - \boldsymbol{\theta}}{c}\|^2 \leq \frac{K}{c^2}, \end{aligned}$$

and

$$\pi(\xi'', \boldsymbol{d}'') \circ \boldsymbol{\rho} = \left(\boldsymbol{q} + \frac{\pi(\xi', \boldsymbol{d}') - \boldsymbol{q}}{c}\right) \circ \boldsymbol{\rho} = \left(\boldsymbol{q} + \frac{\boldsymbol{\theta} - \boldsymbol{q}}{c}\right) \circ \boldsymbol{\rho}.$$

($\Leftarrow$) Let $\xi' = \xi''$ and $d_i' = q + c(d_i'' - q)$ for $i = 1, \cdots, m$. Then

$$\pi(\xi', \boldsymbol{d}') = \boldsymbol{q} + c(\pi(\xi'', \boldsymbol{d}'') - \boldsymbol{q}).$$

Analogous to the if-part of the proof, it is straightforward to verify that ξ' and $\boldsymbol{d}'$ satisfy conditions $\|\pi(\xi', \boldsymbol{d}') - \boldsymbol{\theta}\|^2 \leq K$ and $\pi(\xi', \boldsymbol{d}') \circ \boldsymbol{\rho} = \boldsymbol{\theta} \circ \boldsymbol{\rho}$. ■

Proof of Lemma 6.4.10. (i) Given an instance of OPT-CLUST$'$ specified by K' and $\boldsymbol{\theta}'$, construct an instance of OPT-CLUST as follows: K and $\boldsymbol{\theta}$ are defined as

$$K = \frac{K'}{c^2}, \qquad \boldsymbol{\theta} = \boldsymbol{q} + \frac{\boldsymbol{\theta}' - \boldsymbol{q}}{c}$$

where $c = \sqrt{3\|\boldsymbol{\theta}' - \boldsymbol{q}\|^2 / r}$. We have $\boldsymbol{\theta} \circ \boldsymbol{\rho} = \boldsymbol{q} \circ \boldsymbol{\rho}$ and $\|\boldsymbol{\theta} - \boldsymbol{q}\|^2 = r/3$. Thus $\boldsymbol{\theta} \in \mathcal{B}_{r/2}(\boldsymbol{q}) \cap \mathcal{H}^*$. We will show: $\exists\ \xi'$ and $\boldsymbol{d}'$ such that

$$\|\pi(\xi', \boldsymbol{d}') - \boldsymbol{\theta}'\|^2 \leq K', \qquad \pi(\xi', \boldsymbol{d}') \circ \boldsymbol{\rho} = \boldsymbol{\theta}' \circ \boldsymbol{\rho} \tag{6.4.12}$$

if and only if there exist ξ and $\boldsymbol{d}$ such that

$$\|\pi(\xi, \boldsymbol{d}) - \boldsymbol{\theta}\|^2 \leq K, \qquad \pi(\xi, \boldsymbol{d}) \in \mathcal{H}_a. \tag{6.4.13}$$

Suppose ξ' and $\boldsymbol{d}'$ satisfy (6.4.12). By Proposition 6.4.11, $\exists\, \xi''$ and $\boldsymbol{d}''$ such that

$$\|\pi(\xi'', \boldsymbol{d}'') - \boldsymbol{\theta}\|^2 \leq K, \qquad \pi(\xi'', \boldsymbol{d}'') \circ \boldsymbol{\rho} = \boldsymbol{\theta} \circ \boldsymbol{\rho}. \tag{6.4.14}$$

Of those solutions satisfying (6.4.14), choose ξ^*, $\boldsymbol{d}^*$ such that $\|\pi(\xi^*, \boldsymbol{d}^*) - \boldsymbol{\theta}\|^2 \leq \|\boldsymbol{q} - \boldsymbol{\theta}\|^2$. To show that $\pi(\xi^*, \boldsymbol{d}^*) \in \mathcal{H}_a$, it suffices to establish that $\pi(\xi^*, \boldsymbol{d}^*)$ is within distance r of $\boldsymbol{q}$. We have

$$\begin{aligned} \|\pi(\xi^*, \boldsymbol{d}^*) - \boldsymbol{q}\|^2 &\leq \|\pi(\xi^*, \boldsymbol{d}^*) - \boldsymbol{\theta}\|^2 + \|\boldsymbol{\theta} - \boldsymbol{q}\|^2 \\ &\leq 2\|\boldsymbol{\theta} - \boldsymbol{q}\|^2 = \frac{2}{3}r, \end{aligned}$$

from which (6.4.13) follows. Conversely, if ξ and $\boldsymbol{d}$ satisfy (6.4.13), by Proposition 6.4.11, there exist ξ' and $\boldsymbol{d}'$ satisfying (6.4.12).

(ii) Given an instance K and $\boldsymbol{\theta}$ of OPT-CLUST, use the same instance as input to OPT-CLUST$'$. Suppose OPT-CLUST has a solution ξ, $\boldsymbol{d}$. Since $\mathcal{H}_a \subseteq \mathcal{H}^*$, ξ, $\boldsymbol{d}$ is also a solution to OPT-CLUST$'$. On the other hand, suppose OPT-CLUST$'$ has a solution ξ', $\boldsymbol{d}'$. By the same argument as in (i), we can find a solution ξ^*, $\boldsymbol{d}^*$ which satisfies (6.4.13). ■

The preceding results have shown that

$$\text{OPT-AGG} \overset{\text{P}}{\Longleftrightarrow} \text{OPT-AGG}' \overset{\text{P}}{\Longleftarrow} \text{OPT-AGG}'' \overset{\text{P}}{\Longleftrightarrow} \text{OPT-CLUST} \overset{\text{P}}{\Longleftrightarrow} \text{OPT-CLUST}'$$

where $A \overset{\text{P}}{\Longleftarrow} B$ denotes polynomial reduction from problem B to problem A. Since OPT-AGG is the decision problem corresponding to the optimization problem in Theorem 6.4.1, the main result is proved by showing OPT-CLUST$'$ is NP-complete.

Theorem 6.4.15 *OPT-CLUST$'$ is NP-complete.*

Section 6.5 is devoted to proving Theorem 6.4.15.

6.5 Complexity of Optimal Clustering

We formulate an unconstrained optimization version of OPT-CLUST$'$, called OPT-CLUST$''$, and show that they are equivalent. OPT-CLUST$''$ is then proved NP-complete in Section 6.5.2.

6.5.1 Unconstrained Optimization and Subspace Projection

Given $\boldsymbol{\theta}$, $\boldsymbol{\rho}$, and ξ, let $S_\xi^j = \{i : \xi(i) = j\}$, $j = 1, \cdots, m$ denote the set of flow indices mapped to j under classifier ξ. Define $\boldsymbol{e}_\xi = [e_\xi^1, \cdots, e_\xi^m]$ as $e_\xi^j = \frac{\sum_{\xi(i)=j} \theta_i}{|S_\xi^j|}$ for $j = 1, \cdots, m$, and $\boldsymbol{r}_\xi = [r_\xi^1, \cdots, r_\xi^m]$ as $r_\xi^j = \frac{\sum_{\xi(i)=j} \rho_i}{|S_\xi^j|}$ for $j = 1, \cdots, m$. Let $\boldsymbol{E}_\xi = \pi(\xi, \boldsymbol{e}_\xi)$ and $\boldsymbol{R}_\xi = \pi(\xi, \boldsymbol{r}_\xi)$.

