1998

Molding Best-Effort Bandwith into Graded Services: The SBS Approach

Kihong Park
Purdue University, park@cs.purdue.edu

Shaogang Chen

Report Number:
98-040

MOLDING BEST-EFFORT BANDWIDTH INTO GRADED SERVICES: THE SBS APPROACH

Kihong Park
Shaogang Chen

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR #98-040
November 1998

Molding Best-Effort Bandwidth into Graded Services: The SBS Approach

Kihong Park* Shaogang Chen†
Network Systems Lab
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{park,chensg}@cs.purdue.edu

CSD-TR 98-040
Dept. of Computer Sciences
Purdue University

Abstract

With the increased deployment and commercialization of networked services, provisioning differentiated services to a user or application base with diverse 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 commensurate with the QoS needs of a diverse user base. Our architecture—SBS (stratified best-effort service)—complements, and is compatible with, the guaranteed service architecture sharing a common network substrate comprised of GPS routers. SBS is scalable, efficient, and adaptive, exporting graded services molded to match the QoS needs of the current application base. SBS' strength, in part, stems from forfeiting resource reservation and admission control, however, at the cost of exporting services with weaker protection properties.

SBS is suited to noncooperative network environments where users are selfish and resource contention resolution—i.e., the "who should get what" problem—is mediated by the principle of competitive interaction. SBS shields the user from complex computational responsibilities and guards the network mechanism from direct user control by emulating selfish user behavior inside the network. It does so by executing a self-optimization procedure which is user optimal in the single-switch case and approximates the NP-hard QoS assignment problem in the many-switch case. SBS achieves a simple user/simple network realization using decentralized QoS control.

Keywords: QoS Architectures, Differentiated Services, Selfishness Emulation, Distributed QoS Control

*Supported in part by NSF grants ANI-9714707 and ESS-9806741, and grants from PRF and Sprint.
†Supported in part by NSF grant ANI-9714707.

1 Introduction

1.1 Motivation

Quality of service (QoS) provision in high-speed networks carrying a diverse mixture of traffic from e-mail to bulk data to voice, audio, and video is a difficult problem. The traditional approach uses resource reservation and admission control to provide both *guarantees* and *graded* services to application traffic flows. Analytical tools for computing and provisioning QoS guarantees [5, 11, 12, 6] rely on overprovisioning coupled with traffic shaping/policing to preserve well-behavedness properties across switches that implement a form of generalized processor sharing (GPS) packet scheduling.

The scale-invariant burstiness associated with self-similar network traffic [8, 13] limits the shapability of input traffic while at the same time reserving bandwidth that is significantly smaller than the peak transmission rate. This imposes a trade-off relationship between QoS and resource utilization which limits the degree of utilization achievable while guaranteeing stringent QoS.

For applications needing guaranteed services, the unconditional protection afforded by reservation and admission control is a requirement. However, for QoS-sensitive applications that require services with graded QoS but admit to weaker forms of protection, it would be overkill to provision QoS over reserved channels. In addition to the service mismatch, overhead associated with administering resource reservation and admission control which require per-flow QoS control and state at routers impedes the scalability of the system. On the other hand, relying on homogenous best-effort service, characteristic of today's Internet, would be equally unsatisfactory. Our goal is to organize best-effort bandwidth into stratified services with graded QoS properties such that the QoS needs of a diverse user base can be effectively met. In so doing, we seek to accommodate as much of the QoS-sensitive application traffic via our stratified best-effort service (SBS) architecture as is feasible.

1.2 New Contributions

Recently, significant effort has been directed at designing network architectures with the aim of delivering differentiated services with "soft" or weak forms of guarantees [4, 9, 10, 1]. Of particular interest are Clark's Assured Service [4] and Jacobson's Premium Service [10]—two principal proposals discussed by the IETF Diff-Serv Working Group—which affect weak protection through traffic shaping/policing 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, either through 2-state (in/out) marking with the help of RIO gateways (a form of RED gateway with dual thresholds) [4], or through traffic shaping (leaky bucket) with gateways implementing priority queueing [10]. In both schemes, protection is weakened due to uncertainties introduced by *aggregate* traffic control where routers inside the network are impervious to per-flow information.

Our own work [1, 2, 14] takes a different approach to graded QoS provision with weak guarantees. First, whereas [4, 10] concentrate on providing *protection* through explicit traffic shaping/policing, the SBS 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: we believe (explicit) admission control is a significant impediment to scalability as is maintaining per-flow state at routers. The main contribution of our work may be viewed as designing a scalable QoS provision architecture that uses neither resource reservation nor admission control when organizing best-effort bandwidth into graded services commensurate with user needs.

