Price-coupled Scheduling for Differentiated Services: G$c\mu$ versus GPS

Jan A. Van Mieghem[*] Piet Van Mieghem[†]

Januari, 2002

Abstract

We present an integrated approach to pricing and scheduling for services that are differentiated in terms of throughput, delay and loss specifications. The key building block to the model are quality value curves that specify a user's value of higher quality levels. From the analysis emerges a pricing rule that charges based on rate and quality grade, and a dynamic scheduling rule, called the G$c\mu$ rule. The analysis also derives the economically-optimal probabilistic QoS guarantee parameters.

We compare our model to the deterministic approach of QoS guarantees using burstiness constraints and fair scheduling rules. The scheduling that arises from such deterministic approach is the well-known Generalized Processor Sharing (GPS). A comparative analysis inspires the fair G$c\mu$-PS rule as the scheduling rule that combines the unique strengths of GPS and G$c\mu$. This G$c\mu$-PS rule is proposed as a tailored scheduling solution for *both* the Expedited Forwarding class and the four Assured Forwarding classes in the IETF's Differentiated Services.

1 Introduction

Overview A few years ago, the Internet community standardized "Integrated Services" [40, 33] as a modified ATM implementation of the notion of QoS guarantee. In ATM, the network attempts to satisfy specific lower levels for given service metrics, typically called *service guarantees*. Recognizing the stochastic nature of the network flows and its resulting performance, such service guarantees are expressed as probabilistic constraints. For example, service guarantees that cell transfer delay d and cell loss ratio clr will not exceed an upper bound D and ε_l, respectively, may be expressed as:

$$\Pr[d > D] \le \varepsilon_d \qquad \text{and} \qquad clr \le \varepsilon_l, \tag{1}$$

where ε_d is a (typically small) probability. Since guaranteeing QoS for each flow assumes that state information for each flow is maintained, the lack of scalability in offering QoS for the huge amount of Internet flows has lead to "Differentiated Services," where, apart from expedited forwarding, some *relative* QoS for aggregate flows is introduced [1, 3]. Aside from this scalability problem, there are other difficulties with such "guaranteed service approach." First, it is difficult to specify QoS parameters like D, ε_d and ε_l before services are well developed [11]. Second, any guarantee is only legally meaningful when accompanied by a penalty agreement, which would specify the contingency penalty to be paid by the service provider if the guarantees are not met (similar to a "warranty" package for other goods). Third and related, this approach assumes that user applications require rather hard constraints (if the user sets the parameters D and ε_d); or that a cost-benefit analysis has been carried out in-advance to specify economically the best guarantee parameters by weighing them to the penalties (if the network

[*]Kellogg Graduate School of Management, Northwestern University, 2001 Sheridan Road, Evanston, IL 60208-2009,USA. Email: VanMieghem@kellogg.nwu.edu

[†]Information Technology and Systems, Delft University of Technology P.O Box 5031, 2600 GA Delft, The Netherlands. Email: P.VanMieghem@its.tudelft.nl

sets the parameters); or a combination of these two (if the parameters result from a negotiations between users and network). In summary, meaningful use of a *guaranteed* service approach—if it is to be used at all—requires a larger encompassing model to derive the QoS guarantee parameters.

Therefore, we present a more general approach to incorporating QoS considerations that does not assume exogenously determined QoS guarantees. Rather, anticipated QoS levels are coupled to the willingness of users to pay for such level of QoS. This reflects our position that QoS, like most other product or service attributes in life, is a *relative* concept whose consumption level depends on the price and the user's needs and financial willingness to pay for that QoS level. Rather than seeking QoS guarantees by imposing scheduling control bounds to the network (without considering financial implications), our model lets self-interested users and the network choose an economically-optimal level of QoS by trading off money for QoS. Our model thus replaces the hard constraints from QoS guarantees by a QoS-dependent payment structure. Pricing must be related to valuation, and therefore, we introduce *service value curves* as our model building blocks that specify how valuable a service level is for each application. Recognizing the continuous nature of most applications, these service value curves are typically continuous in contrast to the hard QoS guarantees. For example, the service value curve $C_{i,a}(d)$, or delay cost curve in this case, would describe the monetary cost that user i incurs when his application a message is delayed by d time units. By modeling the *value* of *quality* of service in addition to the usual value or utility of quantity (rate) of service, it becomes meaningful and feasible to address the following questions: what are the network optimal quality of service levels? How should these service levels be priced to induce an individually acting user to choose these network optimal service levels? How should the network be scheduled to provide these optimal service levels? In this paper we will provide an answer to each of these three questions. During our analysis we will derive the optimal service specifications that a network should offer. In other words, our economic-queuing model can be used to determine the optimal set of service guarantee parameters (such as the D and ε_d).

The scheduling rule that emerges from this analysis is a dynamic extension of the well-known static priority $c\mu$ rule that was called appropriately the Generalized $c\mu$ rule (G$c\mu$) in [36]. To appreciate the features of the G$c\mu$ rule, we will relate it to the Generalized Processor Sharing (GPS) [27], which is a basic scheduling rule for a large class of rate-based schedulers for packet networks (such as Internet and ATM). While the G$c\mu$ rule derives from a stochastic analysis, the GPS rule is most easily derived from a deterministic analysis. We briefly review the deterministic approach to guaranteeing quality of service (QoS) and also relate price-setting in the stochastic system to price-setting in the deterministic approximation. From this comparative analysis we propose G$c\mu$-PS, a scheduling rule that combines the unique strengths of GPS and G$c\mu$.

Objective The prime objective of this paper is to present a framework that integrates technology and economics to derive mutually consistent pricing and scheduling decisions for differentiated services. While many papers present pricing results or propose scheduling rules, our integrated framework appears novel in that *both* scheduling and pricing emerge endogenously and are tailored to the specific service value curves. The focus of this paper, therefore, is on modeling rather than algorithm design or performance simulation. Our second objective is to compare the optimal G$c\mu$ scheduling rule to the well-known GPS, which derives its success due to inherent robustness and fairness. We propose G$c\mu$-PS, a scheduling rule that combines the unique strengths of GPS (inherent robustness and fairness) with those of G$c\mu$ (optimality, incentive-compatible pricing, and flexible quality sensitivity modeling).

Outline We start in section 2 with the general model. Section 3 presents system optimal service levels and discusses how to price these levels in a decentralized system. Section 4 presents the G$c\mu$ scheduling rule. Section 5 briefly revisits the deterministic approach to guaranteeing QoS and relates it to the GPS scheduling rule. Section 6 compares G$c\mu$ to GPS and discusses implications for implementation of scheduling and pricing. Section 7 concludes. A short note on our notation: we will not distinguish between scalars and vectors. The number of subscripts should make that clear. For example: if λ_i is the vector rate of customer i, the $\lambda_{i,s}$ will denote his scalar rate to grade s.

Literature review Pricing of queuing systems has a long history in the economics and operations research community. This history is summarized in [37]. Over the last decade, network researchers from the electrical engineering and computer science community have adopted pricing mechanisms and applied them to settings where technology is more richly modeled, as reviewed in [7]. Kelly and co-authors describe charging and rate control of elastic traffic [11, 16]. Kelly's ideas have recently inspired several other articles (see e.g. [29, 19]). As in our model, congestion feedback gives customers information on how to set their rates to maximize their quantity-based utility (minus price). Our approach differs in that we also model quality value, in addition to quantity-based utility, and present an integrated framework to set prices and scheduling that are not only rate-, but also quality-dependent. A broader overview of scheduling is found in Keshav's book [17, chap. 9] and in the recent review by Guérin and Peris [13]. Recently, Sariowan *et al.* [32] have proposed flexible service specifications via a "service curve" to guarantee a spectrum of QoS rather than a single measure (such as minimum bandwidth or maximum delay). Stoica *et al.* [35] have added two dimensions, hierarchy and fairness, to Cruz's "service curve" concept. *The book of Le Boudec and Thiran [20]contains the current state of the deterministic theory of queueing that integrates arrival and service curves using min-plus algebra coherently.*

While these service curves and our service *value* curves both allow for a flexible QoS model, both differ substantially in approach and intent. Sariowan *et al.*'s service curve specifies the minimal service effort that a server should allocate dynamically to an input flow (i.e., it is a lower bound on the scheduling allocation control process). As such, it allows for a continuous generalization of single-numbered user-application-specific hard bounds on delay and backlog, while the goal remains to find a scheduling rule that guarantees that service curve. In contrast, our service value curve denotes the monetary value of a given service level to a user, similar to a typical rate-based utility curve in economics that denotes the value of a given rate. It allows for an integrated model to set *and price* quality levels in an economic manner and to derive a supporting scheduling rule. Finally, the present framework can be applied to the IETF Differentiated Services. Recently, Ren and Park [31] have proposed a different model for Diffserv that is also game-theoretic yet specifically tuned for Diffserv aggregation and implementation. This short literature review is supplemented by additional literature referenced in the text below at appropriate places.