Problem 6.5.1 [OPT-CLUST$''$] Under the assumptions of Theorem 6.4.1, for given $K \in \mathbb{R}_+$ and $\boldsymbol{\theta} \in \mathcal{H}^*$ decide if there exists ξ such that

$$\|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 + \frac{(\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho})^2}{\|\boldsymbol{R}_\xi\|^2} \leq K.$$

Lemma 6.5.2 *OPT-CLUST$'$ and OPT-CLUST$''$ are computationally equivalent.*

Lemma 6.5.2 is proved with the help of the next proposition which states that for a given classifier ξ, the optimal performance vector $\boldsymbol{w}_\xi$ which resides in the subspace $\mathcal{H}^* \cap \mathcal{G}_\xi$ can be calculated in $O(n^2)$ time.

Proposition 6.5.3 *Given ξ, $\boldsymbol{\theta} \in \mathbb{R}_+^n$, $\boldsymbol{\rho} \in \mathbb{R}_+^n$, the solution to*

$$\min_{\boldsymbol{d}} \|\pi(\xi, \boldsymbol{d}) - \boldsymbol{\theta}\|^2, \tag{6.5.4}$$

subject to $\pi(\xi, \boldsymbol{d}) \circ \boldsymbol{\rho} = \boldsymbol{\theta} \circ \boldsymbol{\rho}$ is given by

$$\boldsymbol{d}_\xi = \boldsymbol{e}_\xi - \boldsymbol{r}_\xi \frac{\sum_{j=1}^m |S_\xi^j| e_\xi^j r_\xi^j - \sum_{i=1}^n \theta_i \rho_i}{\sum_{j=1}^m |S_\xi^j| (r_\xi^j)^2}. \tag{6.5.5}$$

Furthermore, let $\boldsymbol{w}_\xi = \pi(\xi, \boldsymbol{d}_\xi)$. Then

$$\boldsymbol{w}_\xi = \boldsymbol{E}_\xi - \boldsymbol{R}_\xi \frac{\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2}, \tag{6.5.6}$$

and

$$\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 = \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 + \frac{(\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho})^2}{\|\boldsymbol{R}_\xi\|^2}. \tag{6.5.7}$$

Proof. Given ξ, we can use Lagrange multipliers to get $\boldsymbol{d}_\xi$. Rewrite (6.5.4) and $\pi(\xi, \boldsymbol{d}) \circ \boldsymbol{\rho} = \boldsymbol{\theta} \circ \boldsymbol{\rho}$ as

$$\min_{\boldsymbol{d}} \sum_{j=1}^{m} \sum_{\xi(i)=j} (\theta_i - d_j)^2, \qquad \sum_{j=1}^{m} d_j \sum_{\xi(i)=j} \rho_i - \boldsymbol{\theta} \circ \boldsymbol{\rho} = 0$$

respectively. The Lagrangian is given by

$$L(\boldsymbol{d}, \lambda) = \sum_{j=1}^{m} \sum_{\xi(i)=j} (\theta_i - d_j)^2 + \lambda \Big(\sum_{j=1}^{m} d_j \sum_{\xi(i)=j} \rho_i - \boldsymbol{\theta} \circ \boldsymbol{\rho} \Big).$$

We have

$$\frac{\partial L}{\partial d_j} = -\sum_{\xi(i)=j} 2(\theta_i - d_j) + \lambda \sum_{\xi(i)=j} \rho_i = 0, \qquad j = 1, \cdots, m, \tag{6.5.8}$$

$$\frac{\partial L}{\partial \lambda} = \sum_{j=1}^{m} d_j \sum_{\xi(i)=j} \rho_i - \boldsymbol{\theta} \circ \boldsymbol{\rho} = 0. \tag{6.5.9}$$

From (6.5.8),

$$d_j = \frac{\sum_{\xi(i)=j} \theta_i}{|S_\xi^j|} - \lambda \frac{\sum_{\xi(i)=j} \rho_i}{2|S_\xi^j|} = e_\xi^j - \frac{\lambda}{2} r_\xi^j, \qquad j = 1, \cdots, m. \tag{6.5.10}$$

Substitute (6.5.10) into (6.5.9),

$$\lambda = \frac{2 \sum_{j=1}^{m} e_\xi^j \sum_{\xi(i)=j} \rho_i - 2\boldsymbol{\theta} \circ \boldsymbol{\rho}}{\sum_{j=1}^{m} r_\xi^j \sum_{\xi(i)=j} \rho_i} = \frac{2 \sum_{j=1}^{m} |S_\xi^j| e_\xi^j r_\xi^j - 2 \sum_{i=1}^{n} \theta_i \rho_i}{\sum_{j=1}^{m} |S_\xi^j| (r_\xi^j)^2}.$$

Thus

$$d_j = e_\xi^j - r_\xi^j \frac{\sum_{j=1}^{m} |S_\xi^j| e_\xi^j r_\xi^j - \sum_{i=1}^{n} \theta_i \rho_i}{\sum_{j=1}^{m} |S_\xi^j| (r_\xi^j)^2}, \qquad j = 1, \cdots, m. \tag{6.5.11}$$

Note that (6.5.5) is a vector version of (6.5.11). By the definition of π, (6.5.6) follows from (6.5.5). For (6.5.7),

$$\begin{aligned}\|\boldsymbol{\theta}-\boldsymbol{w}_\xi\|^2 &= \Big\|\boldsymbol{\theta}-\boldsymbol{E}_\xi+\boldsymbol{R}_\xi\frac{\boldsymbol{E}_\xi\circ\boldsymbol{\rho}-\boldsymbol{\theta}\circ\boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2}\Big\|^2\\ &= \|\boldsymbol{\theta}-\boldsymbol{E}_\xi\|^2+\|\boldsymbol{R}_\xi\|^2\Big(\frac{\boldsymbol{E}_\xi\circ\boldsymbol{\rho}-\boldsymbol{\theta}\circ\boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2}\Big)^2\\ &\quad+2(\boldsymbol{\theta}-\boldsymbol{E}_\xi)\circ\boldsymbol{R}_\xi\frac{\boldsymbol{E}_\xi\circ\boldsymbol{\rho}-\boldsymbol{\theta}\circ\boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2}\\ &= \|\boldsymbol{\theta}-\boldsymbol{E}_\xi\|^2+\frac{(\boldsymbol{E}_\xi\circ\boldsymbol{\rho}-\boldsymbol{\theta}\circ\boldsymbol{\rho})^2}{\|\boldsymbol{R}_\xi\|^2}.\end{aligned}$$

■

Proof of Lemma 6.5.2. Consider an instance specified by θ and K for both OPT-CLUST$'$ and OPT-CLUST$''$ (i) Suppose OPT-CLUST$'$ has a solution ξ, $\boldsymbol{d}$. According to the optimality of $\boldsymbol{w}_\xi$ described in Proposition 6.5.3, $\|\boldsymbol{w}_\xi-\boldsymbol{\theta}\|^2\le\|\pi(\xi,\boldsymbol{d})-\boldsymbol{\theta}\|^2\le K$. Thus ξ is a solution to OPT-CLUST$''$. (ii) Suppose OPT-CLUST$''$ has a solution ξ. We have $\|\boldsymbol{w}_\xi-\boldsymbol{\theta}\|^2\le K$ and $\boldsymbol{w}_\xi\circ\boldsymbol{\rho}=\boldsymbol{\theta}\circ\boldsymbol{\rho}$. Thus ξ, $\boldsymbol{d}_\xi$ is a solution of OPT-CLUST$'$. ■

From a geometric perspective, the optimal performance vector $\boldsymbol{w}_\xi$ for a given classifier is the orthogonal projection of $\boldsymbol{\theta}$ on the subspace $\mathcal{H}^*\cap\mathcal{G}_\xi$. This orthogonal relationship is captured by the next proposition which will be used in the proof of Theorem 6.5.13 in the next section.