Second, whereas Assured and Premium service can be viewed as "bottom-up" approaches to architecting differentiated services in the sense that design considerations such as adherence and compatibility with IP-based internetworks in the form of the number of bits needed to encode both services and their placement in IPv4 and IPv6 headers are given priority, SBS is more of a "top-down" approach which shares with the two approaches goals such as scalability through aggregate traffic control but starts from a more distilled slate—network of GPS switches and selfish users with diverse QoS requirements—and tries to design mechanisms with emphasis on achieving a set of normative goals derived from basic modeling assumptions.

2 WAN QoS Provision Problem: Set-Up

2.1 Network Model

Assume a network comprising of a set of routers and end stations connected via some topology. The routers implement GPS packet scheduling [7, 11] 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 and jitter 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 correspond r choice variables $\xi_i^k \in [1, m]$ which determine which service class application flow i is assigned to at hop k. Figure 2.1 depicts this situation. 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.

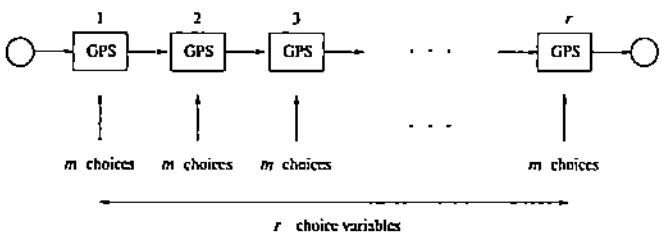


Figure 2.1: End-to-end QoS—given a fixed route—is determined by the local QoS rendered at each of the r switches which, in turn, is determined by the service class assignment at each switch.

[1]There is a subtle QoS ordering effect with respect to jitter which is discussed in [3].

2.2 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 just a tool to represent heterogeneous user preferences and facilitate reasoning about the bahavior of a system, and are not meant to be taken literally nor even be assumed measurable with sufficient accuracy.

Fixing a switch $k \in [1,r]$, user i's flow λ_i^k (note $\lambda_i^1 = \lambda_i$) 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 2.1 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 [14].

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 $\xi_i = (\xi_i^1, \xi_i^2, \ldots, \xi_i^r)$ assuming it is allowed to do so—so as to maximize its individual utility U_i.

2.3 Resource Allocation Problem

2.3.1 Objectives

The network substrate described in Section 2.1, 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. SBS' goal is to take available bandwidth—both nonreserved, and reserved but unused (hence available via GPS' work conservation)—and shape it into services with *graded* QoS properties that meet the QoS requirements of QoS-sensitive applications.

Two tasks befall upon SBS. One, export services with graded QoS properties commensurate with the diverse QoS requirements of a given application pool, and two, render these services as stably as possible. There are intrinsic limitations to achieving both objectives when the overall resource contention level is high. For example, for a given resource configuration and n users with diverse QoS requirements, there may not exist *any* $\Xi = (\xi_1, \xi_2, \ldots, \xi_n)$ that satisfies all n users' QoS requirements. For resource configurations where there exist Ξ that satisfy all n users, the solution set may be "small" such that finding a feasible Ξ is nontrivial. Indeed, we can show that even for a *single* user i, finding $\xi_i = (\xi_i^1, \xi_i^2, \ldots, \xi_i^r)$—even if it exists—that satisfies i's QoS requirements is NP-hard [1]. For those instances when feasible Ξ exist we find that they satisfy a certain *grouping* or *clustering* property. This is illustrated by the following example.

Example (*Grouping*) 2.1 Consider $n = 4$ application flows sharing an output port of a switch with

$m = 2$ service classes. Assume the application flows are only interested in packet drops and their QoS requirements are given by bounds on packet loss rate $\theta^1 = \theta^2 = 0.02$, $\theta^3 = \theta^4 = 0.06$. Assume that the four flows are i.i.d. with data rates $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4$ ($= \lambda$) and assume the service weights of the two service classes are ordered as $\alpha_1 > \alpha_2$. Suppose the multiplexed link bandwidth is such that if 2λ is input to each of the two service classes, the packet loss rates rendered by the two service classes are $c_1 = 0.01$, $c_2 = 0.05$, respectively. Consider the service class assignment where flows 1 & 3 are directed to service class 1 and flow 2 & 4 are sent to service class 2. Only three of the four flows have their QoS requirements satisfied with flow 2's requirement being violated. On the other hand, if flows 1 & 2 are directed to service class 1 and flows 3 & 4 to service class 2, all four flows are satisfied.

