An Architecture for Noncooperative QoS Provision in Many-Switch Systems

Shaogang Chen Kihong Park
Network Systems Lab
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract—**With the proliferation of high-speed networks and networked services, provisioning differentiated services to a diverse user base with heterogeneous QoS requirements has become an important problem. The traditional approach of resource reservation and admission control provides both guarantees and graded services, however, at the cost of potentially underutilized resources and limited scalability. In this paper, we describe a WAN QoS provision architecture that adaptively organizes best-effort bandwidth into stratified services with graded QoS properties such that the QoS needs of a diverse user base can be effectively met.**

Our architecture—SBS (Stratified Best-effort Service)—promotes a simple user/simple network realization where neither the user nor the network is burdened with complex computational responsibilities. SBS is scalable, efficient, and adaptive, and it complements the guaranteed service architecture, sharing a common network substrate comprised of GPS routers. It is also a functional complement, provisioning QoS efficiently commensurate with user needs, albeit at the cost of weaker protection. SBS is suited to noncooperative network environments where users behave selfishly and resource contention resolution is mediated by the principle of competitive interaction. A principal feature of SBS is the transformation of user-centric QoS provision mechanisms—a defining characteristic of competitive interaction entailing intimate user control of internal network resources—into network-centric mechanisms while preserving the former's resource allocation paradigm.

End-to-end QoS control is facilitated by decentralized control based on Lagrangian optimization—achieve a target end-to-end QoS at minimum cost or resource usage—which, in turn, is amenable to distributed implementation. SBS achieves per-flow QoS control with zero per-flow state at routers and a packet header whose size is independent of hop count. SBS, in spite of foregoing both resource reservation and admission control, is able to provision stable, graded QoS.

I. Introduction

A. Motivation

With the advent of global network infrastructures and its emerging role as an important enabling platform for services spanning commerce, entertainment, education, and a compendium of everyday activities, building an efficient network architecture capable of supporting diverse user needs has become a critical problem.

The traditional approach to QoS provision uses resource reservation and admission control such that a traffic stream's data rate and burstiness can be suitably accommodated by a network. Although research abounds [1], [2], [3], analytic tools for computing QoS guarantees rely on shaping of input traffic to preserve well-behavedness across switches which implement some form of packet scheduling discipline such as generalized processor sharing (GPS), also known as weighted fair queuing [4], [5]. Real-time constraints of multimedia traffic and the scale-invariant burstiness associated with self-similar network traffic [6] limit the shapability of input traffic while at the same time reserving bandwidth that is significantly smaller than the peak transmission rate. Thus QoS and utilization stand in a trade-off relationship with each other [7] and transporting application traffic over reserved channels, in general, incurs a high cost.

This makes it important to organize today's best-effort bandwidth, as examplified by the Internet, into *stratified* services with graded QoS properties such that the QoS requirements of a compendium of applications can be effectively met. This is particularly useful for applications that possess diverse but—to varying degrees—flexible QoS requirements. It would be overkill to transport such traffic over reserved channels. On the other hand, relying on homogeneous best-effort service, characteristic of today's Internet, would be equally unsatisfactory. A dual architecture capable of supporting reserved and stratified best-effort service is needed which, in turn, helps amortize the cost of inefficiencies stemming from overprovisioned resources for guaranteed traffic.

Another important consideration when designing QoS provision mechanisms are issues surrounding fairness, stability, and optimality. With respect to fairness, a principal question centers on "who should get what and how much." Resource contention resolution must address the normative issue of networks with users possessing *diverse* QoS requirements where users *should* receive unequal share of resources commensurate with their QoS requirements or other well-defined criteria. From a network efficiency perspective, how to induce users to consume just enough resources to satisfy their QoS needs and thus leave a *maximal* pool of resources for others to use is a problem of central import. Intimately tied to fairness is the issue of stability. A resource assignment deemed fair by some criterion may not be sustainable unless there is a form of consensus, and even if so, it may not be reachable from all initial configurations. Instability, in general, has an adverse effect on QoS—a user's end-to-end QoS may jump from one value to another—and providing stable, predictable service is an important task.

B. New Contribution

In [8], we give a game-theoretic analysis of the multi-class QoS provision system for noncooperative *single-switch* network

S.C.: Supported in part by NSF grant ANI-9714707; chensg@cs.purdue.edu.
K.P.: Supported in part by NSF grants ANI-9714707 and ESS-9806741, and grants from PRF and Sprint; contact author, park@cs.purdue.edu.

systems showing when Nash equilibria exist and under what conditions they are Pareto and/or system optimal. In [9], we advance a multi-class QoS provision architecture for *many-switch* systems which extends the single-switch system to the many-switch case using single-switch reduction based on the notion of selfishness emulation.

In this paper, we focus on the distributed QoS control problem associated with noncooperative many-switch systems by formulating the many-switch QoS assignment problem as a constrained optimization problem—which is NP-hard even for a single user—and transforming the optimization problem into an unconstrained form using the framework of Lagrangian multipliers. The main advantage of the Lagrangian formulation is its decoupled form—i.e., across switches—modulo a confined dependence introduced by Lagrangian multipliers which leads to an approximation procedure that is amenable to efficient decentralized implementation. Distributed control consists of two parts, one, local optimization at every switch which chooses a service class that meets the local QoS responsibility while minimizing resource usage, and two, global optimization via an end-to-end feedback loop that adjusts the Lagrangian multipliers—shared across all switches—to satisfy a target end-to-end QoS at minimum cost. Interestingly, the solution procedure obtained using the Lagrangian framework turns out to be isomorphic to the distributed QoS control proposed in [9] which is based on single-switch reduction, an approach that achieves QoS control by generalizing an optimal solution procedure for the single-switch case to the many-switch case.

We demonstrate the efficacy of our many-switch QoS architecture and its distributed control by simulating a WAN environment based on a vBNS-like topology with multiple traffic flows possessing diverse QoS requirements. We compare the performance of our architecture—SBS (Stratified Best-effort Service)—against a reservation scheme that a priori allocates fixed service classes based on assumed knowledge of all traffic flows—their traffic characteristics and QoS requirements—as well as FIFO packet scheduling and random service class assignment. We show the robustness of our architecture by generating problem instances with varying problem difficulty from the space of probability distributions over user QoS requirements falling into three categories—"easy," "intermediate," and "difficult." We show that our architecture performs favourably when compared to the aforementioned QoS provision schemes.