2 Modeling Differentiated Quality of Service

A basic element of a differentiated QoS formulation is a model of the heterogeneity of customer behavior and of service offerings. For this purpose, each user is classified as one of several *types*, indicated by $i \in I$. Similarly, each service offering by the network is classified as one of several QoS grades, indicated by $s \in S$.

QoS Modeling Multimedia communication consists of information transfer from different applications (e.g. voice, video and data files) that each may require several different dimensions of "service quality" QoS, such as delay, jitter, packet loss, etc. In this section, *we confine ourselves to one QoS measures*: *delay* (or waiting time before a customer receives service). In section 5 we will also consider packet loss. QoS considerations are incorporated through *quality value curves*, which couple experienced QoS levels to the willingness of users to pay for such level of QoS. QoS value curves specify the degradation in value that a user experiences due to a lower QoS level. The QoS metric, delay, is thus modeled by a *delay cost* function $C_i(d)$ that specifies the degradation in value that a user of type i experiences when his job experiences a delay d. We assume that the quality value functions $C_i(\cdot)$ are convex (and thus continuous and differentiable almost everywhere) increasing, reflecting the fact that often more waiting is increasingly more costly. By valuing QoS, the delay cost functions C_i will allow us to price different levels of delay.

A prominent example in Internet today is the deployment of Voice over IP (VoIP) where stringent end-to-end delays applies. For example, a tolerable one-way mouth-to-ear delay for a voice communication [38] is about 150 ms, while the packetization delay depending on the codec varies from 20 ms to 80 ms, leaving a remaining network end-to-end delay budget ranging from 70 ms to 130 ms. Our recent measurements (accurate to within $10\mu s$) on the Internet [15], [5] clearly indicate that the current best effort Internet architecture cannot provide toll-quality telephony since for a large amount of paths, the end-to-end delay broadly exceeds 100 ms. These observations imply that scheduling is necessary for VoIP. Possible delay cost functions for VoIP (curve C_1) and for non-real-time services (e.g. email) are shown in Figure 1.

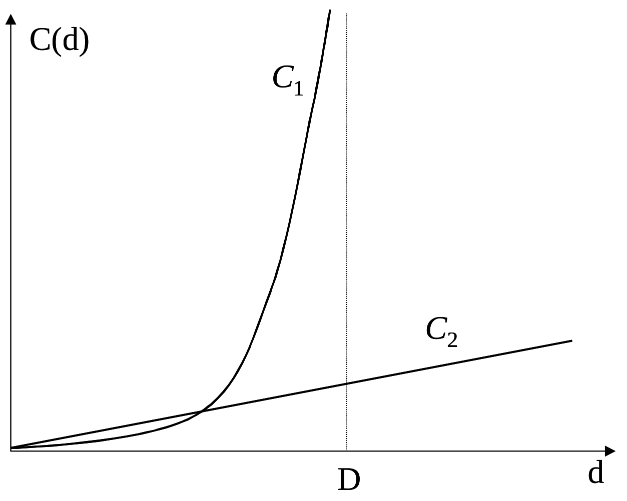

Figure 1: Example of delay cost functions for real-time services (such as VoIP where D is about 100 ms) and non-real-time services.

User Decision Model Market segment or type i consists of many small users who each decide individually whether or not to use the network. Type i customers share the same delay cost function C_i, which may reflect the sensitivity of a given application (text, video, voice...), but are heterogeneous in that they derive different value from a given rate of processing. These settings well corresponds with the IETF Diffserv approach where, besides service differentiation, aggregation of flows is key to provide a scalable network solution.

Similar to traditional demand curves in economics, type i customers can be ordered in decreasing value and it is convenient to endow them with a "label": the customer with label x_i receives value $v_i(x_i)\Delta x_i$ from service at rate Δx_i. Each user may choose a service grade s at a price p_s per unit rate. 'Higher' QoS grade may provide more timely service, thereby reducing delay costs, but at a higher price. When making his decision as to whether to send rate $\Delta x_{i,s}$ to grade s, each user seeks to maximize his expected profit, which equals value $v_i(x_i)\Delta x_{i,s}$ minus cost, which has two components: the charge $p_s\Delta x_{i,s}$ and his expected delay cost. The user learns about his anticipated delay experience d_s when submitting to grade s through either past experience or network feedback signals. Thus, when sending a rate $\Delta x_{i,s}$ to grade s, the user anticipates a delay cost rate $\Delta x_{i,s}\mathrm{E}^r_\Lambda C_i(d_s)$, where E^r_Λ denotes the expectation operator under equilibrium conditions when the network load is Λ and scheduling rule is r (to be defined in next paragraph). The expected profit rate of the user with label x_i when sending rate $\Delta x_{i,s}$ to grade s is denoted by $b_{i,s}(x_i)\Delta x_{i,s}$, where:

$$b_{i,s}(x_i) = v_i(x_i) - \mathrm{E}^{r^*}_\Lambda C_i(d_s) - p_s. \tag{2}$$

Let $\lambda_{i,s}$ denote the label of the "marginal type i customer" who is indifferent between sending to grade s or not: $b_{i,s}(\lambda_{i,s}) = 0$. By construction $v_i(x_i)$ decreases in x_i, so that all customers with labels

$x_i < \lambda_{i,s}$ have $b_{i,s}(x_i) \geq 0$ and submit to grade s. Therefore, the aggregate rate from type i users choosing grade s is $\sum_{x_i<\lambda_{i,s}} \Delta x_{i,s}$. Adopting standard assumptions in economics as Mendelson and Whang [24], Lederer and Li [21], and many others, we assume that customers are atomistic and their rates infinitesimal. The summation $\sum_{x_i<\lambda_{i,s}} \Delta x_{i,s}$ is thus approximated by the integral $\int_0^{\lambda_{i,s}} dx_{i,s}$, so that $\lambda_{i,s}$ represents the aggregate rate from type i users choosing grade s, where $b_{i,s}(\lambda_{i,s}) = 0$. The latter condition represents the familiar "demand curve" for grade i service from market segment i: when charged a price p_s, the aggregate demand rate will be $\lambda_{i,s}$, where $\lambda_{i,s}$ is on the demand curve: $b_{i,s}(\lambda_{i,s}) = 0$. Similarly, the aggregate value rate and delay cost rate to all type i users choosing grade s are denoted by $V_i(\lambda_{i,s}) = \int_0^{\lambda_{i,s}} v_i(x_i)dx_{i,s}$ and $DC^r_{\Lambda,i,s} = \lambda_{i,s}\mathrm{E}^r_\Lambda C_i(t_s)$.

Network Model: The service provider designs its service offering through two strategic control levers: pricing and scheduling. The service provider can partially control external subscription rates by setting the *price* vector $p = (p_s)_{s\in S}$, defined earlier. Indeed, this more tailored control is exactly the motive behind offering differentiated services. In addition, the service provider has *dynamic internal* control in that it can decide at each point in time how to serve customers through the network. The internal scheduling rule used to manage the network is denoted by r. We consider dynamic scheduling rules that may depend on the current internal state of the queuing process and the total rate vector Λ, where $\Lambda_s = \sum_{i\in I} \lambda_{i,s}$, which is the aggregate rate into each grade. Then, regardless of the queuing system's complexity, QoS metrics can be summarized by a *technology function*, which specifies how the control rule r transforms a total rate vector Λ into QoS. With our focus on delay as the QoS metric, this technology function would represent the delay time distribution functions $F^r_s(\cdot\,|\Lambda)$ that a user would experience when submitting to grade s while the network uses scheduling rule r and the total load is Λ:

$$F^r_s(x\,|\Lambda) = \Pr[d_s \leq x|r, \Lambda], \qquad \forall x \geq 0. \tag{3}$$

Thus, user i's expected delay cost when submitting one job or packet to grade s is:

$$\mathrm{E}^r_\Lambda\left[C_i(d_s)\right] = \int C_i(t)dF^r_s(t\,|\Lambda), \tag{4}$$

where E^r_Λ represents the expectation operator when rule r is used and the server total arrival vector is Λ (and the distinction between delay d_s and differential operator d should be clear). In equilibrium, using price rate schedules and internal control rule r the service provider will earn profit rate π_S :

$$\pi_S = \sum_{s\in S} p_s\Lambda_s = \sum_{s\in S} p_s \sum_{i\in I} \lambda_{i,s}. \tag{5}$$

The service provider sets prices and scheduling to maximize system profits Π, where

$$\Pi = \sum_{i,s} \left(V_i(\lambda_{i,s}) - DC^r_{\Lambda,i,s}\right). \tag{6}$$

Notice that (4) shows that customer i's delay cost of grade s depends on the total subscription vector Λ. Reversing the argument, by sending a job to grade s, customer i may impact the delay cost of other customers. As we will see later, these *externality* effects will impact the pricing decisions. Finally, we define an *equilibrium* of this non-cooperative decision model as any triplet (P, λ, r) of price schedules, customer subscription vectors and control rule that satisfies the users' and the network's decision problems, and that is such that no agent has an incentive to unilaterally deviate from the equilibrium triplet (which thus is a Nash equilibrium).