Proposition 6.5.12 *Given $\boldsymbol{\theta}\in\mathbb{R}^n_+$, $\boldsymbol{\rho}\in\mathbb{R}^n_+$, and ξ, for all $\boldsymbol{w}$ satisfying $\boldsymbol{w}\in G_\xi$ and $\boldsymbol{w}\circ\boldsymbol{\rho}=\boldsymbol{\theta}\circ\boldsymbol{\rho}$,*

$$\|\boldsymbol{\theta}-\boldsymbol{w}\|^2=\|\boldsymbol{\theta}-\boldsymbol{w}_\xi\|^2+\|\boldsymbol{w}_\xi-\boldsymbol{w}\|^2$$

where $\boldsymbol{w}_\xi$ is defined by (6.5.6).

Proof. Since $\boldsymbol{w}\in G_\xi$, $\exists\,\boldsymbol{d}\in\mathbb{R}^m$ such that $\boldsymbol{w}=\pi(\xi,\boldsymbol{d})$. We have

$$(\boldsymbol{\theta}-\boldsymbol{E}_\xi)\circ(\boldsymbol{w}_\xi-\boldsymbol{w})=\sum_{j=1}^m\sum_{\xi(i)=j}(\theta_i-e^j_\xi)(d^j_\xi-d_j)=0$$

since $\sum_{\xi(i)=j}(\theta_i - e^j_\xi) = 0$ and

$$\boldsymbol{R}_\xi \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) = \sum_{j=1}^{m} \sum_{\xi(i)=j} r^j_\xi (d^j_\xi - d_j) = \sum_{j=1}^{m} \sum_{\xi(i)=j} \rho_i (d^j_\xi - d_j) = \boldsymbol{\rho} \circ \boldsymbol{w}_\xi - \boldsymbol{\rho} \circ \boldsymbol{w} = 0.$$

Thus,

$$\begin{aligned}
(\boldsymbol{\theta} - \boldsymbol{w}_\xi) \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) &= \Big(\boldsymbol{\theta} - \boldsymbol{E}_\xi + \boldsymbol{R}_\xi \frac{\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2}\Big) \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) \\
&= (\boldsymbol{\theta} - \boldsymbol{E}_\xi) \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) + \frac{\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho}}{\|\boldsymbol{R}_\xi\|^2} \boldsymbol{R}_\xi \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) \\
&= 0.
\end{aligned}$$

We have

$$\begin{aligned}
\|\boldsymbol{\theta} - \boldsymbol{w}\|^2 &= \|\boldsymbol{\theta} - \boldsymbol{w}_\xi + \boldsymbol{w}_\xi - \boldsymbol{w}\|^2 \\
&= \|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 + \|\boldsymbol{w}_\xi - \boldsymbol{w}\|^2 + 2(\boldsymbol{\theta} - \boldsymbol{w}_\xi) \circ (\boldsymbol{w}_\xi - \boldsymbol{w}) \\
&= \|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 + \|\boldsymbol{w}_\xi - \boldsymbol{w}\|^2.
\end{aligned}$$

■

6.5.2 Hardness of One-dimensional Clustering

Theorem 6.5.13 *OPT-CLUST″ is NP-complete.*

To prove NP-completeness of OPT-CLUST″, we will reduce the following problem—a variant of PARTITION known to be NP-complete [29]—to OPT-CLUST″.

Problem 6.5.14 [EVEN-PARTITION] Given $X_1, \cdots, X_n$, $X_i \in \mathbb{N}$, $i = 1, \cdots, n$, n is even, is there a subset $S \subseteq \{1, \cdots, n\}$ with cardinality $n/2$ such that

$$\sum_{i \in S} X_i = \frac{1}{2} \sum_{i=1}^{n} X_i.$$

The next proposition is used in the reduction from EVEN-PARTITION to OPT-CLUST″ in the proof of Theorem 6.5.13.

Proposition 6.5.15 *Given real numbers $w'_1, \cdots, w'_k$, $\rho'_1, \cdots, \rho'_k$ and $w''_1, \cdots, w''_\ell$, $\rho''_1, \cdots, \rho''_\ell$ with mean $\bar{w}'$, $\bar{\rho}'$, $\bar{w}''$, $\bar{\rho}''$, respectively, and k, $\ell \geq 1$, define $w_1, \cdots, w_n$, $\rho_1, \cdots, \rho_n$, where $n = k + \ell$, as follows:*

$$w_i = \begin{cases} w'_i \quad , & 1 \leq i \leq k, \\ w''_{i-k}\,, & k+1 \leq i \leq n, \end{cases}$$

$$\rho_i = \begin{cases} \rho'_i \quad , & 1 \leq i \leq k, \\ \rho''_{i-k}\,, & k+1 \leq i \leq n. \end{cases}$$

Let $\bar{w}$, $\bar{\rho}$ be the mean of the w_i and ρ_i's, respectively. Then

$$\sum_{i=1}^{n}(w_i - \bar{w})^2 = \sum_{i=1}^{k}(w'_i - \bar{w}')^2 + \sum_{i=1}^{\ell}(w''_i - \bar{w}'')^2 + \frac{k\ell}{k+\ell}(\bar{w}'' - \bar{w}')^2, \tag{6.5.16}$$

$$\sum_{i=1}^{n}(w_i - \bar{w})\rho_i = \sum_{i=1}^{k}(w'_i - \bar{w}')\rho'_i + \sum_{i=1}^{\ell}(w''_i - \bar{w}'')\rho''_i + \frac{k\ell}{k+\ell}(\bar{w}'' - \bar{w}')(\bar{\rho}'' - \bar{\rho}'). \tag{6.5.17}$$

Proof.

$$\begin{aligned}
\sum_{i=1}^{n}(w_i - \bar{w})^2 &= \sum_{i=1}^{k}(w'_i - \bar{w})^2 + \sum_{i=1}^{\ell}(w''_i - \bar{w})^2 \\
&= \sum_{i=1}^{k}(w'_i - \bar{w}')^2 + k(\bar{w}' - \bar{w})^2 + \sum_{i=1}^{\ell}(w''_i - \bar{w}'')^2 + \ell(\bar{w}'' - \bar{w})^2 \\
&= \sum_{i=1}^{k}(w'_i - \bar{w}')^2 + k\big(\frac{\ell}{n}(\bar{w}' - \bar{w}'')\big)^2 \\
&\quad + \sum_{i=1}^{\ell}(w''_i - \bar{w}'')^2 + \ell\big(\frac{k}{n}(\bar{w}'' - \bar{w}')\big)^2 \\
&= \sum_{i=1}^{k}(w'_i - \bar{w}')^2 + \sum_{i=1}^{\ell}(w''_i - \bar{w}'')^2 + \frac{k\ell}{k+\ell}(\bar{w}'' - \bar{w}')^2.
\end{aligned}$$