C. Related Work

Significant work has been carried out in formulating resource allocation problems spanning a number of different domains using tools from microeconomics and game theory [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. The models and approaches proposed in the literature differ along several dimensions, some of the important ones being whether applications or users are assumed to be cooperative or selfish, whether pricing is used or not, and how much computing responsibility is delegated to the user. Several papers have addressed the issue of multi-class QoS provision in high-speed networks [10], [20], [21], [18], [17]. Some of the works employ a cooperative framework or place significant computing responsibilities on the part of the user [21], [17], some investigate the effect of pricing incentives [10], and others represent flow/congestion control and routing models [20], [16], [18] that partially address the quality of service provision problem whose essence is users possessing *diverse* QoS requirements. QoS provision architectures for noncooperative WAN environments is still in its infancy with the *Smart Market* [22] and our previous work in noncooperative many-switch QoS provision based on single-switch reduction [9] representing two instances. In [22], pricing—in the form of packets carrying bids—is used to resolve scheduling conflicts of packets at switches inside a network implementing priority queues. With respect to its core features, the Smart Market is similar to a simplified version of our many-switch architecture where instead of QoS indicators—used to dynamically select different service classes at every switch—a *fixed* priority or service class label is propagated to every switch.

A related development are efforts directed at designing network architectures with the aim of delivering differentiated service with "soft" or weak forms of guarantees using aggregate—as opposed to per-flow—traffic control [23], [24]. Of particular interest are Assured Service [23] and Premium Service [24] which affect weak protection through traffic shaping/marking and support from routers. In both cases, it is assumed that service level—i.e., QoS—is computed using admission control, and the core task revolves around providing protection from ill-behaving flows that exceed their contract specifications. This is done through 2-state (in/out) marking with the help of RIO gateways (a form of RED gateway with dual thresholds) [23], or through leaky bucket traffic shaping with gateways implementing priority queuing [24]. Our own work ([25], [8] and present), couched in the context of providing graded QoS provision with weak guarantees, takes a different approach. Whereas [23], [24] concentrate on providing protection through explicit traffic shaping/policing, our architecture concentrates on providing graded QoS with protection handled by implicit admission control through usage pricing. An important objective of SBS is the elimination of *explicit admission control*—except in the provision of guaranteed services—as a mechanism for per-flow QoS control, an impediment to scalability as detrimental as maintaining per-flow state at routers. SBS is a scalable QoS provision architecture that uses neither resource reservation nor admission control when organizing best-effort bandwidth into graded service commensurate with user needs.

The rest of the paper is organized as follows. In the next section, we define the many-switch multi-class QoS provision problem leading to a resource allocation problem in the noncooperative context. This is followed by a decription of the many-switch QoS provision architecture based on the Lagrangian formulation, its properties, and decentralized implementation. Section IV shows performance results depicting the behavior of SBS. We give a comparative evaluation in the context of reservation, FIFO, and random service class assignment schemes, and for problem instances of varying difficulty.

II. Multi-Class QoS Provision Problem

A. Network Model

Assume a network comprising of a set of routers and end stations connected via some topology. The routers implement GPS packet scheduling [4], [5] where packets labeled by their service class number receive service commensurate with the resources allocated for that service class and the traffic impinging on that service class. If every application flow is mapped to a unique service class at every switch, then the service class number is synonymous with flow ID and the system can be viewed as implementing per-flow QoS control. If the mapping is many-to-one, QoS control is exercised on an aggregate flow basis. Other things being equal, the larger the service weight or the smaller the aggregate traffic flowing into a service class, the better the QoS—e.g., as measured by delay, packet loss rate, jitter—rendered by that class[1].

Assuming fixed routes, the end-to-end QoS experienced by an application flow is determined by the service levels received at each of the routers along a path which, in turn, is determined by the service class assignments—possibly different—at each of the routers. There is a calculus for computing end-to-end QoS in terms of the QoS rendered locally at each of the switches, e.g., with packet loss behaving multiplicatively and delay behaving additively. For example, if c^k denotes the packet loss rate at switch $k \in [1, r]$ on an r-hop path, then the end-to-end packet loss rate is given by $1 - \prod_{k=1}^{r} (1 - c^k)$. Assuming there are m (in general, m_k) service classes at every switch $k \in [1, r]$, then to flow i there correpond r choice variables $\xi_i^k \in [1, m]$ which determine which service class application flow i is assigned to at hop k. This is shown in Figure II.1.

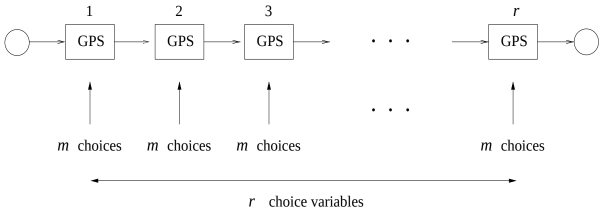

Fig. II.1. End-to-end QoS—given a fixed route—is determined by the local QoS rendered at each of the r switches which, in turn, is determined by the service class assignment at each switch.

Routing introduces a new set of decision variables; in this paper, we will confine ourselves to the case where routing is handled by a separate subsystem, i.e., is given.

B. User Model

Assume n users or applications where each user $i \in [1, n]$ has a traffic demand given by its mean data rate λ_i. The most important property associated with a user in the QoS provision context is its *QoS requirement* which may be different for each user. For example, if QoS requirements were represented by bounds on packet loss rate, delay, and jitter, then a user i with a bound $\theta^i = 33$ms on end-to-end delay would have a more stringent QoS requirement than a user i' who has a more relaxed delay bound of $\theta^{i'} = 200$ms. In general, a user's QoS requirement can be represented by a utility function U_i which captures the "satisfaction" experienced by user i when receiving a certain QoS. Utility functions are a tool to represent heterogeneous user preferences and facilitate reasoning about the bahavior of a system.