Determining QoS guarantees As discussed in the introduction, a meaningful use of QoS guarantees requires a broader encompassing model to specify how the (delay) QoS guarantee parameters D_s and ε_s satisfying

$$\Pr[d_s > D_s] \leq \varepsilon_s. \tag{7}$$

are determined. The model presented here can be used for that purpose. Indeed, once the network has determined the optimal load vector Λ and scheduling rule r (which depend on the users' value and quality value curves), the delay distribution function (3) can be determined. This then specifies a family of *equivalent* QoS guarantee parameters D_s and ε_s, because they should satisfy:

$$\Pr[d_s > D_s] = 1 - F_s^r(D_s\,|\Lambda) \leq \varepsilon_s. \tag{8}$$

Summarizing, the model presented here incorporates QoS considerations through quality value curves. Pricing and scheduling become the solution of optimization problems that incorporate users value and quality sensitivity. Using the optimal scheduling rule, a family of equivalent QoS guarantee parameters are specified in terms of the optimal delay distributions. Hence, the model gives a consistent approach to determine a family of optimal guarantee parameters. Any couple (D_s, ε_s) satisfying (8) are equivalent because users are indifferent between them (by construction). Interpreted differently, (8) specifies the optimal trade-off between upper bound D_s and stringency ε_s that the network should provide.

3 Determining Rates and Pricing

Centralized Solutions A first step in analyzing the multi-person decision model presented above, is to analyze the relaxed centralized model where one omnipotent central manager makes all decisions and knows all the model data. Because the central manager directly controls all rates $\lambda_{i,s}$, indirect control through offering service grades and pricing is superfluous. The only decisions that are left for the central manager is to determine the optimal rates $\lambda^*_{i,s}$ and the optimal scheduling rule r^*. In the centralized system: the central planner aggregates all users of type i directly to queuing class i, which is equivalent to grade $s = i$, as that offers the finest information set on which can be scheduled. (Such allocation clearly dominates any "mixing" of user flows of different types into one class, as the finer allocation can always replicate the coarser.) Thus, optimal scheduling will route the aggregate flow of users of type i into a dedicated queueing class (buffer). While service grades are superfluous in the centralized model, they will coincide with queueing classes in the decentralized system and we will use both terms "service grade" and "queuing class" interchangeably[1]. Therefore, henceforth, we will have a one-to-one correspondence between aggregate flows of a type i and the service grade s and queuing class k designed for that type.

The central-optimal scheduling rule r^* is a rule that minimizes total delay costs

$$DC_\Lambda^r = \sum_{i\in I, s\in S} \lambda_{i,s} \mathrm{E}_\Lambda^r\left[C_i(d_s)\right], \tag{9}$$

where we assume for now that r^* is known (more on that later in the next section). We know that $\lambda_{i,s} = 0$ if $i \neq s$; thus, let $\Lambda^* = (\lambda^*_{i,i})_{i\in I}$ denote the optimal aggregate rate vector. The the central-optimal rate vector Λ^* satisfies the first-order conditions of (6):

$$v_i(\Lambda_i^*) \leq \mathrm{E}_{\Lambda^*}^{r^*}\left[C_i(d_i)\right] + \sum_{j\in I} \Lambda_j^* \frac{\partial}{\partial \lambda_{i,i}} \mathrm{E}_\Lambda^r\left[C_j(d_j)\right], \tag{10}$$

where equality holds if and only if $\Lambda_i^* > 0$. The necessary first-order conditions (10) have a familiar economic interpretation: the optimal rate Λ_i^* equates the value of marginal type i user with its total marginal cost. The latter is born by different parties: the "self-regulating" term $\mathrm{E}_{\Lambda^*}^{r^*} C_i(d_i)$ accrues to

[1]This will lead to no confusion in our model. In general, the number of queuing classes is less or equal to the number of service grades. In optimal scheduling, one keeps service differentiated and thus has grades coincide with classes. (Notice, however, that a suboptimal scheduling rule could, for example, serve all service grades indiscrimenantly and put all grades into one FIFO buffer.)

the marginal type i user himself, but the "externality term" $\sum_{j\in I}\Lambda_j^*\frac{\partial}{\partial\lambda_{ii}}\mathrm{E}_\Lambda^r\left[C_j(d_j)\right]$ is inflicted onto all the other users. Finally, a zero rate $\Lambda_i^*=0$ obtains if its marginal value does not outweigh its marginal cost.

Decentralized System: In the true decentralized system, there is no central omnipotent single planner and all users make their own decisions. As explained earlier, users of type i make their own individual decisions, resulting in an aggregate rate $\lambda_{i,s}$ to grade s, where $\lambda_{i,s}$ satisfies $b_{i,s}(\lambda_{i,s})=0$. (In general, any user could choose any grade. Below, however, we will see that the pricing scheme will induce all users of type i to submit only to grade i.) Comparing these individual optimality equations (2) with the centralized optimality equations (10), we directly see that the optimal price p_i^* for grade i must satisfy:

$$p_i^*=\sum_{j\in I}\Lambda_j^*\frac{\partial}{\partial\lambda_{i,i}}\mathrm{E}_\Lambda^r\left[C_j(d_j)\right], \tag{11}$$

which is exactly the externality effect. This directly explains the optimal pricing approach: To induce users to incorporate the externality effect they should be charged for the externality cost they inflict onto others.

There is, however, a second set of conditions that the prices must satisfy for the rates $\lambda_{i,i}^*=\Lambda_i^*$ to be optimal in the decentralized system. These conditions, called incentive-compatibility conditions, stipulate that a type i user should find it in his best interest to submit to only grade $s=i$. That is, the expected benefit for a type i user with label x_i of choosing grade $j\neq i$ must be less than when choosing grade i: $b_{i,j\neq i}(x_i)<b_{i,i}(x_i)$. These incentive-compatibility conditions guarantee that type i users will all choose only grade i, while the choice (11) of prices and the use of the scheduling rule r^* guarantee that total type i rates in the decentralized system equal the rates $\lambda_{i,i}^*=\Lambda_i^*$. We can show that this is indeed the case for single-server systems (notice that the previous analysis is also valid for networks), which extends the results of Lederer and Li [21] and Mendelson and Whang [24] to arbitrary delay cost functions:

Proposition 1 *The rates Λ^*, prices p^* and scheduling rule r^* are optimal and incentive-compatible if service time distributions are homogeneous (i.e., type-independent).*

Proof: Suppose the mechanism were not incentive-compatible. Then, there would exist an atomistic type i customer with label $x_i\leq\lambda_i$, that has incentive to submit to a grade $j\neq i$ so that $b_{i,j}(x_i)>b_{i,i}(x_i)$. Since the term $v_i(x_i)$ in (2) cancels, substituting (11) into (2) yields:

$$\mathrm{E}_\Lambda^{r^*}C_i(d_j)+\sum_{j'\in I}\Lambda_{j'}^*\frac{\partial}{\partial\lambda_{j,j}}\mathrm{E}_{\Lambda^*}^{r^*}C_{j'}(d_{j'})<\mathrm{E}_\Lambda^{r^*}C_i(d_i)+\sum_{i'\in I}\Lambda_{i'}^*\frac{\partial}{\partial\lambda_{i,i}}\mathrm{E}_{\Lambda^*}^{r^*}C_{i'}(d_{i'}). \tag{12}$$

With equal service time distributions, type i jobs and type j jobs are indistinguishable from a scheduling perspective. Thus, adding an infinitesimal rate $d\nu$ of either type i jobs or type j jobs to grade (=class) j impacts delay distributions, and thus expected delay costs, identically: $\frac{\partial}{\partial\lambda_{i,j}}\mathrm{E}_\Lambda^{r^*}C_{j'}(d_{j'})d\nu=\frac{\partial}{\partial\lambda_{j,j}}\mathrm{E}_\Lambda^{r^*}C_{j'}(d_{j'})d\nu$. Hence, (12) simplifies to $\frac{\partial}{\partial\lambda_{ij}}DC_\Lambda^{r^*}-\frac{\partial}{\partial\lambda_{ii}}DC_\Lambda^{r^*}<0$, which means that total delay costs $DC_\Lambda^{r^*}$ would decrease by reallocating an infinitesimal rate $d\nu$ of type i from queue class i to queue class $j\neq i$. This would contradict the optimality of r^* for any flow vector Λ. Hence, there exist no x,i or $j\neq i$ that satisfies $b_{i,j}(x)>b_{i,i}(x)$. Thus, the mechanism is incentive compatible and, by construction of the prices p^* so that $b_{ii}(\Lambda_i^*)=0$, the mechanism is also optimal.■

Given that the service time of a packet is typically independent[2] of its type, the proposition solves the problem of pricing and scheduling. To implement this solution, however, we must know the optimal rates Λ^* and the optimal scheduling rule r^*. The determination of Λ^* (needed for the prices p^*) requires knowledge of each type's value function, which is rather unlikely. However, the price

[2]Pricing for heterogeneous service times is discussed in [37] for a model of flow-by-flow scheduling, which is equivalent to a type consisting of a single, non-atomistic customer.

expressions can also be evaluated at the *actual* rates $\Lambda(t)$ observed at time t, instead of at Λ^*. That leads to dynamic pricing, obviating the need of knowledge of value functions. (Under mild regularity conditions, dynamic pricing leads to Λ^* in equilibrium as shown by Masuda and Whang [23].) Thus, only marginal quality value curves and the scheduling rule r^* are necessary knowledge to set prices. The next section 4 presents the G$c\mu$ rule as a proxy for the exact optimal scheduling rule r^*; using the G$c\mu$ rule, all prices and the optimal service levels can be implemented.