$$\begin{aligned}
\sum_{i=1}^{n}(w_i-\bar{w})\rho_i &= \sum_{i=1}^{k}(w_i'-\bar{w})\rho_i' + \sum_{i=1}^{\ell}(w_i''-\bar{w})\rho_i'' \\
&= \sum_{i=1}^{k}(w_i'-\bar{w}')\rho_i' + \sum_{i=1}^{\ell}(w_i''-\bar{w}'')\rho_i'' \\
&\quad + \sum_{i=1}^{k}(\bar{w}'-\bar{w})\rho_i' + \sum_{i=1}^{\ell}(\bar{w}''-\bar{w})\rho_i'' \\
&= \sum_{i=1}^{k}(w_i'-\bar{w}')\rho_i' + \sum_{i=1}^{\ell}(w_i''-\bar{w}'')\rho_i'' \\
&\quad + \sum_{i=1}^{k}\frac{\ell}{n}(\bar{w}'-\bar{w}'')\rho_i' + \sum_{i=1}^{\ell}\frac{k}{n}(\bar{w}''-\bar{w}')\rho_i'' \\
&= \sum_{i=1}^{k}(w_i'-\bar{w}')\rho_i' + \sum_{i=1}^{\ell}(w_i''-\bar{w}'')\rho_i'' + \frac{k\ell}{k+\ell}(\bar{w}''-\bar{w}')(\bar{\rho}''-\bar{\rho}').
\end{aligned}$$

■

Proof of Theorem 6.5.13. It is easy to check that the problem is in NP. We will reduce EVEN-PARTITION to OPT-CLUST″. Given an instance of EVEN-PARTITION specified by $X_1, \cdots, X_\ell$, let X_{max}, X_{min} denote the maximum and minimum, $k = \frac{\ell}{2}(\ell-1)+1$, and let

$$A(\ell_1,\ell_2) = \left(\frac{(\ell_1^2+\ell_2^2)k+\ell_1\ell_2\ell}{2(k+\frac{\ell}{2})}\right)^{\frac{1}{2}}, \qquad 1 \le \ell_1,\ell_2 \le \ell,$$

$$C = \max_{\ell_1\neq\ell_2} \frac{\left(\frac{\ell_2}{k+\ell_2} + \frac{\ell_1-\ell_2}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)\right)X_{max} - \frac{\ell_1}{k+\ell_1}X_{min} + \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}(1+A(\ell_1,\ell_2))}{\frac{\ell_1-\ell_2}{(k+\ell_1)(k+\ell_2)}(k-A(\ell_1,\ell_2))} + 1.$$

Construct an instance of OPT-CLUST″ specified by n, m, K, $\boldsymbol{\rho}$, $\boldsymbol{\theta}$ as follows:

$$n = 2k+\ell, \quad m = 2, \quad K = \frac{k\,\ell}{k+\frac{\ell}{2}}, \quad \rho_i = \rho_i'/D,$$

$$\theta_i = \begin{cases} 1\,, & 1 \le i \le k, \\ 2\,, & k+1 \le i \le k+\ell, \\ 3\,, & k+\ell+1 \le i \le n, \end{cases}$$

where

$$D \;=\; \sum_{i=1}^{n} \rho_i' + 1 \quad \text{and} \quad \rho_i' \;=\; \begin{cases} 1 & , \quad 1 \le i \le k, \\ C + X_{i-k}\,, & k+1 \le i \le k+\ell, \\ 1 & , \quad k+\ell+1 \le i \le n. \end{cases}$$

We shall show: $\exists\, S \subseteq \{1, \cdots, \ell\}$, $|S| = \frac{\ell}{2}$ and $\sum_{i\in S} X_i = \frac{1}{2}\sum_{i=1}^{\ell} X_i$, if and only if $\exists\, \xi$ such that

$$\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 = \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 + \frac{(\boldsymbol{E}_\xi \circ \boldsymbol{\rho} - \boldsymbol{\theta} \circ \boldsymbol{\rho})^2}{\|\boldsymbol{R}_\xi\|^2} \le K$$

where $\boldsymbol{w}_\xi$ is defined by (6.5.6).

($\Rightarrow$) Suppose $\exists\, S \subseteq \{1, \cdots, \ell\}$, $|S| = \frac{\ell}{2}$, and $\sum_{i\in S} X_i = \frac{1}{2}\sum_{i=1}^{\ell} X_i$. Define ξ as

$$\xi(i) \;=\; \begin{cases} 1\,, & i \in \{1, \cdots, k\} \cup \{i : i-k \in S\}, \\ 2\,, & i \in \{k+\ell+1, \cdots, n\} \cup \{i : i-k \notin S\}. \end{cases}$$

Using (6.5.16) from Proposition 6.5.15, we have

$$\begin{aligned} \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 \;&=\; \sum_{i\in S_\xi^1} (\theta_i - e_\xi^1)^2 + \sum_{i\in S_\xi^2} (\theta_i - e_\xi^2)^2 \\ &=\; \frac{k\,\frac{\ell}{2}}{k+\frac{\ell}{2}}(2-1)^2 + \frac{\frac{\ell}{2}\,k}{\frac{\ell}{2}+k}(3-2)^2 \;=\; \frac{k\,\ell}{k+\frac{\ell}{2}}. \end{aligned}$$

Let $y = \frac{\sum_{i=1}^{\ell} X_i}{\ell}$, then $\frac{\sum_{i\in S} X_i}{|S|} = \frac{\sum_{i\notin S} X_i}{\ell - |S|} = y$. Using (6.5.17), we have

$$\begin{aligned} \boldsymbol{\theta} \circ \boldsymbol{\rho} - \boldsymbol{E}_\xi \circ \boldsymbol{\rho} \;&=\; \sum_{i\in S_\xi^1} (\theta_i - e_\xi^1)\rho_i + \sum_{i\in S_\xi^2} (\theta_i - e_\xi^2)\rho_i \\ &=\; \frac{k\,\frac{\ell}{2}}{k+\frac{\ell}{2}}\,(2-1)\,\frac{C+y-1}{D} + \frac{\frac{\ell}{2}\,k}{\frac{\ell}{2}+k}\,(3-2)\,\frac{1-C-y}{D} \;=\; 0\,. \end{aligned}$$

Thus $\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 = \frac{k\,\ell}{k+\frac{\ell}{2}} \le K$.

($\Leftarrow$) Suppose there is no $S \subseteq \{1, \cdots, \ell\}$ that satisfies $|S| = \frac{\ell}{2}$ and $\sum_{i\in S} X_i = \frac{1}{2}\sum_{i=1}^{\ell} X_i$. We will show for all ξ, $\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 > K$. ξ can be expressed as

$$\xi(i) \;=\; \begin{cases} 1\,, & i \in S_a \cup S_b \cup S_c, \\ 2\,, & i \in S_a' \cup S_b' \cup S_c', \end{cases} \tag{6.5.18}$$

where $S_a \cup S'_a = \{1, \cdots, k\}$, $S_b \cup S'_b = \{k+1, \cdots, k+\ell\}$, and $S_c \cup S'_c = \{k+\ell+1, \cdots, n\}$. Thus, all ξ must fall into one of the following cases:

(a) $S_a = \{1, \cdots, k\}$, $S_c = \emptyset$, $|S_b| = \frac{\ell}{2}$;

(b) $S_a = \{1, \cdots, k\}$, $S_c = \emptyset$, $|S_b| \neq \frac{\ell}{2}$;

(c) $\emptyset \subsetneq S_a \subseteq \{1, \cdots, k\}$, $\emptyset \subsetneq S_c \subseteq \{k+\ell+1, \cdots, n\}$;

(d) $\emptyset \subsetneq S_a \subsetneq \{1, \cdots, k\}$, $S_c = \emptyset$;

(e) $S_a = \emptyset$.