Fixing a switch $k \in [1, r]$, user i's flow λ_i^k (note $\lambda_i^1 = \lambda_i$ and $\lambda_i^1 \geq \lambda_i^2 \geq \cdots \geq \lambda_i^r$) can be chosen by the user—or by the network—to be assigned to one or more of the m different service classes at k. This assignment of "where and how much" is represented by user i's service class assignment vector $\Lambda_i^k = (\lambda_{i1}^k, \lambda_{i2}^k, \ldots, \lambda_{im}^k)^T$ where $\lambda_{ij}^k \geq 0$ and $\sum_j \lambda_{ij}^k = \lambda_i^k$. Thus, the aggregate flow entering into service class $j \in [1, m]$ is given by $q_j^k = \sum_i \lambda_{ij}^k$. In this paper, we will be interested in the *unsplittable* case where $\lambda_{ij}^k \in \{0, \lambda_i^k\}$, for all $j \in [1, m]$. For the unsplittable case, the choice variables ξ_i^k defined in Section II-A completely determine the QoS that a flow will receive at the routers. Both the splittable and unsplittable case for the *single-switch* network (i.e., $r = 1$) are studied in [8].

One last item to define is the behavioral mode of a user—selfish or cooperative. *Selfishness*, in our context, will mean that each application $i \in [1, n]$ will try to take actions—i.e., setting $\boldsymbol{\xi}_i = (\xi_i^1, \xi_i^2, \ldots, \xi_i^r)^T$ assuming it is allowed to do so—so as to maximize its individual utility U_i.

Remark II.1 The network substrate described in Section II-A, can be used to provide both guaranteed and graded services through resource reservation and admission control. *Guaranteed* here means the protection—i.e., from interference by other flows—afforded to a flow by GPS through resource reservation and admission control. Stratified best-effort service, by adopting a *weaker* form of protection, consequently also exports QoS to the user that is, in general, more variable and subject to the other flows' potentially detrimental influence.

C. Noncooperative QoS Provision Game

C.1 Many-Switch Network Game

Fix user $i \in [1, n]$ and assume its traffic is assigned a route with $r \geq 1$ hops or switches. We will use $k \in [1, r]$ as the switch index. The user has a choice of r selection variables $\xi_i^k \in [1, m]$ where $\xi_i^k = j$ indicates that user i has selected to channel his traffic through service class j at switch k. Let $\boldsymbol{\xi}_i$ denote user i's service class assignment vector and let $\boldsymbol{\xi}$ denote the service class assignment of all users.

The *end-to-end QoS* received by user i, $\boldsymbol{x}^i \in \mathbb{R}^s$ (s denotes the number of QoS indicators), is a function of $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n)^T$,

$$\boldsymbol{x}^i = \boldsymbol{x}^i(\boldsymbol{\xi}),$$

and given i's traffic demand λ_i, we arrive at the individual utility $U_i(\lambda_i, \boldsymbol{x}^i)$, consistent with the single-switch formulation [8], [9]. Nash equilibria, Pareto optima, and system optima can be defined with respect to $U_i(\lambda_i, \boldsymbol{x}^i)$.

Remark II.2 A configuration is a *Nash equilibrium* if each user cannot improve his individual lot through unilateral actions affecting his traffic allocations. Thus if every player finds himself in such a "local optimum," then from the noncooperative perspective, the system is at an impasse—i.e., stable rest point. A configuration is a *Pareto optimum* if in order to improve the lot of some player, the lot of others must be sacrificed. A configuration is *system optimal* if the sum of the individual lots is maximized. A formal definition can be found in [8].

[1] There is a subtle QoS ordering effect with respect to jitter which is discussed in [26].

C.2 Network Game with Pricing

Pricing is introduced as a mechanism for monitoring relative resource usage by imposing the relation

$$\boldsymbol{c}_a^k \succeq \boldsymbol{c}_b^k \implies p_a^k \geq p_b^k, \qquad a, b \in [1, m], \tag{II.3}$$

where $\boldsymbol{c}_j^k$ is the QoS rendered in service class j at switch k and p_j^k is its price. I.e., the superior the QoS, the higher the price. Other things being equal, for GPS switches, $\boldsymbol{c}_a^k \succeq \boldsymbol{c}_b^k$ iff the relative resource consumption per unit flow is higher in service class a than service class b. Notice the difference with congestion-based pricing schemes where high demand—i.e., congestion—leads to a higher price irrespective of the actual QoS rendered. Our pricing scheme is *fine-granular* in the sense that pricing occurs at every switch on a per-flow basis.

The *cost* or *relative resource usage* accrued to user i's traffic flow can be defined in a number of ways including the following three: $\lambda_i^{r+1} \sum_{k=1}^r p^k_{\xi_i^k}$, $\sum_{k=1}^r p^k_{\xi_i^k} \lambda_i^k$, and $\lambda_i \sum_{k=1}^r p^k_{\xi_i^k}$. Here λ_i^{r+1} denotes the net flow arriving at the receiver. By monotonicity, we have

$$\lambda_i^{r+1} \sum_{k=1}^r p^k_{\xi_i^k} \;\leq\; \sum_{k=1}^r p^k_{\xi_i^k} \lambda_i^k \;\leq\; \lambda_i \sum_{k=1}^r p^k_{\xi_i^k}. \tag{II.4}$$

Relation (II.4) collapses into a single expression when packet loss is zero. Assuming the above expressions are used to compute the ultimate cost exported to a user via composition with a monotone price function $\mathcal{P} : \mathbb{R}_+ \to \mathbb{R}_+$, $\lambda_i \sum_{k=1}^r p^k_{\xi_i^k}$ can be interpreted as being the most "service provider-friendly" measure whereas $\lambda_i^{r+1} \sum_{k=1}^r p^k_{\xi_i^k}$ is the most "user-friendly one."

For *threshold* utilities, one version of the noncooperative many-switch QoS provision game with pricing is given by

$$\min_{\boldsymbol{\xi}_i} \;\; \lambda_i \sum_{k=1}^r p^k_{\xi_i^k} \tag{II.5}$$

subject to $\boldsymbol{x}^i(\boldsymbol{\xi}) \leq \boldsymbol{\theta}^i$. That is, the user seeks a minimum cost assignment that satisfies the user's QoS requirement. This problem, even for a *single user* is NP-hard—it can be reduced to a version of multiple choice knapsack—and approximate solutions need to be sought.