In [37], we consider a flow-per-flow model where each type corresponds to one user. In that case, a "two-part tariff" that charges a fixed subscription fee in addition to a variable price is shown optimal. The fixed subscription fee for type i equals the expected profit that "the" type i user makes. In this paper, we consider an aggregate flow model, where each type consists of many atomistic users that each send an infinitesimal rate. Therefore, each atomistic user makes infinitesimal profits so that in an aggregate flow model the fixed subscription fee is infinitesimal and a variable price only is optimal. While the aggregate model is in a sense simpler and more realistic, the framework presented here is directly applicable to flow-per-flow models as shown in [37]. The scheduling rule that emerges is the Generalized $c\mu$ rule (explained in the next section) in both models, the only difference being that a queuing class holds an aggregate flow[3] here instead of a unique user flow in [37].

4 Determining Scheduling and Service Levels: The G$c\mu$ Rule

The previous section 3 highlighted the role of the optimal scheduling rule r^*: it minimizes total expected delay costs, denoted by $DC_\Lambda^{r^*}$ for any given load vector Λ and stores aggregate flow of user type i in queue class i. In addition, we need its delay distributions F_Λ^r to quantify the expected delay cost via (4) and its gradients to compute the optimal prices p^*.

Scheduling control $\mu_i(t)$ A scheduling rule specifies how the service capacity is allocated to the different jobs in the queues at any point in time t. With convex-increasing delay costs, it is optimal to serve each class in FIFO order [36, Prop. 1]. Hence, to define a scheduling rule it only remains to specify how capacity is allocated to queues. We will denote the actual service rate allocated to class i at time t by $\mu_i(t)$. As usual, we will denote the maximal service rate of the server by μ, which yields the capacity constraint on the scheduling control:

$$0 \leq \sum_{i \in I} \mu_i(t) \leq \mu. \tag{13}$$

We will restrict attention to work-conserving scheduling policies, which are policies where the processor serves at full capacity whenever any jobs are in the system. Hence, denoting by $N_i(t)$ the number of class i jobs in the system at time t, we have

$$\sum_i N_i(t) > 0 \Longleftrightarrow \sum_i \mu_i(t) = \mu. \tag{14}$$

Also, long-term stability requires that long run utilization (or traffic intensity)

$$\rho = \frac{\sum_i \Lambda_i}{\mu} = \sum_i \rho_i \leq 1, \tag{15}$$

where the class i utilization is $\rho_i = \frac{\Lambda_i}{\mu}$. Notice that long-term stability is guaranteed through the use of optimal prices p^*: because at least one type's delay distribution in a queueing system grows unbounded as $\rho \to 1$, the externality term and thus the prices would also grow unbounded as $\rho \to 1$, preventing users to load the system to $\rho = 1$.

[3]The G$c\mu$ rule described below even applies when multiple aggregate flows are routed into a queueing class. However, we don not consider that situation here because it is suboptimal.

If the delay sensitivity functions C_i are linear, then it is well-known [18, pp. 125] that r^* is a static-priority rule that gives static priority to classes i in the order of their index $c_i\mu$, where $c_i = \frac{d}{dt}C_i(t)$ and is called the marginal delay cost. For many applications, however, the delay sensitivity is non-linear (more delay is increasingly worse), in which case the exact optimal scheduling rule is unknown. In [36], however, we show that the optimization problem is asymptotically tractable for a single server queuing system. To state that result, we first need some additional notation. In [36], it is shown that the scheduling optimization problem is asymptotically tractable in "heavy traffic" (i.e., if traffic intensity approaches capacity $\rho \to 1$) and present the Generalized cμ (Gcμ) rule, which is shown to be asymptotically optimal as $\rho \to 1$. Here we show how to specify the Gcμ rule and its delay distributions in our setting in concrete, executable terms and we provide some intuition behind the rule. To stress the approximate mode of analysis, we will use $\simeq$ to denote approximate relationships that are asymptotically exact if $\rho \to 1$.

Intuition The Gcμ rule is founded on three facts from queuing theory that are best described in terms of the total workload process W, which measures the total amount of work (in time units for the server) that is waiting: $\mathrm{E}[W(t)] = \sum_i N_i(t)/\mu$. First, the total workload is invariant for work-conserving scheduling rules. Indeed, new arrivals contribute to an increase in $\mathrm{E}[W(t)]$ at (average) rate $\sum_i \Lambda_i/\mu = \rho$, while serving drains $\mathrm{E}[W(t)]$ at rate 1 by definition[4] regardless of which class is served. This yields a net decrease of $\mathrm{E}[W]$ at rate $\rho - 1$, as long as there is work present. The scheduling rule, however, does impact how this total workload W is distributed over the different classes: when serving class i, its average *class*-workload $\mathrm{E}[W_i]$ decreases at rate $\rho_i - 1$, while arrivals increase the average workload $\mathrm{E}[W_j]$ of any other class j at rate ρ_j. The two other facts follow from heavy traffic approximations: class workloads 'live on a faster time scale' than the total workload process. Indeed, as $\rho \to 1$, the total workload hardly changes, while class workloads keep changing at a finite rate. At the time-scale of the total workload, it is as if one can almost instantaneously shift workload away from one class to the other classes by serving that class for an infinitesimal amount of time while the total workload is unchanged. Third, in well-behaved heavy traffic limit systems, the class-workload process "converges": W_i/W approaches a W-dependent constant. In effect, scheduling switches among classes precisely so as to keep the class workload vector $(W_1, W_2, ..., W_I)$ close to the point $g(W)$ on a switching curve $g(\cdot)$, which we will define shortly. Aside from fast, but small disturbances around the switching curve, the W_i thus follow the W movement and change very slowly. Meanwhile many class i jobs flow through the system at rate Λ_i while the class workload vector remains relatively constant and close to $g(W)$. "It is as if a job takes a *snapshot* of the network when it enters and all queues remain at that same value during the job's sojourn throughout the network. (Reimann [30, p. 413])". In [36], we apply Little's law to yield the state-dependent delay:

$$d_i \overset{\text{distribution}}{\simeq} \frac{N_i}{\Lambda_i} \overset{\text{distribution}}{\simeq} \frac{W_i}{\rho_i}. \tag{16}$$

Thus, the instantaneous delay cost rate can be expressed in terms of class workloads as $\sum_i \Lambda_i C_i \left(\frac{W_i(t)}{\rho_i}\right)$. The intuition behind the Gcμ rule then is to "distribute" the total workload $W(t) = \sum_i W_i(t)$ over the different classes such that this delay cost rate is minimized at each point in time. This greedy cost-minimizing allocation of total workload W to class workloads is found by solving the following linearly-constrained convex optimization problem parametrically for $W \geq 0$:

$$g(W) = \arg \min_{\{W_i \geq 0: i \in I\}} \sum_{i \in I} \Lambda_i C_i \left(\frac{W_i}{\rho_i}\right) \tag{17}$$

$$\text{subject to : } \sum_{i \in I} W_i = W. \tag{18}$$

[4]By definition, W is the amount of work in time units for the server. Thus, under a work-conserving policy, when the server is "on" it serves one unit of work per unit of time.

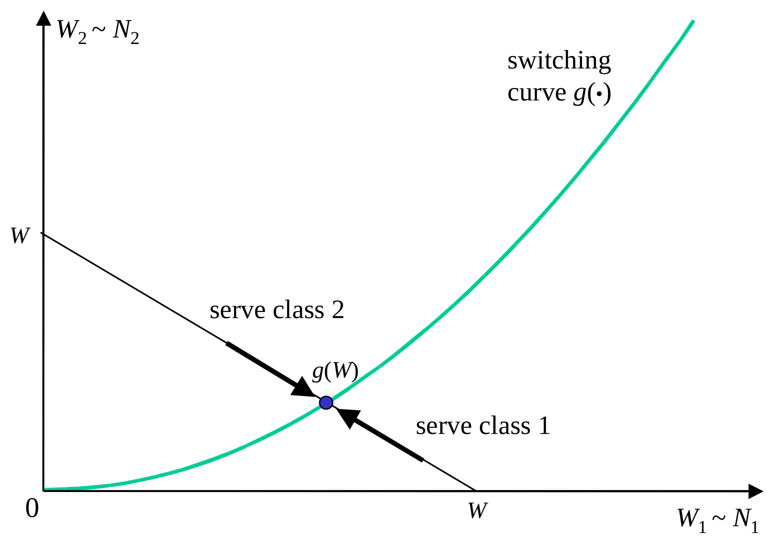

Figure 2: The switching curve $g(\cdot)$ defines the dynamic G$c\mu$ scheduling rule.