Note that (d) follows from (c), and (e) follows from (a), (b) and (c). We will show that $\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 > K$ holds for (a), (b), and (c).

Case (a). Let $y_1 = \frac{\sum_{i+k \in S_b} X_i}{|S_b|}$, $y_2 = \frac{\sum_{i+k \in S'_b} X_i}{|S'_b|}$, and $r_b = \frac{\sum_{i \in S_b} \rho_i}{|S_b|} = \frac{C+y_1}{D}$, $r'_b = \frac{\sum_{i \in S'_b} \rho_i}{|S'_b|} = \frac{C+y_2}{D}$. By the contrapositive hypothesis, $y_1 \neq y_2$, thus $r_b \neq r'_b$. Using (6.5.16) and (6.5.17), we have

$$\begin{aligned} \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 &= \sum_{i \in S^1_\xi} (\theta_i - e^1_\xi)^2 + \sum_{i \in S^2_\xi} (\theta_i - e^2_\xi)^2 \\ &= \frac{k\,\frac{\ell}{2}}{k + \frac{\ell}{2}}(2-1)^2 + \frac{\frac{\ell}{2}\,k}{\frac{\ell}{2} + k}(3-2)^2 = \frac{k\,\ell}{k + \frac{\ell}{2}}, \end{aligned} \tag{6.5.19}$$

and

$$\begin{aligned} \boldsymbol{\theta} \circ \boldsymbol{\rho} - \boldsymbol{E}_\xi \circ \boldsymbol{\rho} &= \sum_{i \in S^1_\xi} (\theta_i - e^1_\xi)\rho_i + \sum_{i \in S^2_\xi} (\theta_i - e^2_\xi)\rho_i \\ &= \frac{k\,\frac{\ell}{2}}{k + \frac{\ell}{2}}(2-1)(r_b - 1) + \frac{\frac{\ell}{2}\,k}{\frac{\ell}{2} + k}(3-2)(1 - r'_b) \\ &= \frac{k\,\frac{\ell}{2}}{k + \frac{\ell}{2}}(r_b - r'_b) \neq 0. \end{aligned}$$

Thus $\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 > \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 = \frac{k\ell}{k+\frac{\ell}{2}} = K$.

Case (b). Let $|S_b| = \ell_1$, $|S_b'| = \ell_2$, $y_1 = \frac{\sum_{i+k\in S_b} X_i}{\ell_1}$, $y_2 = \frac{\sum_{i+k\in S_b'} X_i}{\ell_2}$, and $r_b = \frac{\sum_{i\in S_b}\rho_i}{\ell_1} = \frac{C+y_1}{D}$, $r_b' = \frac{\sum_{i\in S_b'}\rho_i}{\ell_2} = \frac{C+y_2}{D}$. We have $\ell_1 \neq \ell_2$. Without loss of generality, assume $\ell_1 > \ell_2$. Using (6.5.16),

$$\|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 = \frac{k\,\ell_1}{k+\ell_1} + \frac{k\,\ell_2}{k+\ell_2}. \tag{6.5.20}$$

By the definition of C,

$$C > \frac{\frac{k\,\ell_2}{k+\ell_2}y_2 - \frac{k\,\ell_1}{k+\ell_1}y_1 + \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)X_{max} + \frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)} + \frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)}{\frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)} - \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)}.$$

Since $k = \frac{\ell}{2}(\ell-1)+1$, $k - A(\ell_1,\ell_2) > 0$. Hence,

$$\begin{aligned}\Big(\frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)} - \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)\Big)C \quad &> \quad \frac{k\,\ell_2}{k+\ell_2}y_2 - \frac{k\,\ell_1}{k+\ell_1}y_1 \\ + \ \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2)X_{max} \ &+ \ \frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)} \ + \ \frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}A(\ell_1,\ell_2),\end{aligned}$$

Noting $\frac{k^2(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)} = \frac{k\,\ell_1}{k+\ell_1} - \frac{k\,\ell_2}{k+\ell_2}$, we have

$$\begin{aligned}\Big(\frac{k\,\ell_1}{k+\ell_1} - \frac{k\,\ell_2}{k+\ell_2}\Big)(C-1) + \frac{k\,\ell_1}{k+\ell_1}y_1 - \frac{k\,\ell_2}{k+\ell_2}y_2 & \\ > \quad \frac{k(\ell_1-\ell_2)}{(k+\ell_1)(k+\ell_2)}\sqrt{\frac{(\ell_1^2+\ell_2^2)k+\ell_1\ell_2\ell}{2(k+\frac{\ell}{2})}}(C+X_{max}+k).&\end{aligned}$$

Divide both sides by $C + X_{max} + k > 0$ and square them. Since $\ell_1 - \ell_2 > 0$,

$$\begin{aligned}\frac{\big((\frac{k\,\ell_1}{k+\ell_1} - \frac{k\,\ell_2}{k+\ell_2})(C-1) + \frac{k\,\ell_1}{k+\ell_1}y_1 - \frac{k\,\ell_2}{k+\ell_2}y_2\big)^2}{(C+X_{max}+k)^2} & \\ > \frac{k^2(\ell_1-\ell_2)^2}{(k+\ell_1)^2(k+\ell_2)^2}\frac{(\ell_1^2+\ell_2^2)k+\ell_1\ell_2\ell}{2(k+\frac{\ell}{2})} & \\ = \frac{k^2(\ell_1-\ell_2)^2}{2(k+\ell_1)(k+\ell_2)(k+\frac{\ell}{2})}\Big(\frac{\ell_1^2}{k+\ell_1} + \frac{\ell_2^2}{k+\ell_2}\Big). &\end{aligned} \tag{6.5.21}$$

Thus

$$\begin{aligned}
\frac{(\boldsymbol{\theta}\circ\boldsymbol{\rho}-\boldsymbol{E}_\xi\circ\boldsymbol{\rho})^2}{\|\boldsymbol{R}\|^2} &= \frac{\big(\sum_{i\in S_\xi^1}(\theta_i-e_\xi^1)\rho_i+\sum_{i\in S_\xi^2}(\theta_i-e_\xi^2)\rho_i\big)^2}{\sum_{i\in S_\xi^1}(r_\xi^1)^2+\sum_{i\in S_\xi^2}(r_\xi^2)^2}\\
&= \frac{\big(\frac{k\,\ell_1}{k+\ell_1}(2-1)(r_b-1)+\frac{\ell_2\,k}{\ell_2+k}(3-2)(1-r_b')\big)^2}{\frac{(k+\ell_1 r_b)^2}{k+\ell_1}+\frac{(k+\ell_2 r_b')^2}{k+\ell_2}}\\
&= \frac{\big(\frac{k\,\ell_1}{k+\ell_1}(C+y_1-1)-\frac{k\,\ell_2}{k+\ell_2}(C+y_2-1)\big)^2}{\frac{(\ell_1(C+y_1)+k)^2}{k+\ell_1}+\frac{(\ell_2(C+y_2)+k)^2}{k+\ell_2}}\\
&\geq \frac{\big((\frac{k\,\ell_1}{k+\ell_1}-\frac{k\,\ell_2}{k+\ell_2})(C-1)+\frac{k\,\ell_1}{k+\ell_1}y_1-\frac{k\,\ell_2}{k+\ell_2}y_2\big)^2}{\frac{\ell_1^2(C+y_1+k)^2}{k+\ell_1}+\frac{\ell_2^2(C+y_2+k)^2}{k+\ell_2}}\\
&\geq \frac{\big((\frac{k\,\ell_1}{k+\ell_1}-\frac{k\,\ell_2}{k+\ell_2})(C-1)+\frac{k\,\ell_1}{k+\ell_1}y_1-\frac{k\,\ell_2}{k+\ell_2}y_2\big)^2}{\big(\frac{\ell_1^2}{k+\ell_1}+\frac{\ell_2^2}{k+\ell_2}\big)(C+X_{max}+k)^2}\\
&> \frac{k^2(\ell_1-\ell_2)^2}{2(k+\ell_1)(k+\ell_2)(k+\frac{\ell}{2})}. \qquad (6.5.22)
\end{aligned}$$

The second equality follows from Proposition 6.5.15. The last inequality follows from (6.5.21).