Theorem II.6 (Many-Switch QoS Assignment) *The per-user constrained many-switch QoS assignment problem given by* (II.5) *is NP-hard.*

III. Many-Switch QoS Provision Architecture

The noncooperative QoS provision game is defined by solving the constrained optimization problem (II.5) for each user (or player) $i \in [1, n]$. In a *user-centric* realization of the game, each user is assumed to have direct control over its choice variables ξ_i^k, $k \in [1, r]$. In a *network-centric* realization of the game, the network solves (II.5) *on behalf* of each user, resulting in an equivalent outcome. The advantage of the latter is that users need not have access to detailed network state—an unrealistic assumption in WAN environments comprised of numerous switches—and users are shielded from engaging in complex computations. Emulation of selfish user behavior is also the network's *contract* to the user with respect to its modus operandi thus preserving noncooperativeness.

The primary user-network interface is a *QoS requirement vector* $\boldsymbol{\theta}^i \in \mathbb{R}^s$ which represents user i's bounds on desired end-to-end QoS. The network system takes $\boldsymbol{\theta}^i$ as input and tries to deliver end-to-end QoS $\boldsymbol{x}^i$ such that $\boldsymbol{x}^i \leq \boldsymbol{\theta}^i$, at least cost.

A. Distributed QoS Control: Lagrangian Formulation

A.1 Transformation to Normalized Constrained Form

The Lagrangian method starts from the abstract optimization formulation (II.5). It transforms the constrained optimization problem into an equivalent unconstrained form. This transformation leads to a set of independent optimization problems—one for each switch—which are coupled only by a set of common Lagrangian multipliers. The latter have a simple interpretation: the larger the multipliers' value, the more stringent the QoS rendered by the system. Nonuniformity in the QoS rendered at different switches is facilitated by local optimization which is modulated by the coupling constants, i.e., multipliers. The decoupled form lends itself to efficient distributed implementation: the same local optimization procedure—parameterized by the multipliers—is executed at all switches.

The first task is to transform the minimization problem (II.5) into an equivalent maximization problem and, in tandem, transform the end-to-end QoS constraints $\boldsymbol{x}^i(\boldsymbol{\xi}) \leq \boldsymbol{\theta}^i$ into an additive form. Let $u_j^k = p_0 - p_j^k$, $j \in [1, m], k \in [1, r]$, where p_0 is a positive constant such that $p_0 > \max_{j,k} p_j^k$. For packet loss rate and delay, $\boldsymbol{x}^i(\boldsymbol{\xi}) \leq \boldsymbol{\theta}^i$ takes on the forms

$$1 - \prod_{k=1}^r (1 - c^k_{\xi_i^k}) \;\leq\; \theta_c^i,$$

$$\sum_{k=1}^r (d_{\xi_i^k} + L_k + T_k) \;\leq\; \theta_d^i,$$

respectively. Using the following change of variables,

$$w_{j1}^k = -\ln(1 - c_j^k), \quad b_1^i = -\ln(1 - \theta_c^i),$$
$$w_{j2}^k = d_j + L_k + T_k, \quad b_2^i = \theta_d^i,$$

the constraint components for packet loss rate and delay can be written in an additive form as $\sum_{k=1}^r w^k_{\xi_i^k h} \;\leq\; b_h^i$, $h = 1, 2$. The end-to-end expression for jitter $v_{\xi_i^k}$ is more complex depending on its precise definition. We employ the strong form $\max_k v_{\xi_i^k} \;\leq\; \theta_v^i$ which is equivalent to r constraints $v_{\xi_i^k} \;\leq\; \theta_v^i$, one for each $k \in [1, r]$.

Thus the network game given by (II.5) can be rewritten as

$$\max_{\boldsymbol{\xi}_i} \;\; \sum_{k=1}^r u^k_{\xi_i^k} \tag{III.1}$$

subject to $\sum_{k=1}^r w^k_{\xi_i^k h} \;\leq\; b_h^i$, $h \in [1, l]$, where u_j^k, w_{jh}^k, b_h^i are positive, and l is the number of constraints. The normalized constrained optimization problem represents the multidimensional multiple-choice knapsack problem (MMKP). A special case of MMKP is the multiple-choice knapsack problem (MCKP) which has only a single constraint, i.e., $l = 1$. Both MCKP and MMKP are known to be NP-complete.

A.2 Transformation to Unconstrained Form

Transformation of the normalized constrained optimization problem (III.1) into an equivalent unconstrained optimization problem is facilitated by the next theorem which has been adopted from [27] to fit our context.

Theorem III.2 (Lagrangian Optimization) *Let $\mu^1_i, \ldots, \mu^l_i$ be l non-negative Lagrange multipliers and $\boldsymbol{\xi}^*_i$ a solution of*

$$\max_{\boldsymbol{\xi}_i} \Big\{ \sum_{k=1}^{r} u^k_{\xi^k_i} - \sum_{h=1}^{l} \mu^h_i \sum_{k=1}^{r} w^k_{\xi^k_i h} \Big\}.$$

Then, $\boldsymbol{\xi}^*_i$ is a solution to

$$\max_{\boldsymbol{\xi}_i} \sum_{k=1}^{r} u^k_{\xi^k_i}$$

subject to $\sum_{k=1}^{r} w^k_{\xi^k_i h} \le b^{i*}_h$ *where* $b^{i*}_h = \sum_{k=1}^{r} w^k_{\xi^{k*}_i h}$.

To understand the above theorem, we can consider (III.1) an optimization problem parameterized by b^i_h. Theorem III.2 allows us to solve an unconstrained optimization problem to obtain an optimal solution to a constrained optimization problem with parameters b^{i*}_h which are not necessarily equal to b^i_h, the parameters of the original constrained problem (III.1). Different values of b^{i*}_h are obtained when we vary the multipliers μ^h_i's and thus b^{i*}_h is a function of the μ^h_i's. If the multipliers μ^h_i are known such that b^{i*}_h coincides with the original QoS requirements, i.e., $b^{i*}_h = b^i_h$ for all h, the original optimization problem is solved. Since for fixed μ^h_i, the conditional of Theorem III.2 can be solved exactly in polynomial time, the hardness of the problem comes into play via μ^h_i.

A.3 Approximation Procedure for Computing μ_i

For MCKP, μ_i can be perturbed systematically so that a global maximum of the objective function is obtained. This is possible because of the following simple fact.

Proposition III.3 *As μ_i increases in the interval $[0, +\infty]$, b^{i*} decreases monotonically.*

In a centralized algorithm, we could find the range of μ_i and use binary search to make b^{i*} as close to b^i as desired. A special form of MMKP—i.e., QoS vectors are in total order—also possesses the monotonicity property. The is given by the next proposition.