This optimization problem defines a mapping $g(\cdot)$ from $\mathbb{R}_+$ to "class-space" $\mathbb{R}_+^I$ that represents the lowest-cost workload configuration as $W_i = g_i(W)$. As a positive sum of convex functions, $\sum_i \Lambda_i C_i\left(\frac{W_i}{\rho_i}\right)$ is convex so that the sufficient first order conditions of the optimization problem are: Let $\zeta(W)$ denote unique optimal Lagrange multiplier of (18). Then: any positive W_i and W_l solve the sufficient first order conditions:

$$c_i\left(\frac{W_i}{\rho_i}\right) = c_l\left(\frac{W_l}{\rho_l}\right) = \zeta(W), \tag{19}$$

and $W_{i'} = 0$ if $c_{i'}(0) > \zeta(W)$. As we will illustrate by an example, the solution to (19) defines the mapping g, which can be interpreted as the *switching curve* of the G$c\mu$ rule parameterized by scalar W.

`Example`: Assume a two-class queuing system where $C_1(t) = \frac{1}{3}t^3$ and $C_2(t) = \frac{1}{2}t^2$. The optimal workload allocation for a given W is given by (19):

$$\left(\frac{W_1}{\rho_1}\right)^2 = \frac{W_2}{\rho_2},$$

which yields the quadratic switching curve shown in Figure 2.□

G$c\mu$ Index and Implementations In the heavy-traffic limit (and under appropriate scaling), it is possible to control the system such that the class workload vector that corresponds to a total workload W always lies on the optimal point $g(W)$ on the switching curve. In moderate traffic, however, discrete job arrivals will make the workload vector fluctuate around the switching curve. For a given total workload $W(t)$ at time t, the G$c\mu$ rule therefore will schedule so as to keep the workload vector as close as possible to its optimal point $g(W(t))$ on the switching curve. Recognizing that $W_i/\rho_i \simeq N_i/\Lambda_i$, define the dynamic G$c\mu$ index for each class i:

$$I_i(t) = c_i\left(\frac{N_i(t)}{\Lambda_i}\right)\mu, \tag{20}$$

and denote by the class with highest index at time t by:

$$i^*(t) = \arg\max_{i\in I} I_i(t). \tag{21}$$

This directly inspires an easy implementation of the G$c\mu$ rule: at each point in time t, serve class $i^*(t)$ at full capacity. This gives the highest instantaneous reduction in marginal cost and makes G$c\mu$ a myopic rule. While reducing $I_{i^*}(t)$ the index of the other classes $i \neq i^*(t)$ will increase. Thus, by serving the highest index class $i^*(t)$, this rule attempts to satisfy the asymptotic optimality condition (19) in a stable manner.

G$c\mu$ Distributions Finally, using (16), we have that the delay distribution $F_i^{Gc\mu}(x) = \Pr[d_i \leq x] \simeq \Pr[g_i(W) \leq \rho_i x] = F_W(g_i^{-1}(\rho_i x))$, where the total workload distribution $F_W(\cdot)$ is typically known or a good approximation exists. For example, for general multiclass GI/G/1 queuing systems one can take $F_W(x|\Lambda) \simeq 1 - \rho e^{-\gamma x}$, where $\gamma^{-1} = \mu^{-1}\frac{\rho}{1-\rho}\frac{C_a^2+C_s^2}{2}$ and C_s^2 and C_a^2 denote the aggregate squared coefficient of variation of the service and interarrival times, respectively. Thus, this allows the calculation of the delay distributions that can be used to set the QoS guarantees (8).

On the optimality of G$c\mu$ The rule is exactly optimal under linear delay cost (it then coincides with the familiar $c\mu$ rule). The interesting question arises whether the *G$c\mu$*-rule is also globally optimal, i.e. for all values of the traffic intensity ρ and not just near the heavy traffic limit $\rho \to 1$, as was proved in [36]. Today, the question is still open. There is considerable evidence that control rules that are optimal in heavy traffic also perform very well in moderate traffic. The comprehensive benchmarking study in [37] shows that the approximation is excellent for moderate traffic. Moreover, the rule is *prima facie* reasonable, intuitively sound and its simplicity is appealing for implementation. In addition, Bertsimas [2] recently showed that the G$c\mu$ rule is exactly optimal for a two-class M/G/1 system with holding costs that are quadratic in the queue count vector N. This is very encouraging and suggests that the G$c\mu$ rule may be optimal for a quadratic cost criterion expressed in terms of delay d.

Nevertheless, proving exact optimality of delay costs criterions (instead of queue count) is much harder as the state-space explodes: any dynamic programming formulation must keep track of the arrival time of each job in the queue to compute delays. In the heavy traffic delay the analysis simplifies because delays are proportional to queue count via (16). The same holds for a fluid approximation; Haji and Newell [14] showed that a G$c\mu$ rule is optimal in a two-class fluid model. Hence, it is not unreasonable to conjecture that G$c\mu$ will be exactly optimal for some class of systems, but it remains a conjecture up to this point. It also is likely that there will be systems where the rule is not exactly optimal. For example, consider the case where all service time distributions have two mass points at $t = t_1$ and t_2. Once the server has been serving a job longer than t_1, the service time becomes deterministic and there may be combinatorial arguments that can improve on the G$c\mu$ rule. Thus, when special structure in the arrival or service processes reveals more information, there exists a possibility that *G$c\mu$* may be suboptimal. Since most of data-traffic (in Internet and Ethernet LAN's) is demonstrated to be long-range dependent (see for references [39]), an investigation into the performance of *G$c\mu$* for such systems may be useful.

5 Using GPS to guarantee QoS in a deterministic fluid model

To relate the G$c\mu$ rule to the well-known GPS [27], we briefly revisit the deterministic approach to QoS guarantees pioneered by Cruz [8]. Basically, this approach hinges on three assumptions:

1. each customer's arrival process satisfies a certain burstiness constraint, which puts an upper limit on the number of arrivals during any time interval,
2. the service time of each packet or bit is assumed to be deterministic,
3. and a scheduling is used that guarantees each non-empty class i a minimal service rate $\underline{\mu}_i$.

Such scheduling rule is called "fair" because no class can prevent another class from getting served. Formally:

Definition 1 *Let $B(t) = \{i : N_i(t) > 0\}$ be the set of backlogged classes at time t. A fair scheduling policy guarantees each backlogged class a minimal service rate $\underline{\mu}_i$:*

$$\begin{array}{lll} \mu_i(t) & \geq \underline{\mu}_i & \text{if } i \in B(t) \\ & = 0 & \text{otherwise.} \end{array} \tag{22}$$

Notice that any fair policy in a single server node requires processor sharing. As we will show, the result of these three assumptions is that the amount of work in the system at any point in time is bounded. Thus, also all delays and backlogs have a deterministic upper bound.

5.1 The Burstiness Constraint

Recall that Λ_i denotes the average arrival rate of class i in bits/sec. The "burstiness constraint" will impose restrictions on the stochasticity of the arrival process as follows. Let $\Lambda_i(t)$ denote the instantaneous arrival rate[5] of class i [in bits/s] at time t. Then, for $u \leq t$, $L_i(u,t) = \int_u^t \Lambda_i(\tau)\, d\tau$ represents the amount of data in bits that arrives during any time interval $[u,t]$. Let $\bar{\sigma}_i$ and $\bar{\Lambda}_i$ be positive scalars. The flow $\Lambda_i(t)$ is said [8] to satisfy the *burstiness constraint* $(\bar{\sigma}_i, \bar{\Lambda}_i)$ if, for all $u \leq t$,

$$L_i(u,t) \leq \bar{\sigma}_i + \bar{\Lambda}_i\,(t-u). \tag{23}$$

Relation (23) means that there is an upper bound to the amount of traffic contained in any interval that is equal to a constant $\bar{\sigma}_i$ plus a quantity proportional to the length of the interval. The constant of proportionality $\bar{\Lambda}_i$ can be bounded in two straightforward ways. First, since $L_i(u,t) = \int_u^t \Lambda_i(\tau)\, d\tau \leq \max_t \Lambda_i(t)(t-u)$, we have from (23) that $\bar{\Lambda}_i$ is upper bounded by the peak rate $\max_t \Lambda_i(t)$ in which case $\bar{\sigma}_i = 0$. Second, dividing (23) by $(t-u)$ and taking the limit $t \to \infty$, shows that $\bar{\Lambda}_i$ is lower bounded by the long term average rate of $\Lambda_i(t)$ which, for finite u, equals $\bar{\Lambda}_i \geq \lim_{t\to\infty} \frac{1}{t-u} \int_u^t \Lambda_i(\tau)\, d\tau$. In the latter case, where $\lim_{t\to\infty} \frac{\bar{\sigma}_i}{t-u} = 0$, a maximum (finite) value for $\bar{\sigma}_i$ is possible. In between these extremes, the rate $\bar{\Lambda}_i$ can be traded-off for the burstiness $\bar{\sigma}_i$. Given the profile $\Lambda_i(t)$ for all $u \leq t$, Naudts [25] presents efficient algorithms to extract couples $(\bar{\sigma}_i, \bar{\Lambda}_i)$. He also shows that the rate $\bar{\Lambda}_i$ versus burstiness $\bar{\sigma}_i$ curve is convex. A generalized version of (23) is $L_i(u,t) \leq f\,(t-u)$ for some nondecreasing nonnegative function $f(.)$ as used by Sariowan *et al.* [32]. The conformance of a i-job to the burstiness constraint $(\bar{\sigma}_i, \bar{\Lambda}_i)$ can be checked by a token bucket, where the tokens are generated at rate $\bar{\Lambda}_i$.