From (6.5.20) and (6.5.22), we get

$$\begin{aligned}
\|\boldsymbol{\theta}-\boldsymbol{w}_\xi\|^2 &= \|\boldsymbol{\theta}-\boldsymbol{E}_\xi\|^2+\frac{(\boldsymbol{\theta}\circ\boldsymbol{\rho}-\boldsymbol{E}_\xi\circ\boldsymbol{\rho})^2}{\|\boldsymbol{R}\|^2}\\
&> \frac{k\,\ell_1}{k+\ell_1}+\frac{k\,\ell_2}{k+\ell_2}+\frac{k^2(\ell_1-\ell_2)^2}{2(k+\ell_1)(k+\ell_2)(k+\frac{\ell}{2})}\\
&= \frac{(k^2\ell+2k\ell_1\ell_2)(2k+\ell)+k^2(\ell_1-\ell_2)^2}{2(k+\ell_1)(k+\ell_2)(k+\frac{\ell}{2})}\\
&= \frac{2k\ell(k^2+k\ell+\ell_1\ell_2)}{2(k+\ell_1)(k+\ell_2)(k+\frac{\ell}{2})}\\
&= \frac{k\ell}{k+\frac{\ell}{2}}.
\end{aligned}$$

Case (c). Let $|S_a| = k_1$, $|S'_a| = k_2$, $|S_b| = \ell_1$, $|S'_b| = \ell_2$, and $|S_c| = j_1$, $|S'_c| = j_2$. By (6.5.16),

$$\begin{aligned}
\|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 &= \sum_{i\in S^1_\xi}(\theta_i - e^1_\xi)^2 + \sum_{i\in S^2_\xi}(\theta_i - e^2_\xi)^2 \\
&= \frac{k_1 j_1}{k_1+j_1}2^2 + \frac{(k_1+j_1)\ell_1}{k_1+j_1+\ell_1}\Big(1 - \frac{2j_1}{k_1+j_1}\Big)^2 \\
&\quad + \frac{k_2 j_2}{k_2+j_2}2^2 + \frac{(k_2+j_2)\ell_2}{k_2+j_2+\ell_2}\Big(1 - \frac{2k_2}{k_2+j_2}\Big)^2 \\
&= 4\frac{k_1 j_1}{k_1+j_1} + \frac{\ell_1}{k_1+j_1+\ell_1}\frac{(k_1-j_1)^2}{k_1+j_1} + 4\frac{k_2 j_2}{k_2+j_2} + \frac{\ell_2}{k_2+j_2+\ell_2}\frac{(j_2-k_2)^2}{k_2+j_2}.
\end{aligned}$$

Without loss of generality, assume $k_1+j_1 \geq k_2+j_2$ and $k_1 \geq j_1$. Noting $k_1-j_1 = j_2-k_2$ since $k_1+k_2 = j_1+j_2 = k$, we have

$$\begin{aligned}
\|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 &\geq 4\frac{k_1 j_1}{k_1+j_1} + \frac{\ell_1}{k_1+j_1+\ell}\frac{(k_1-j_1)^2}{k_1+j_1} + 4\frac{k_2 j_2}{k_2+j_2} + \frac{\ell_2}{k_1+j_1+\ell}\frac{(k_1-j_1)^2}{k_1+j_1} \\
&= 4\frac{k_1 j_1}{k_1+j_1} + \frac{\ell}{k_1+j_1+\ell}\frac{(k_1-j_1)^2}{k_1+j_1} + 4\frac{k_2 j_2}{k_2+j_2}. \qquad (6.5.23)
\end{aligned}$$

Furthermore, $k_1 \geq j_1$ implies $k_2 \leq j_2$, thus

$$4\frac{k_2 j_2}{k_2+j_2} \geq \frac{k_2(\ell+2j_1)^2}{(k+j_1+\ell)(k_1+j_1+\ell)}. \qquad (6.5.24)$$

(6.5.23) and (6.5.24) imply

$$\begin{aligned}
\|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 &\geq 4\frac{k_1 j_1}{k_1+j_1} + \frac{\ell(k_1-j_1)^2}{(k_1+j_1+\ell)(k_1+j_1)} + \frac{k_2(\ell+2j_1)^2}{(k+j_1+\ell)(k_1+j_1+\ell)} \\
&= \frac{4k_1 j_1 + \ell(k_1+j_1)}{k_1+j_1+\ell} + \frac{k_2(\ell+2j_1)^2}{(k+j_1+\ell)(k_1+j_1+\ell)} \\
&= \frac{(k+j_1)\ell + 4kj_1}{k+j_1+\ell}.
\end{aligned}$$

Since $k > \frac{\ell}{2}(\ell-1)$, for all j_1, $1 \leq j_1 \leq k$, we have $\frac{(k+j_1)\ell+4kj_1}{k+j_1+\ell} > \frac{k\ell}{k+\frac{\ell}{2}}$. Therefore, $\|\boldsymbol{\theta} - \boldsymbol{w}_\xi\|^2 \geq \|\boldsymbol{\theta} - \boldsymbol{E}_\xi\|^2 > \frac{k\ell}{k+\frac{\ell}{2}}$. ■

The proof of Theorem 6.5.13 actually shows a stronger result: OPT-CLUST″ with constant m, in particular, $m = 2$, is already NP-complete. Thus the optimal aggregate-flow scheduling problem in Theorem 6.4.1, with OPT-AGG being its decision form, is NP-hard even for $m = 2$.

6.6 Conclusion and Discussion

In this chapter, we have extended Coffman and Mitrani's optimal per-flow scheduling framework [19] by considering aggregate-flow schedulers for general stochastic input under work-conserving, non-preemptive and non-anticipative schedulers. Whereas optimal per-flow scheduling subject to conservation laws is poly-time solvable—as is optimal aggregate-flow scheduling without conservation laws—we show that optimal aggregate-flow scheduling in multi-class $G/G/1$ systems subject to Kleinrock's conservation law is NP-hard.

There remain a number of open problems. They include:

Approximation. A natural question to explore is the approximability property of optimal aggregate-flow scheduling. The optimization version of PARTITION—more precisely, EVEN-PARTITION, which was used to show that OPT-CLUST$''$ is NP-complete—is known to have a polynomial time approximation scheme [1]. A heuristic approach is to solve the unconstrained optimization problem which can be done in cubic time using dynamic programming. We conjecture that OPT-CLUST$''$ is much harder to approximate than PARTITION, perhaps even not constant factor approximable.