Proposition III.4 *Let the QoS vectors w^k_{jh} at every switch k obey a total order. Then as $\mu^{h_0}_i$ increases in the interval $[0, +\infty]$, b^{i*}_h decreases monotonically for all h.*

A.4 Decentralized Implementation of Lagrangian Optimization

The unconstrained optimization problem, for fixed μ^h_i, is straightforward to solve. To see this, notice that the unconstrained optimization problem can be rewritten as

$$\max_{\boldsymbol{\xi}_i} \sum_{k=1}^{r} (u^k_{\xi^k_i} - \sum_{h=1}^{l} \mu^h_i w^k_{\xi^k_i h}).$$

For given μ^h_i's, the terms $(u^k_{\xi^k_i} - \sum_{h=1}^{l} \mu^h_i w^k_{\xi^k_i h})$, $k \in [1, m]$, are independent of each other and can be solved separately. In other words, the sum is maximized iff each individual term is maximized. Thus the solution is given by

$$\xi^k_i = \text{argmax}_j \{ u^k_j - \sum_{h=1}^{l} \mu^h_i w^k_{jh} \}. \qquad \text{(III.5)}$$

The decentralized approximation procedure consists of two parts, one, the local optimization carried out at every switch k based on (III.5), and two, the global optimization carried out via an end-to-end feedback loop that exploits Proposition III.4. First, with respect to computing (III.5), u^k_j and w^k_{jh} are variables depicting the state of switch k hence locally available. Thus the only nonlocal information needed for computing (III.5) are the μ^h_i's. This can be done by enscribing μ^h_i in the packet header. The packet header is shown in Figure III.1.

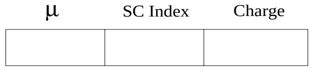

Fig. III.1. Packet header format for Lagrangian method.

The `SC` (service class) `Index` field is used to enscribe the service class label computed by (III.5) which is then forwarded to the GPS switch proper. The `Charge` field is used to accrue the relative resource usage cost according to (II.4). Figure III.2 shows the structure of a GPS switch augmented by the procedure for computing (III.5).

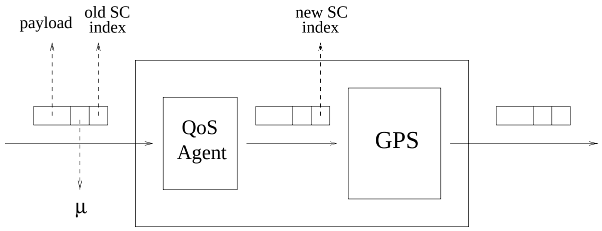

Fig. III.2. GPS switch augmented by QoS agent. The latter intercepts incoming packets, chooses a service class assignment for the packet, and forwards it to the GPS scheduler.

Now to the update of μ_i. The Lagrangian multiplier is adjusted in an end-to-end feedback loop using Proposition III.4. Specifically, the receiver of the end-to-end control measures the end-to-end QoS rendered at the receiver and compares this value with the target or desired end-to-end QoS which is available at the receiver. If the rendered end-to-end QoS is excessively good vis-à-vis the target QoS, then μ_i is decreased to affect a deprovement in QoS. If the achieved QoS is unsatisfactory vis-à-vis the target QoS, then μ_i is increased to affect an improvement in QoS except when doing so has proved futile—i.e., a feasible solution may not exist.

Let $\epsilon = (b^i - b^{i*})/b^i$. The update rule for μ_i is given by

$$\mu_i \leftarrow \begin{cases} \mu_i, & \text{if } \mu_* \le \epsilon \le \mu^*, \\ \max\{\mu_i - \beta, 0\}, & \text{if } \epsilon < \mu_*, \\ \mu_i + \beta, & \text{if } \epsilon > \mu^* \text{ and } d\epsilon/d\mu_i < 0, \\ \mu_i, & \text{otherwise,} \end{cases}$$

where $0 < \mu_* \le \mu^*$ and $\beta > 0$ are fixed parameters. A further refinement that sets the magnitude β adaptively is omitted here due to space constraints. In the case when there are two or more QoS requirements, we update the Lagrangian multiplier corresponding to the QoS indicator experiencing the worst performance. The end-to-end part of the decentralized QoS control is depicted in Figure III.3.

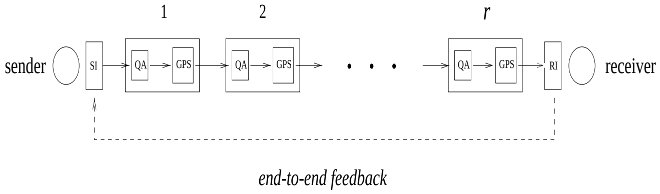

Fig. III.3. Distributed QoS control over an r-hop path.

B. Desirable Properties

Following is a summary of some of SBS' desirable architectural properties:

- *Simple User/Network Interface* The user's interface with the network system is narrow and well-defined. The user conveys its QoS requirement vector to the network and the network tries to deliver the target end-to-end QoS at least cost—i.e., resource consumption—to the user. A service provider may use the resource usage signature maintained by the network to set the service price exported to the user; alternatively, it may override it.
- *Zero Per-Flow State at Routers* SBS exercises per-flow QoS control with zero per-flow state at routers. This is conducive to scalability. SBS packet headers are of constant size independent of hop count.
- *Fine Granular Resource Usage Signature* The network, as a by-product of its protocol, maintains a fine granular account of resource usage by each flow. This information can be used by the service provider to set its service price advertised to the end user or it may be used as an internal tool to track resource usage.
- *Compatibility with Guaranteed Service Architecture* SBS runs on top of generic GPS-based internetworks, and its transparent handling of traffic flows makes it compatible with guaranteed service provisioning through resource reservation and admission control.

It is interesting that the distributed QoS control obtained from the Lagrangian formulation is isomorphic to the one arrived at using the single-switch reduction approach. We omit its discussion due to space constraints.

IV. Performance Evaluation

A. Simulation Set-Up

A.1 Network Configuration

We use the LBNL Network Simulator *ns* (version 2) as the basis of our simulation environment. We have modified *ns* in order to model our multi-class QoS provision architecture. This entailed, among other things, implementing the QoS Agents as separate modules invoked by the routing agent, and implementing a GPS packet scheduler module with extended functionalities. We show results for a vBNS-based network topology which is depicted in Figure IV.1. In the current vBNS—an NSF-sponsored backbone network designated for networking and high-performance computing research—the backbone runs at OC-12 (∼622Mbps) with link latency ranging from 3ms to 15ms. Some of the links that connect member institutions to the backbone are OC-3 (∼155Mbps) with link latency of about 1ms, whereas others are DS-3 (e.g., Purdue University). We make two simplifications: one, we only consider OC-3 drops from the backbone to institution nodes, and two, we scale down *all* bandwidths uniformly by a factor of 10 to reduce simulation overhead.