Given a burstiness constraint, the arriving work during a compact time interval is bounded. If, in addition, the service times are deterministic and we use a fair scheduling policy, the queue count is bounded. Indeed, assuming the system starts initially empty, elementary queuing theory yields that the queue count is the difference between cumulative inflow and outflow:

$$N_i(t) = \max_u\,[L_i(u,t) - M_i(u,t) : u \leq t] \tag{24}$$

where $M_i(u,t) = \int_u^t \mu_i(\tau)\, d\tau$. As we are interested in bounding the number of jobs in the queue, let us consider a busy class i period so that $N_i > 0$ for all u and t during that period. The burstiness constraints gives the upper bound (23) on L_i. At the same time, a fair scheduling policy gives a lower bound to M_i by serving class i at minimal rate $\underline{\mu}_i$ whenever $N_i(t) > 0$ so that $M_i(u,t) \geq \underline{\mu}_i(t-u)$. Hence,

$$N_i(t) \leq \max_u \left[\bar{\sigma}_i + \bar{\Lambda}_i\,(t-u) - \underline{\mu}_i\,(t-u) : u \leq t\right] \overset{\text{if } \underline{\mu}_i \geq \bar{\Lambda}_i}{\leq} \bar{\sigma}_i, \tag{25}$$

which shows that N_i is bounded provided $\underline{\mu}_i \geq \bar{\Lambda}_i$.

[5]Following [8], we adopt a continuous time fluid description.

5.2 Upper bounds on delay and packet loss

Let us investigate a single queue (or multiplexer) processing all QoS classes i, which is the simplest representation of an actual switch or router. The previous bound on N_i directly shows:

Lemma 2 *If each class has deterministic service times and satisfies a burstiness constraint $(\bar{\sigma}_i, \bar{\Lambda}_i)$ and if a fair scheduling policy is used with $\underline{\mu}_i \geq \bar{\Lambda}_i$, then a zero packet loss is guaranteed if the total buffer size $G \geq \sum_{i\in I} \bar{\sigma}_i$ and delays $d_i \leq D_i$ are guaranteed if $D_i \geq \frac{\bar{\sigma}_i}{\bar{\Lambda}_i}$.*

Indeed, given (25) we know that the sum of all queue lengths is bounded by $\sum_{i\in I} \bar{\sigma}_i$. And, with deterministic service times and a fair policy with $\underline{\mu}_i \geq \bar{\Lambda}_i$, we have that the actual delay d_i of job i is upper bounded by

$$d_i \leq \frac{\max_t(N_i(t))}{\min_t(\mu_i(t))} = \frac{\bar{\sigma}_i}{\underline{\mu}_i} \leq \frac{\bar{\sigma}_i}{\bar{\Lambda}_i}. \tag{26}$$

Hence, under the conditions of the Lemma, *both* a zero loss and an upper bound of the delay for that i-job can be expressed in function of $\bar{\sigma}_i$ and $\bar{\Lambda}_i$ only. This underlines the interest in this deterministic approach for packet networks such as the Internet and ATM; it shows that they are able to *guarantee* loss and delay constraints.

This result on guarantees must be qualified, however. First, zero packet loss is guaranteed provided the burstiness constraint is satisfied. This is typically ensured by use of a token bucket, which is a form of admission control. Thus, if a user's flow does not satisfy the burstiness constraint, some of its packets may be denied admission (and are lost), while zero packet loss is guaranteed to *admitted* packets. Second, the delay guarantees are deterministic because all stochasticity is either assumed to be bounded or be deterministic. Indeed, only under those assumptions are the probabilistic QoS guarantee constraints (7) of our general model satisfied with probability one (or $\epsilon_i = 0$). An extension of (23) to a probabilistic setting in order to incorporate ϵ_i-requirements as in (7),

$$\Pr[L_i(u,t) \leq \bar{\sigma}_i(\epsilon_i) + \bar{\Lambda}_i(\epsilon_i)\,(t-u)] \leq \epsilon_i, \tag{27}$$

where the burstiness constraint parameters become function of the stringency ϵ_i, introduces the need for large deviation theory and the concept of equivalent bandwidths. In fact, $\bar{\Lambda}_i(\epsilon_i)$ can be defined as the equivalent bandwidth of class i-job corresponding to a ϵ_i-stringency on QoS requirements. Such analysis rapidly becomes complex and even intractable. The deterministic analysis buys tractability at the expense of a more restricted and approximate model. Finally, the Lemma requires a fair scheduling rule that guarantees a minimal service rate $\underline{\mu}_i \geq \bar{\Lambda}_i$. That requirement directly relates to the capacity constraint (13):

$$\sum_i \Lambda_i \leq \sum_i \bar{\Lambda}_i \leq \sum_i \underline{\mu}_i \leq \mu.$$

In a certain sense, input shaping corresponding to the burstiness constraints $(\bar{\sigma}_i, \bar{\Lambda}_i)$ allow us to replace the difficult probabilistic constraints specified via (D_i, ε_i) with this simple, linear capacity constraint. Unfortunately, there is no simple way to express the burstiness parameters in terms of our general stochastic model. Moreover, for many (especially real-time) applications, the burstiness constraint parameters $(\bar{\sigma}_i, \bar{\Lambda}_i)$ can only be estimated roughly, as mentioned earlier. The fact that these burstiness parameters are not easily analyzed and determined somewhat undermines the attractiveness of the deterministic approach as a sound theoretical model, although the associated token bucket is widely popular. In particular, besides ATM's Usage Parameter Control (or policing function) also the IETF's Integrated Services, both the guaranteed service [33] and the controlled-load service [40] are expressed in terms of token bucket parameters $(\bar{\sigma}_i, \bar{\Lambda}_i)$.

5.3 Generalized Processor Sharing (GPS)

Fair policies not only differ in the minimal service rates $\underline{\mu}_i$, but also in how they allocate the excess capacity, denoted by

$$\Delta\mu(t) = \mu - \sum_{i \in B(t)} \bar{\Lambda}_i. \tag{28}$$

The well-known Generalized Processor Sharing [28, 26] is a fair scheduling rule as defined in the Lemma that guarantees a minimal service rate $\underline{\mu}_i \geq \bar{\Lambda}_i$, and that allocates this excess capacity proportional to the minimal service rates. We review the definition:

Definition 2 *Generalized Processor Sharing with weight vector $\phi \geq 0$, denoted GPS(ϕ), serves class i at rate:*

$$\begin{aligned} \mu_i(t) &= \frac{\phi_i}{\sum_{j \in B(t)} \phi_j}\mu \qquad && \textit{if } i \in B(t), \\ &= 0 && \textit{otherwise.} \end{aligned} \tag{29}$$

Since $B(t) \subseteq I$, GPS guarantees each class a minimal service rate of

$$\underline{\mu}_i^{GPS} = \frac{\phi_i}{\sum_{j \in I} \phi_j}\mu, \tag{30}$$

regardless of the presence of other backlogged classes. Thus GPS is "fair" (Parekh and Gallager [27] present additional attractive properties of GPS). To offer the minimal service rate $\underline{\mu}_i \geq \bar{\Lambda}_i$ the weight vector should satisfy:

$$\frac{\phi}{\sum_{j \in I} \phi_j} \geq \frac{1}{\mu}\bar{\Lambda}. \tag{31}$$

Thus, a simple choice is $\phi = \frac{1}{\mu}\bar{\Lambda} \geq \rho$, where $\bar{\Lambda}_i$ is specified via (23) implying that all jobs are constraint by a leaky-bucket[6], for which the previous deterministic analysis shows that both loss and delay can be guaranteed. (If, as mentioned earlier, the parameters $(\bar{\sigma}_i, \bar{\Lambda}_i)$ are not known and flows are not regulated, there is no simple or apparent choice for the weight vector ϕ, which diminishes the attractiveness of GPS in such setting.)