Objective function. This chapter considered an MMSE objective function for both its practical relevance and its common use in optimization and control. Other objective functions of interest include the L_1-norm $\|\boldsymbol{w} - \boldsymbol{\theta}\|$, signed optimization problems corresponding to $\boldsymbol{w} \leq \boldsymbol{\theta}$ including one that counts how many user requirements are satisfied, proportional fairness, and linear objective functions. For a subset of these objective functions, we can show equivalence of optimal solutions in the per-flow performance space. In the aggregate-flow case, the optimal solution for linear objective functions depends on the right-hand-side of the inequalities of the strong conservation law. This stands in contrast with per-flow optimization of linear functions via linear programming which only depends on the n normalized coefficients of the objec-

tive function [23, 76]. We conjecture that optimal aggregate-flow scheduling remains NP-hard for the above functions.

Performance space relaxation. The open ball containment assumption (6.3.9) requires that all flows within a service class receive the same performance. It may be interesting to explore if the equality constraint can be meaningfully relaxed, for example, by requiring that any two flows belonging to the same service class receive performance that is at most ε apart. The hardness result would not be affected by an ε-relaxation as we have already proved NP-completeness for the more difficult special case $\varepsilon = 0$. However, for approximability relaxation may play a role.

Two-dimensional clustering. When transforming one-dimensional clustering with linear constraint into an unconstrained form enabled by Proposition 6.5.3, the consequent two-dimensional clustering problem bears resemblance to a simpler two-dimensional MMSE clustering problem: Given $\boldsymbol{\theta} \in \mathbb{R}^n \times \mathbb{R}^n$, find $\xi : \{1, \ldots, n\} \to \{1, \ldots, m\}$ and $\boldsymbol{d} \in \mathbb{R}^m \times \mathbb{R}^m$ that minimizes $\|\pi(\xi, \boldsymbol{d}) - \boldsymbol{\theta}\|^2$ where π is the corresponding two-dimensional extension. The complexity of unconstrained two-dimensional MMSE clustering remains open. We believe that the methods used in this chapter may be applied to show that the problem is NP-complete.

Aggregation and efficiency. In addition to optimality, another important aspect of aggregate-flow scheduling involves characterizing the loss of efficiency stemming from flow aggregation. In previous chapters where stochasticity of the input and conservation laws were not considered, a quantitative result on the loss of performance as a function of the degree of flow aggregation was given (cf. Chapter 3, 4). A similar exploration may be of interest for the generalized framework. The computational hardness of optimal aggregate-flow scheduling may have implications on finding effective characterizations of the performance loss, however, this needs not be necessarily the case.

7 CONCLUSION AND FUTURE WORK

7.1 Thesis Summary

This dissertation studies providing QoS to individual flows using aggregate-flow scheduling. Our work is carried out on both theoretical and practical sides.

In theory part, we present a theoretical framework to analyze network architectures based on aggregate-flow scheduling and study optimal aggregate-flow scheduling problem. We show that the optimal aggregate-flow scheduling problem in general queueing systems with stochastic input is NP-hard under MMSE QoS provisioning criterion. The complexity comes from the combination of conservation law and flow aggregation. Our result is an extension of the well studied stochastic optimization of per-flow schedulers. Under relative QoS differentiation objective function, we show the optimal aggregate-flow scheduling problem is poly-time solvable by dynamic programming and efficient linear algorithm exists in practical scenario where user's QoS requirements are coded in the ToS field of IP header. We study quantitative properties of the induced optimal aggregate-flow scheduling solution that are related to class differentiation and end-to-end QoS provisioning. The optimal solution and its properties jointly build theoretical foundation for PHB design in Diff-serv networks.

In practice part, we design an optimal aggregate-flow per-hop control that achieves the induced optimal aggregate-flow scheduling solution and implement the optimal per-hop control in Cisco routers. We conduct a comprehensive performance evaluation by both simulations and experiments over Q-bahn testbed comprised of Cisco routers running the implemented optimal per-hop control. The benchmarking results confirm our theoretical framework and analysis, and reveal further quantitative features of both structural and dynamical properties of the system. Our results, collectively, show that user-specified services can be efficiently and effectively achieved over networks

with optimal aggregate-flow per-hop control substrate when coupled with either open-loop or closed-loop (adaptive label control) edge control.

7.2 Future Work

Our study focuses on the core of networks with aggregate-flow scheduling as building block. There remain a number of open problems related to other components and system-wise properties of our QOS provisioning architecture. They include:

Many-switch system. The properties (A1), (A2), and (B) imposed on per-hop control play an essential role in our QoS provisioning framework. As discussed in Chapter 2, these three properties are extensible to WAN environment since if a property holds for any single per-hop control, it also holds for a sequence of hops as a composite function of individual hops. Thus our results based on the properties apply to both single-switch and many-switch systems. On the other hand, the quantitative features related to the three properties have only been studied in single switch case. The extension of quantitative analysis to many-switch case will further characterize system behavior and help design more effective and efficient end-to-end control schemes.

End-to-end control. The close loop end-to-end QoS control dynamically adjusts label values carried in the ToS fields of IP headers to meet user's QoS requirements. Our current scheme increases/decreases label values in accordance with congestion level without changing data rate. Another direction of end-to-end control is to adjust data rate during congestion. This can be achieved either by using TCP congestion control or by adding this functionality into close loop end-to-end QoS control. Setting label value and data rate are both necessary in real networks. Thus, in addition to studying individual control schemes, we need to analyze interaction between these two control schemes and design coordinating mechanism between them.

Game theory analysis. We study global resource allocation properties of the system in the non-cooperative game environment where each user is changing its label

value independently to maximize its utility. Our stability and optimality results are based on the nonemptiness of $\mathcal{A}^*$, the set of configurations where all user's QoS requirements are satisfied (cf. Section 3.2). The dynamical properties inside $\mathcal{A}^*$ is also characterized. The remaining issues on game theory analysis include system stability when $\mathcal{A}^*$ is empty, system dynamics outside $\mathcal{A}^*$, and the extended game theoretic structure with selfish service provider as an additional player.

Edge control implementation. In system building part, the most important task is to implement a QoS agent that performs edge control functionality (end-to-end control and access control). The QoS agent may reside at the edge of the network or in the end host on behalf of the network. The desired feature of QoS agent is application transparent, i.e. legacy applications can run without knowing QoS agent is setting the ToS fields of its outgoing packets. The development of QoS agent is the ongoing work carried out in the network system lab at Purdue University.

LIST OF REFERENCES

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation—Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[2] D. Bertsimas. The achievable region method in the optimal control of queueing systems; formulations, bounds and policies. *Queueing Systems*, 21(3–4):337–389, 1995.

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

[4] R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Effective envelopes: Statistical bounds on multiplexed traffic in packet networks. In *Proc. IEEE INFOCOM '00*, 2000.

[5] A. Brandt, P. Franken, and B. Lisek. *Stationary Stochastic Models*. John Wiley & Sons, 1990.

[6] P. Brucker. On the complexity of clustering problems. In R. Henn, B. Korte, and W. Oletti, editors, *Optimierung und Operations Research*, Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 1978.

[7] H. Chen and D. Yao. *Fundamentals of Queueing Networks*. Springer, 2001.

[8] 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.

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

[10] S. Chen, K. Park, and M. Sitharam. On the ordering properties of GPS routers for multi-class QoS provision. In *Proc. SPIE International Conference on Performance and Control of Network Systems*, pages 252–265, 1998.

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

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

[13] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: motivation, formulation, and example. *IEEE/ACM Trans. Networking*, 1(6):614–627, 1993.