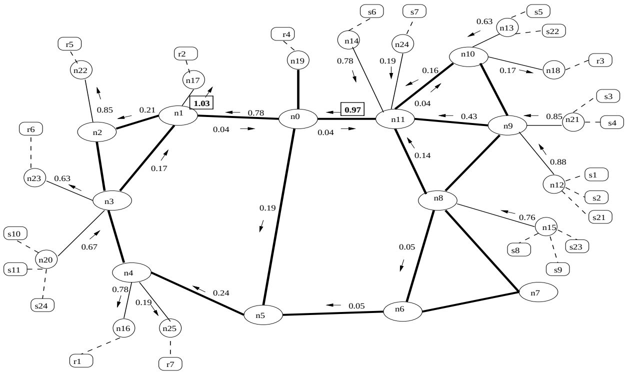

Fig. IV.1. vBNS-like wide area network topology.

A.2 Traffic Configuration

The traffic configuration is shown in Table I. There are in total 15 individual flows of which 11 are application traffic with various QoS constraints (shown in separate QoS requirement configuration tables), and 4 are background traffic which are used to inject further resource contention and system variability. This particular traffic configuration has been engineered to create a number of localized *hot spots* where several connections are multiplexed onto the same output link thereby potentially causing severe contention. There are two main bottlenecks. The first one is from n_{11} to n_0 and has a load factor or utilization of 0.97. The second one is from n_1 to n_{17} and has a load factor of 1.03. Both are shown highlighted in Figure IV.1.

Traffic Flow				
	Flow No.	Src.	Dest.	Rate (Mbps)
Application	1	s1	r2	2.99
	2	s2	r4	8.53
	3	s3	r5	10.24
	4	s4	r5	2.99
	5	s5	r6	3.98
	6	s6	r1	12.05
	7	s7	r4	2.99
	8	s8	r2	2.99
	9	s9	r7	2.99
	10	s10	r2	7.88
	11	s11	r3	2.59
Background	12	s21	r2	2.05
	13	s22	r6	5.85
	14	s23	r4	5.85
	15*	s24	r3	4.81

TABLE I

Traffic Configuration

Flow	Pls. Req.	Lagrangian				SWR				Fixed Class			
		pls.	vio.	charge	U/P	pls.	vio.	charge	U/P	pls.	vio.	charge	U/P
1	.05	.0347	.07	49.65	.0187	.0365	.03	49.64	.0195	.0487	**.43**	49.51	.0115
2	.05	.0321	.00	39.68	.0252	.0341	.00	39.66	.0252	.0321	.00	39.68	.0252
3	.05	.0319	.00	59.68	.0168	.0338	.00	59.66	.0168	.0316	.00	59.68	.0168
4	.01	.0004	.01	60.00	.0165	.0002	.01	60.00	.0165	.0000	.00	60.00	.0167
5	.01	.0003	.01	60.00	.0165	.0003	.01	60.00	.0165	.0000	.00	60.00	.0167
6	.05	.0322	.00	49.68	.0201	.0341	.00	49.66	.0201	.0320	.00	49.68	.0201
7	.01	.0003	.01	30.00	.0330	.0004	.01	30.00	.0330	.0000	.00	30.00	.0330
8	.05	.0355	.08	49.65	.0185	.0363	.11	49.65	.0179	.0490	**.43**	49.51	.0115
9	.01	.0000	.00	50.00	.0200	.0000	.00	50.00	.0200	.0000	.00	50.00	.0200
10	.05	.0272	.01	29.74	.0333	.0264	.00	29.74	.0336	.0175	.00	29.83	.0335
11	.001	.0000	.00	60.00	.0167	.0000	.00	60.00	.0167	.0000	.00	60.00	.0167
		$\sum U/P = \mathbf{.235}$				$\sum U/P = .235$				$\sum U/P = .222$			

TABLE II

LAGRANGIAN METHOD, SINGLE SWITCH REDUCTION, AND THE FIXED ALLOCATION SCHEME.

A.3 Problem Instance Generation

To test the robustness of our algorithm and compare its performance against that of reservation- and FIFO-based schemes, we employ a problem instance generator that takes Table I as input and outputs the QoS requirements associated with the application flows. We use a random service class assignment method for this purpose whereby for each flow, at every switch, the flow is randomly assigned (permanently) to a fixed service class. We then let the system evolve in time and observe the measured end-to-end QoS received by each application flow. These values—mean packet loss, mean delay, and jitter—are then chosen as “handles” in generating final problem instances.

They are handles in the sense that they are subsequently used to generate three separate problem instances which we classify as *resource-plentiful* (alternatively “easy”), *resource-scarce* (alternatively “difficult”), and *in-between*. Since we can always shift or translate the QoS requirement indicators to make the QoS assignment problem easier (upwards shift) or harder (downwards shift) given a fixed resource configuration, by the shifting operation we are able to identify instances where the problem instance changes its character from instrinsically easy to intermediate to difficult. For the “easy” and “difficult” problem categories, we choose (by trial-and-error) instances that are as close to the intermediate stage (i.e., transition) as possible to avoid generating trivial instances for which no algorithm can find a feasible QoS assignment or almost all algorithms can do so.

B. Lagrangian Method, Single Switch Reduction and Fixed Reservation-Based Allocation

Table II shows the performance of Lagrangian method, single switch reduction and a fixed service class assignment scheme at delivering end-to-end QoS given the flows’ QoS requirements. All switches possess three service classes of which one is set aside for the background traffic. The user population consists of 0.05-, 0.01-, and 0.001-packet loss rate applications (column 2), and the end-to-end packet loss rate they received is shown in the column marked *pls.*. *vio.* denotes the fraction of instances, *over time*, that a QoS requirement is violated, and *charge* denotes the total cost accrued by all packets belonging to the same connection. Comparing Lagrangian method and single switch reduction, we observe that the QoS requirements of all flows are satisfied. In terms of minimizing cost, they achieve comparable $\sum U/P$, a measurement of how well the optimization problem is solved.