Like G$c\mu$, GPS is a work-conserving scheduling discipline. As any processor sharing rule, however, GPS is "ideal" in the sense that it assumes that arbitrary small amounts of data from each of the I different jobs can be served in cyclic order. In reality, a packet-based version called "Weighted Fair Queuing" [9] or PGPS [27] is implemented[7]. The actual service rate $\mu_i(t)$ is constant during intervals where the set $B(t)$ of backlogged jobs is constant. Thus, during such intervals, GPS is a static rule and the allocation of excess capacity $\Delta\mu(t)$ is independent of the actual queucount vector $N(t)$. The word "generalized" refers to arbitrary weights ϕ, as opposed to the specific choice of $\phi_i = \phi_j$ in traditional uniform processor sharing, and to the dynamic update of that "constant" rate due to the externality effect, the interference with others jobs which alters over time. Let, for any $\xi \in (t_1, t_2)$,

[6]Subjecting flows to a bustiness constraints $(\bar{\sigma}_i, \bar{\Lambda}_i)$ is, however, not a prerequisite for the operation of GPS. If each flow's traffic intensity ρ_i is smaller than ϕ_i for all flows i served under the GPS discipline, Borst *et al.* [4] have demonstrated that the tail behavior of the buffer content of an individual flow with long-tailed traffic characteristics is equivalent to the tail behavior of that flow when served in isolation at constant rate (the link rate minus the aggregate rate of all other sources). The condition $\rho_i < \phi_i$ essentially ensures that each flow is served at minimum rate $\underline{\mu}_i$ which is necessary for stability because it avoids significant queue build-up. Hence, also their analysis shows that some kind of input control (e.g. via a token bucket) is necessary.

[7]Due to the desirable properties of GPS for integrated services networks, many implementable algorithms that approximate GPS have been proposed. The class of these packet-approximation algorithms for the GPS discipline has been recently [22, 6, 34] called *packet fair queueing (PFQ)* algorithms, while, in 1995, Goyal *et al.* [12] have proposed the name *Guaranteed Rate Scheduling Algorithms* for a subclass of PFQ. PFQs do not make the fluid flow assumption of infinitesimally small packet sizes. The first presented and best-known PFQ is Weighted Fair Queueing, which is identical to Packet-by-packet Generalized Processor Sharing (PGPS) proposed by Parekh and Gallager [27]

$M_i(t_1, t_2) = \int_{t_1}^{t_2} \mu_i(\tau)\, d\tau = \mu_i(\xi)(t_2 - t_1)$ denote the amount of work served for a i-job in the interval (t_1, t_2) during which the set of backlogged jobs $B(t)$ does not change and, on (29), $\mu_i(t) = \mu_i(\xi)$. Via integration of (29), it follows that GPS ensures that for any two flows k and l back-logged during (t_1, t_2) holds

$$\frac{M_k(t_1,t_2)}{\phi_k} = \frac{M_l(t_1,t_2)}{\phi_l} = \mu \int_{t_t}^{t_2} \frac{d\tau}{\sum_{j\in B(\tau)} \phi_j}$$

or,

$$\frac{\mu_k(\xi)}{\phi_k} = \frac{\mu_l(\xi)}{\phi_l} = \text{cte} = \frac{\mu}{\sum_{j\in B(\xi)} \phi_j} \tag{32}$$

Thus, GPS provides inherent fairness, where fairness is measured relative to the amount of resources that are reserved for each job as indicated by the weight vector ϕ.

6 On the relationship between $Gc\mu$ and GPS

We showed that G$c\mu$ is the scheduling rule that provides optimal quality levels by minimizing the total expected delay costs $DC = \sum_{i\in I} \Lambda_i \mathrm{E}\left[C_i(d_i)\right]$, where the delay cost functions $C_i(\cdot)$ are the basic building block to model QoS (section 2). We showed that G$c\mu$ is a dynamic priority policy that gives full bandwidth to class $i^*(t)$:

$$\begin{aligned} \mu_i^{Gc\mu}(t) &= \mu \qquad && \text{if } i = i^*(t), \\ &= 0 && \text{otherwise.} \end{aligned}$$

GPS, on the other hand, is a fair scheduling rule that can guarantee deterministic delay bounds in a burstiness constrained, deterministic fluid model. GPS is an attractive scheduling rule not only because it is fair and allows worst-case delay guarantees, but also because of its simplicity, flexibility from changing the weight vector, and analyzability [27]. In this section we consider the following questions: are G$c\mu$ and GPS related and how can their unique strengths be combined?

By comparing (32) and (19), it is tempting to conclude that $c_i\left(\frac{W_i}{\rho_i}\right) = \frac{1}{\phi_i}$ or that the delay cost function used in GPS should be linear $C_i(t) = \frac{t}{\phi_i}$. We know, however, that with linear delay costs, the G$c\mu$ rule becomes a static priority rule (the $c\mu$-rule) where only the class with largest index $i^*(t)$ (20, 21) is served, while GPS shares capacity among all backlogged classes $B(t)$ proportional to ϕ_i of each class i. Thus, GPS and G$c\mu$ as defined above are different as they are the result to different optimization problems. G$c\mu$ was designed to minimize delays and processor sharing is strictly suboptimal with regards to minimizing delays[8]:

Proposition 3 *Priority service yields smaller delays and delay costs than processor sharing*

`Proof`: Consider the following interchange argument: consider two jobs A and B of class 1 and 2 with service times (packet lengths) l_1 and l_2. Assume first that both jobs arrived at an empty system at the same time $t = 0$. Assuming strict priority service, say to class 1, we have that

$$d_A^{static} = \frac{l_1}{\mu} \text{ and } d_B^{static} = d_A^{static} + \frac{l_2}{\mu} = \frac{l_1 + l_2}{\mu}.$$

Under "real" sharing over the two jobs, that is GPS with weight vector $(\phi_1, \phi_2 = 1 - \phi_1)$ and $0 < \phi_1 < 1$, where we assume (otherwise just switch labels A and B) that $\frac{l_1}{\phi_1} \le \frac{l_2}{\phi_2}$ (from which follows that $\frac{l_1}{\phi_1} \le \frac{l_1 + l_2}{\phi_1 + \phi_2} = l_1 + l_2 \le \frac{l_2}{\phi_2}$), we have that

$$d_A^{static} = \frac{l_1}{\mu} < d_A^{GPS} = \frac{l_1}{\mu\phi_1} \le d_B^{GPS} = d_A^{GPS} + \frac{l_2 - \frac{\phi_2}{\phi_1} l_1}{\mu} = \frac{l_1 + l_2}{\mu} = d_B^{static}, \tag{33}$$

[8]We are grateful to Professor Tsitsiklis, who inspired this result through an example.

so that priority service strictly improves class delays, and hence total delay costs. The same argument applies to any two jobs currently in service: switching from GPS to static priority strictly decreases the delay of the priority job (and of all jobs arriving during the busy cycle of that priority class), without increasing the delay of the non-priority job (and of all jobs arriving during the busy cycle). Hence, delays and delay costs can be decreased by switching to priority service for two such jobs; therefore, priority service dominates processor sharing in general.■

Thus, when it comes to end-to-end delay minimization, strict dynamic priority service dominates processor sharing and G$c\mu$ dominates GPS to minimize total delay costs. GPS, on the other hand, was designed to enforce fairness among classes; fairness in the sense that each class receives a minimal bandwidth at all times that the class is non-empty. Serving all classes in one FIFO queue is not fair because one bursty arrival from a given class makes all subsequent other class arrival wait. (In our proof, the waiting times for A and B are zero under GPS, where (obviously) one job has strictly positive waiting time under priority service.)

While G$c\mu$ and GPS are different, an alternative implementation of G$c\mu$ that satisfies the fairness criterion makes GPS as subclass of G$c\mu$:

Definition 3 *The processor-shared G$c\mu$ rule, denoted G$c\mu$-PS, allocates capacity as*

$$\forall i \in B(t) : \mu_i(t) = \begin{cases} \Lambda_i & \text{if } i \neq i^*(t), \\ \Lambda_i + \Delta\mu(t) & \text{if } i = i^*(t). \end{cases} \tag{34}$$

By construction, G$c\mu$-PS is fair, but different from *generalized* processor sharing with weights $\phi_i = \Lambda_i$. While both rules allocate a minimal service rate $\underline{\mu}_i = \Lambda_i$ to each non-empty class i, the difference stems from the allocation of the excess[9] capacity $\overline{\Delta}\mu(t)$. Under GPS this excess capacity is distributed over all non-empty classes in fixed proportion, regardless of the queue count. Under G$c\mu$-PS, however, class $i^*(t)$ gets *all* excess capacity, exactly in an attempt to minimize total delay costs. It should be clear from our discussion above that G$c\mu$ will yield lower total delay costs than G$c\mu$-PS because the latter gives up some cost performance to satisfy an additional fairness requirement. It also is clear that G$c\mu$-PS is a delay cost minimizing fair rule; as a dynamic processor sharing rule, it outperforms GPS with respects to cost minimization. G$c\mu$-PS can also be interpreted as a dynamic GPS rule: it is attractive because it automatically adjusts its processor sharing fractions depending on the state of the system to provide optimal service levels (in the context of our model). As such, G$c\mu$-PS combines the unique strengths of GPS and G$c\mu$. As a fair rule, it can guarantee delay bounds in a deterministic setting with burstiness constraints as we showed in section 5. As a rule that implements optimal QoS levels, it automatically adjusts processor sharing fractions. This may be important in stochastic environments where it is hard to come up with the burstiness constraints parameters $(\bar{\sigma}_i, \bar{\Lambda}_i)$ and a good choice of the GPS weight vector ϕ, which would incorporate effective bandwidth allocations. G$c\mu$-PS automatically implements optimal quality levels in our model without needing complex effective bandwidths. Finally, the fact that *G$c\mu$* and G$c\mu$-PS apply to non-stationary and finite horizon settings makes it appealing to current multimedia environments.