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

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

[16] 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.

[17] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Res. Exper.*, 1:3–26, 1990.

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

[19] E. Coffman, Jr. and I. Mitrani. A characterization of waiting time performance realizable by single-server queues. *Operations Research*, 28(3):810–821, 1980.

[20] A. Elwalid and D. Mitra. Effective bandwidth of general Markovian traffic sources and admission control in high speed networks. *IEEE/ACM Trans. Networking*, 1(3), 1993.

[21] 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.

[22] A. Federgruen and H. Groenevelt. Characterization and optimization of achievable performance in general queueing systems. *Operations Research*, 36(5):733–741, 1988.

[23] A. Federgruen and H. Groenevelt. M/G/c queueing systems with multiple customer classes: Characterization and control of achievable performance under nonpreemptive priority rules. *Management Science*, 34(9):1121–1138, 1988.

[24] 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.

[25] D. Ferguson, C. Nikolaou, and Y. Yemini. An economy for flow control in computer networks. In *Proc. IEEE INFOCOM '89*, pages 110–118, 1989.

[26] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic algorithms for load balancing in distributed computer systems. In *Proc. 8th International Conference on Distributed Computing Systems*, pages 491–499, 1988.

[27] S. Gal and B. Klots. Optimal partitioning which maximizes the sum of the weighted averages. *Operations Research*, 43(3):500–508, 1995.

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

[29] M. Garey and D. Johnson. *Computers and Intractability*, page 223. W. H. Freeman and Company, 1979.

[30] E. Gelenbe and I. Mitrani. *Analysis and synthesis of computer systems.* Academic Press, New York, 1980.

[31] 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.

[32] L. Georgiadis and I. Viniotis. On the conservation law and the performance space of single server systems. *Operations Research*, 42(2):372–379, 1994.

[33] R. Gibbens. Traffic characterization and effective bandwidths for broadband network traces. In F. Kelly, S. Zachary, and I. Ziedins, editors, *Stochastic Networks: Theory and Applications*, pages 169–179. Clarendon Press, Oxford, 1996.

[34] R. J. Gibbens, S. K. Sargood, F .P. Kelly, H. Azmoodeh, R. Macfadyen, and N. Macfadyen. An approach to service level agreement for ip networks with differentiated services. In *Phil. Trans. Royal. Soc. Lond. A*, pages 2165–2182, 2000.

[35] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[36] A. Heddaya and K. Park. Parallel computing on high-speed wide-area networks: a pricing policy for its communication needs. In *Proc. 3rd IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, pages 188–191, 1995.

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

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

[39] L. Kleinrock. A conservation law for a wide class of queueing disciplines. *Naval Research Logistics Quarterly*, 12:181–192, 1965.

[40] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*, chapter 3, pages 113–118. Wiley-Interscience, New York, 1976.

[41] Y. Korilis and A. Lazar. Why is flow control hard: Optimality, fairness, partial and delayed information. In *Proc. 2nd ORSA Telecommunications Conference*, March 1992.

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

[43] Y. Korilis, A. Lazar, and A. Orda. Achieving network optima using Stackelberg routing strategies. *IEEE/ACM Trans. Networking*, 5(1):161–173, 1997.

[44] J. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. on Computers*, 38(5):705–717, 1989.

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

[46] 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.

[47] 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.

[48] J. Little. A proof for the queueing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.

[49] S. Low and P. Varaiya. An algorithm for optimal service provisioning using resource pricing. In *Proc. IEEE INFOCOM '94*, pages 368–373, 1994.

[50] 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.

[51] Peter Marbach. Pricing differentiated services networks: Bursty traffic. In *Proc. IEEE INFOCOM '2001*, pages 650–658, 2001.

[52] 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.

[53] John-Francis Mergen. Personal communication.

[54] R. Nagarajan and J. Kurose. On defining, computing and guaranteeing quality-of-service in high-speed networks. In *Proc. IEEE INFOCOM '90*, pages 2016–2025, 1992.

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

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

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

[58] C. Papadimitriou and J. Tsitsiklis. The complexity of optimal queueing network control. *Mathematics of Operations Research*, 24(2):293–305, 1999.

[59] 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.

[60] 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.

[61] 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.

[62] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In *Proc. SPIE International Conference on Performance and Control of Network Systems*, pages 296–310, 1997.

[63] 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.

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

[65] Kihong Park. Warp control: a dynamically stable congestion protocol and its analysis. In *Proc. ACM SIGCOMM '93*, pages 137–147, 1993.

[66] Kihong Park. Self-organized multi-class QoS provision for ABR traffic in ATM networks. In *Proc. 15th IEEE International Phoenix Conference on Computers and Communications*, pages 446–453, 1996.

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

[68] H. Ren and K. Park. Efficient shaping of user-specified QoS using aggregate-flow control. In *Proc. IEEE/IFIP International Workshop on Quality of Future Internet Services*, pages 259–271, 2000.

[69] 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.

[70] H. Ren and K. Park. On the QoS provisioning power of optimal aggregate-flow scheduling. In *Proc. SPIE International Conference on Scalability and Traffic Control in IP Networks*, 2001.

[71] H. Ren and K. Park. On the complexity of optimal aggregate-flow scheduling. Technical report, CSD-TR 02-021, Dept. of Computer Sciences, Purdue University, 2002.

[72] H. Ren and K. Park. Performance evaluation of optimal aggregate-flow scheduling: A simulation study. *Computer Communications*, 2002.

[73] 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.

[74] J. Sairamesh, D. Ferguson, and Y. Yemini. An approach to pricing, optimal allocation and quality of service provisioning in high-speed networks. In *Proc. IEEE INFOCOM '95*, pages 1111–1119, 1995.

[75] L. Schrage. An alternative proof of a conservation law for the queue G/G/1. *Operations Research*, 18(1):185–187, 1970.

[76] J. Shanthikumar and D. Yao. Multiclass queueing systems: Polymatroidal structure and optimal scheduling control. *Operations Research*, 40(2):293–299, 1992.

[77] Scott Shenker. Making greed work in networks: a game-theoretic analysis of switch service disciplines. In *Proc. ACM SIGCOMM '94*, pages 47–57, 1994.

[78] Cisco Systems. Cisco ios software. http://www.cisco.com/en/US/products/sw/iosswrel/index.html.

[79] Cisco Systems. How to choose the best router switching path for your network. http://www.cisco.com/warp/public/105/20.html.

[80] Cisco Systems. Qc: Cisco ios quality of service solutions configuration guide, release 12.2. http://www.cisco.com/en/US/products/sw/iosswrel/index.html.

[81] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta. Spawn: a distributed computational economy. *IEEE Trans. Software Engineering*, 18(2):103–117, 1992.

[82] S. Webster and K. Baker. Scheduling groups of jobs on a single machine. *Operations Research*, 43(4):692–703, 1995.

[83] Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[84] W. Whitt. A review of $L = \lambda W$ and extensions. *Queueing Systems*, 9(3):235–268, 1991.

[85] 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.

[86] Z. Zhang, D. Towsley, and J. Kurose. Statistical analysis of the generalized processor sharing scheduling discipline. *IEEE J. Select. Areas Commun.*, 13(6):1071–1080, 1995.

VITA

Huan Ren received his B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong Univerisity, People's Republic of China. Since 1998 he has been working toward a Ph.D. degree in the Department of Computer Sciences at Purdue University. His research interest include scalable quality of service provisioning, network modeling, network protocols and algorithms design, and scheduling.