The *Fixed Class* algorithm is a static, fixed service class allocation scheme where the service class assignment, once picked, is held invariant during a connection’s lifetime. It is an off-line, centralized QoS allocation scheme that assigns a strict priority ordering based on a connection’s QoS requirement, giving higher precedence to stringent-QoS applications over less stringent ones. A service class assignment is then chosen such that this precedence condition is satisfied at every switch for every flow. The Fixed Class algorithm is used as a reference point in evaluating the relative performance. We observe that the former satisfies the end-to-end packet loss rate requirements for all connections. However, one noticeable difference with Lagrangian method and single switch reduction is the high violation penalty flows 1 and 8 incur under the Fixed Class assignment.

C. Dynamical Properties of Lagrangian Method

Flow 10 is one of the connections that goes through one of the two bottleneck links, (n_1, n_{17}). Figure IV.2 (top) show the trace of the end-to-end QoS experienced by flow 10. We can see that flow 10’s QoS requirement 0.05 is satisfied without a single violation. Figure IV.2 (bottom) shows flow 10’s service class assignment at bottleneck switch n_1. We observe that flow 10 is assigned to class 2 stays there throughout.

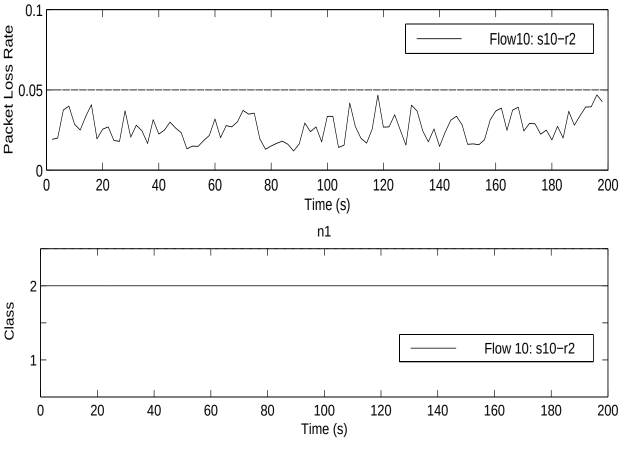

Fig. IV.2. Dynamics of flow 10. Top: Trace of end-to-end packet loss. Bottom: Trace of service class assignment on n_1.

Flow 1 is one of the connections that travels through both of the two bottleneck links. Figure IV.3 (top) shows the end-to-end packet loss trace of flow 1. We observe that there are four brief instances when the QoS requirement is violated. Figure IV.3 (middle) shows that on switch n_{11} flow 1 is assigned to service class 2. Figure IV.3 (bottom) shows the service class assignment trace for flow 1 on switch n_1. We observe that although flow 1 is most of the time assigned to the superior service class (here it is class 1), occasionally there are excursions to service class 2 which, in turn, are correlated to the elevation in the end-to-end packet loss rate.

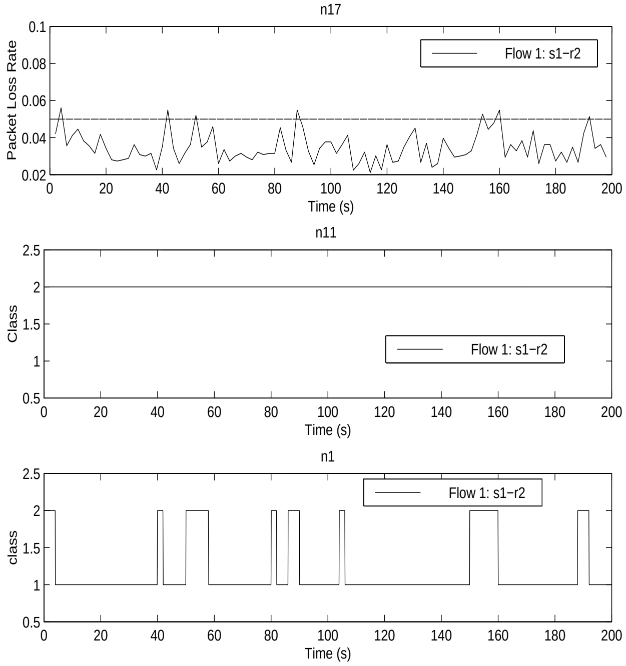

Fig. IV.3. Dynamics of flow 1. Top: Packet loss trace. Middle: Service class assignment trace on (n_{11}, n_0). Bottom: Service class assignment trace on (n_1, n_{17}).

The reason for the temporary excursions is that when the QoS Agent at n_1 notices that the end-to-end QoS delivered is overly good for flow 1, then service class 2 becomes the locally optimal choice for flow 1, but this reassignment is not sustainable in the long run without violating flow 1's QoS requirement. Thus the switch back to service class 1 momentarily thereafter. We omit the description of the behavior of other flows due to space constraints.

D. Robustness and Comparative Performance

In this section, we show the robustness of our algorithm on three classes of problem instances—resource plentiful ("easy"), in-between, and resource scarce ("difficult")—with six instances in each category generated using the random service class assignment scheme with shifting. We also compare the performance of our algorithm against the fixed reservation-based scheme as well as a FIFO-based scheme and random (fixed) allocation scheme. On average, we expect the random fixed allocation scheme to perform similar to the FIFO scheme due to the uniform nature of the service weights. Table III summarizes the comparative QoS performance results. The numbers inside the table indicate the number of application flows whose QoS requirements are violated. Thus "0" means that QoS requirements of *all* applications were satisfied.

We observe that for instance in the "easy" problem class, our algorithm (Lagrangian) performs perfectly with the fixed reservation-based scheme close behind. However, both FIFO- and random service class assignment-based schemes exhibit violations in the range 2–5. This is so since although we classify this problem class as "easy", the resource plentiful and resource scarce instances were chosen to be as close to the intermediate problem instances as possible. Thus what is easy to Lagrangian method and the fixed reservation-based scheme is not necessarily easy for the FIFO and random schemes. For the in-between instances we observe violations occurring even for Lagrangian (0–2 range) and for the resource scarce instances. Overall, we observe that Lagrangian is superior to the fixed reservation-based scheme and both are superior to the FIFO and random schemes. For the in-between cases, Lagrangian method performs much better than the single switch reduction scheme. On the other hand, for the resource plentiful and resource scarce instances, they have achieved almost the same results.