Within the framework of the IETF's Diffserv, G$c\mu$-PS appears to be a promising scheduling rule for both Expedited Forwarding (EF) and Assured Forwarding (AF). The virtual leased line class, EF, needs admission control and benefits from a minimum service rate while the AF classes are more appropriately scheduled in the line of the economic optimization criterion that lead to *G$c\mu$*. Hence, the minimum service rates Λ_i in (34) for EF-class jobs can be determined via admission control (e.g. based on a token bucket). The remaining server capacity $\Delta\mu(t)$ can be devoted to the four AF classes that are differentiated via suitably chosen delay cost functions $C_i(.)$.

[9]The idea of redistributing excess capacity in GPS has been proposed earlier by Duffield *et al.* [10]. The trigger in [10] was to question the need of queueing systems to emulate GPS closely. The resulting schedulers are different modifications of GPS. Our rule G$c\mu$-PS, on the other hand, combines two different optimization criteria: minimization of total expected delay cost and fairness with minimal service rate guarantees.

7 Conclusion

We have presented an integrated approach to pricing and scheduling for services that are differentiated in terms of throughput, delay and loss specifications. The key building block to the model are quality value curves that specify the user value of higher quality levels. From the analysis emerges a pricing rule that charges based on rate and quality grade, and a dynamic scheduling rule, called the G$c\mu$ rule. The analysis also derives the economically-optimal probabilistic QoS guarantee parameters.

We have reviewed the deterministic approach of burstiness constraints and fair scheduling rules to deterministic QoS guarantees. The scheduling that arises from this analysis is the well-known GPS. A comparative analysis inspires the fair G$c\mu$-PS rule as the scheduling rule that combines the unique strengths of GPS and G$c\mu$. We propose the G$c\mu$-PS rule as a tailored scheduling solution for both the EF-class and the AF-classes in Diffserv.

Our analysis begs for further study along several dimensions. It would be interesting to further investigate the relationship between the delay-cost minimization approach imbedded in the G$c\mu$ rule to the concepts of effective bandwidths. Also, an algorithmic, packet-based implementation of the G$c\mu$-PS rule may be designed and its performance may be compared to the ideal version presented here. Finally, our analysis of G$c\mu$ at the node level should be extended to the network case. By focusing on the fair G$c\mu$-PS, it is expected that such network analysis may be analytically tractable under burstiness-constraints.

References

[1] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated services. *¡draft-ietf-diffserv-framework-02.txt¿*, Febr. 1999.

[2] Dimitris Bertsimas. The optimality of the G$c\mu$ rule under quadratic holding costs, December 1999. Private conversation.

[3] S. Blake, D. Black, M. Calson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *RFC 2475*, Dec. 1998.

[4] S. Borst, O. Boxma, and P. Jelenkovic. Generalized processor sharing with long-tailed traffic sources. *International Teletraffic Conference, ITC 16*, pages 345–354, June 1999.

[5] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem. Analysis of end-to-end delay measurements in internet. *Proceedings of Passive and Active Measurement: PAM-2002, March 25-27, Fort Collins, Colorado.*

[6] H. J. Chao, Y.-R. Jenq, X. Guo, and Lam C. H. Design of packet-fair queueing schedulers using a RAM-based searching engine. *IEEE JSAC*, 17(6):1105–1126, June 1999.

[7] Costas Courcoubetis. Pricing and economics of networks. Tutorial presented at IEEE InfoCom Conference, San Francisco. Available from www.ics.forth.gr/˜courcou, March 1998.

[8] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Trans. Information Theory*, 37(1):pp. 114–131, Jan. 1991.

[9] A. Demers, S. Keshav, and S. Shenkar. Analysis and simulation of a fair queueing algorithm. *Internet Res. and Exper.*, 1, 1990.

[10] N. G. Duffield, T. V. Lakshman, and D. Stiliadis. On adaptive bandwidth sharing with rate guarantees. *IEEE INFOCOM'98*, pages 1122–1130, 1998.

[11] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *To appear in Automatica, 35*, 1999.

[12] P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay bounds in heterogeneous networks. *Prc. 5th Intl. Workshop on Network and Operating System support for Digital Audio and Video*, April 1995.

[13] R. Guerin and V. Peris. Quality-of-service in packet networks: basic mechanisms and directions. *Computer Networks*, 31:169–189, 1999.

[14] Rasoul Haji and G. F. Newell. Optimal strategies for priority queues with nonlinear costs of delay. *SIAM J. Appl. Math.*, 20(2):224–240, March 1971.

[15] G. Hooghiemstra and P. Van Mieghem. Delay distributions on fixed internet paths. *sumbitted to IEEE/ACM Transactions on Networking*, 2002.

[16] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *J. of the Operational Research Society*, 49:237–252, 1998.

[17] S. Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network.* Addison-Wesley, Reading, Massachusetts, 1997.

[18] L. Kleinrock. *Queueing Systems*, volume 2 : Computer Applications. John Wiley and Sons, N.Y., 1976.

[19] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: Utility functions, random losses and ECN marks. *IEEE INFOCOM2000*, 2000.

[20] J.-Y. Le Boudec and P. Thiran. *Network Calculus, A Theory of Deterministic Queuing Systems for the Internet.* LNCS 2050. Springer Verlag, Berlin, 2001.

[21] Phillip J. Lederer and Lode Li. Pricing, production, scheduling and delivery-time competition. *Operations Research*, 45(3):407–420, 1997.

[22] J. Liebeherr and D. E. Wrege. Priority queue schedulers with approximate sorting in output-buffered switches. *IEEE JSAC*, 17(6):1127–1144, June 1999.

[23] Y. Masuda and S. Whang. Dynamic pricing for network service: Equilibrium and stability. *Management Science*, 45:857–869, 1999.

[24] Haim Mendelson and S. Whang. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, 38:870–883, 1990.

[25] Jan Naudts. Towards realtime measurement of traffic control parameters. *Computer Networks*, 34:157–167, 2000.

[26] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks - the single node case. *IEEE INFOCOM'92*, pages 915–924, 1992.

[27] A. K. Parekh and R. G. 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, June 1993.

[28] A. K. J. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* Ph.D. thesis. MIT, Laboratory for Information and Decision Systems, Cambrigde, 1992.

[29] I. Ch. Paschalidis and J. N. Tsitsiklis. Congestion-dependent pricing of network services. *IEEE/ACM Trans. On Networking*, 8(22):171–184, April 2000.

[30] Martin I. Reiman. The heavy traffic diffusion approximation for sojourn times in jackson networks. In R.L. Disney and T.J. Ott, editors, *Applied Probability-Computer Science, The Interface (Volume II)*, pages 409–422, Boston, 1982. Birkhauser.

[31] H. Ren and K. Park. Toward a theory of differentiated services. *Proc. IEEE/IFIP Int. Workshop on QoS*, 2000.

[32] H Sariowan, R. L. Cruz, and G. C. Polyzos. SCED: A Generalized Scheduling Policy for Guaranteeing Quality of Service. *IEEE/ACM Trans. on Networking*, 7(5):669–684, October 1999.

[33] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *RFC 2212*, Sept. 1997.

[34] D. C. Stephens, J. C. R. Bennett, and H. Zhang. Implementating scheduling algorithms in high-speed networks. *IEEE JSAC*, 17(6):1145–1158, June 1999.

[35] I. Stoica, H. Zhang, and T. S. E. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. *IEEE/ACM Trans. On Networking*, 8(22):185–199, April 2000.

[36] Jan A. Van Mieghem. Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *Ann. Appl. Prob.*, 5(3):809–833, 1995.

[37] Jan A. Van Mieghem. Price and service discrimination in queueing systems: Incentive compatibility of G$c\mu$ scheduling. *Management Science*, 2000. to appear.

[38] A. Van Moffaert, D. De Vleeschauwer, J. Janssen, M. J. C. Buchli, G. H. Petit, and P. Coppens. Tuning the voip gateways to transport international voice calls over a best-effort ip backbone. *Proceedings of the 9th IFIP Conference on Performance Modelling and Evaluation of ATM & IP Networks (IFIP01), Budapest (Hungary), 27-29 June*, pages 193–205, 2001.

[39] W. Willinger, M. S. Taqqu, and A. Erramilli. A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks. *Stochastic Networks, Theory and Applications, edited by F. P. Kelly, S. Zachary and I. Ziedens, Clarendon Press, Oxford*, pages 339–366, 1996.

[40] J. Wroclawski. Specification of the Controlled-Load Network Element Service. *RFC 2211*, Sept. 1997.