E. Comparison of Single Switch Reduction and Lagrangian Method

Table IV shows the same comparative performance results on three of problem type—resource plentiful, in-between, and resource scarce—for three-dimensional QoS requirement vectors containing packet loss rate, delay, and jitter. The numbers in the table show the number of flows whose QoS requirements were violated. We observe that the Lagrangian method performs comparably with single switch reduction.

Instance	Resource Plentiful		In-between		Resource Scarce	
	Lagr.	SWR	Lagr.	SWR	Lagr.	SWR
1	0	0	3	4	6	4
2	0	0	1	1	3	2
3	0	0	2	2	3	6
4	0	0	3	0	3	3
5	0	0	2	2	6	2
6	0	0	3	2	4	4

TABLE IV

THREE-DIMENSIONAL QOS VECTOR: PACKET LOSS RATE, DELAY, AND JITTER.

F. User Population QoS Diversity

Table V shows a user pool with diverse QoS requirements with packet loss bounds of 0.001, 0.005, 0.008, 0.01, 0.02, and 0.05, and the QoS rendered by the system. All QoS requirements are being met for both Lagrangian method and the Fixed Assignment scheme. This is in the context of a 2-service class system. In general, for a diverse user population make-up more service classes are needed to provide refined services.

V. CONCLUSION

We have presented an architecture for noncooperative multi-class QoS provision in many-switch systems. End-to-end QoS control is facilitated by decentralized control based on Lagrangian optimization which, in turn, is amenable to distributed

Instance	Resource Plentiful					In-between					Resource Scarce				
	Lagr.	SWR	fix.	fifo	ran.	Lagr.	SWR	fix.	fifo	ran.	Lagr.	SWR	fix.	fifo	ran.
1	0	0	0	4	4	0	2	1	2	1	5	5	4	4	5
2	0	0	0	2	2	0	1	2	2	2	2	3	7	3	5
3	0	0	0	3	2	1	1	0	5	2	4	4	3	5	4
4	0	0	0	5	2	1	3	0	3	3	3	3	3	3	4
5	0	0	1	4	3	3	0	1	5	3	4	3	3	5	4
6	0	0	0	4	3	2	2	1	4	3	3	5	4	4	3

TABLE III

COMPARATIVE PERFORMANCE WITH RESPECT TO THREE CLASSES OF PROBLEMS—RESOURCE PLENTIFUL, IN-BETWEEN, AND RESOURCE SCARCE—WITH SIX PROBLEM INSTANCES IN EACH CATEGORY.

implementation. An appealing property of our architecture is that selfish user mode is preserved without burdening the user with complex resource allocation computations. This imparts our system with a well-defined and practical resource allocation paradigm—selfishness—on which resolution of resource contention conflicts are based.

Flow	Pls. Req.	Lagrangian			Fixed Class		
		pls.	vio.	charge	pls.	vio.	charge
1	.05	.0347	.07	49.65	.0487	.43	49.51
2	.05	.0321	.00	39.68	.0321	.00	39.68
3	.05	.0319	.00	59.66	.0316	.00	59.68
4	.005	.0004	.01	60.00	.0000	.00	60.00
5	.02	.0003	.01	60.00	.0000	.00	60.00
6	.05	.0322	.00	49.68	.0320	.00	49.68
7	.008	.0003	.01	29.99	.0000	.00	30.00
8	.05	.0355	.08	49.65	.0490	.43	49.51
9	.01	.0000	.00	50.00	.0000	.00	50.00
10	.05	.0272	.01	29.74	.0175	.00	29.83
11	.001	.0000	.00	60.00	.0000	.00	60.00

TABLE V

DIVERSE QOS REQUESTS

Complementing the *simple user* approach, our distributed control protocol is easily implementable as a preprocessing step in routers with GPS packet scheduling. Pricing and accounting are performed at a fine-granular level (per packet) where usage-based pricing is efficiently achieved. This leads to a *simple network* realization. SBS is scalable, efficient, and adaptive, and it complements the guaranteed service architecture, sharing a common network substrate comprised of GPS routers. It is also a functional complement, provisioning QoS efficiently commensurate with user needs, albeit at the cost of weaker protection.

REFERENCES

[1] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE J. Select. Areas Commun.*, vol. 13, no. 6, pp. 1048–1056, 1995.

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

[3] 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*, vol. 2, no. 2, pp. 137–150, 1994.

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

[5] 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*, vol. 1, no. 3, pp. 344–357, 1993.

[6] 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*, vol. 2, pp. 1–15, 1994.

[7] 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*, 1996, pp. 171–180.

[8] 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*, 1998, pp. 111–127.

[9] 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*, 1998, pp. 98–107.

[10] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in computer networks: motivation, formulation, and example," *IEEE/ACM Trans. Networking*, vol. 1, no. 6, pp. 614–627, 1993.

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

[12] 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*, 1988, pp. 491–499.

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

[14] J. Kurose and R. Simha, "A microeconomic approach to optimal resource allocation in distributed computer systems," *IEEE Trans. on Computers*, vol. 38, no. 5, pp. 705–717, 1989.

[15] S. Low and P. Varaiya, "A new approach to service provisioning in ATM networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 5, pp. 547–553, 1993.

[16] A. Orda, R. Rom, and N. Shimkin, "Competitive routing in multiuser communication networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 5, pp. 510–521, 1993.

[17] 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*, 1995, pp. 1111–1119.

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

[19] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta, "Spawn: a distributed computational economy," *IEEE Trans. Software Engineering*, vol. 18, no. 2, pp. 103–117, 1992.

[20] Y. Korilis and A. Lazar, "On the existence of equilibria in noncooperative optimal flow control," *Journal of the ACM*, vol. 42, no. 3, pp. 584–613, 1995.

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

[22] J. MacKie-Mason and H. Varian, "Economic FAQs about the Internet," in *Internet Economics*, L. McKnight and J. Bailey, Eds., pp. 27–63. MIT Press, 1996.

[23] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 362–373, 1998.

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

[25] S. Chen and K. Park, "An architecture for noncooperative QoS provision in many-switch systems," Tech. Rep. CSD-TR-98-013, Department of Computer Sciences, Purdue University, 1998.

[26] 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*, 1998, pp. 252–265.

[27] Hugh Everett III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 2, pp. 399–417, 1963.