In general, feasible solutions tend to satisfy a clustering property where flows with similar QoS requirements are assigned to the same service class. In fact, standard clustering algorithms can be used to determine a minimal m needed to service a given user population.

2.3.2 Fairness, Stability, and Optimality

An important consideration when designing network mechanisms are issues surrounding fairness, stability, and optimality. With respect to fairness, a principal question centers on "who should get what and how much." Notions based on equal share (e.g., max-min fairness)—although useful in limited contexts and interesting in their own right—do not 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. How to induce users to consume just enough resources to satisfy their QoS needs—i.e., not overconsume—and thus leave a *maximal* pool of resources for others to use is a question of central import. In the context of Example 2.1, this corresponds to affecting a configuration where flow 3 is assigned to service class 2.

Intimately tied to fairness is the issue of stability. A configuration 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 provisioning—a user's end-to-end QoS may continuously jump from one value to another—and providing stable, predictable service is an important task. Lastly, even if a configuration is fair and stable, it may not be one that utilizes resources effectively and another configuration may exist—also fair and stable—that achieves higher utilization.

3 Noncooperative Game Approach and Self-Organization

This section outlines a *user-centric* approach to network QoS provision mechanism design unconstrained by practical constraints. It is meant to illustrate the core ideas and some relevant properties based on recent results [1, 2, 14]. Section 4 transforms the user-centric framework into a *network-centric* framework with the goal of achieving a simple user/simple network realization conducive to feasible implementation in WAN environments.

3.1 User-Centric QoS Provision Mechanism

Consider the network of GPS switches architecture described in Section 2 with N switches and n users with diverse QoS requirements represented by their utility functions U_i, $i \in [1, n]$. To every switch $k \in [1, N]$, there correspond n choice variables $\xi_1^k, \xi_2^k, \ldots, \xi_n^k$, one for each user, that determine the service class selection of each user at that switch. Given fixed routes, for each user, only those

variables corresponding to switches along the given route for a user will be of relevance. Denoting user i's choice vector (or *strategy*) by $\boldsymbol{\xi}_i = (\xi_i^k)_{k\in[1,N]}$, the collection of choice vectors $\boldsymbol{\Xi} = (\boldsymbol{\xi}_i)_{i\in[1,n]}$ determine the end-to-end QoS experienced by every user. Assuming selfish users, user i will choose $\boldsymbol{\xi}_i$ so as to optimize U_i.

To induce users to consume just enough resources to satisfy their QoS requirements but not more, we may either *assume* that users behave so or "penalize" users for not doing so. First, note that fixing a switch k and letting $\mathbf{c}_j^k$ denote the QoS rendered at service class $j \in [1,m]$ at switch k, $\mathbf{c}_j^k < \mathbf{c}_{j'}^k$ (interpreted as QoS rendered at j is superior to that of j'), implies, other things being equal, that flows belonging to service class j at switch k (i.e., $J = \{i \in [1,n] : \xi_i^k = j\}$) collectively consume more resources at switch k than flows assigned to service class j' [2]. We may, therefore, define a scalar function p_j^k to each service class and impose the ordering relation

$$\mathbf{c}_j^k \leq \mathbf{c}_{j'}^k \;\Rightarrow\; p_j^k \geq p_{j'}^k \tag{3.1}$$

and assign a relative cost $p_{\xi_i^k}^k \lambda_i^k$ to user i for using service class ξ_i^k in the amount of λ_i^k at switch k. We call $p_{\xi_i^k}^k \lambda_i^k$ the *relative usage signature* of user i at switch k. Of course, for this to have an impact on the behavior of user i, relative usage signature must impart a negative effect on U_i.

p_j^k is an internal book keeping tool used to implement the principle "the better service you receive relative to some user (as a consequence of consuming more resources) the more you will be penalized relative to that user." p_j^k can be used directly as *price* to affect a cost to the user but, in general, there is a disconnect and the ultimate cost exported by the network or service provider to the user can be any monotone function of its internal price satisfying (3.1). Computing (3.1)—a simple table maintained at every switch—and assuring that users are accorded negative utility as a function of their relative usage signature is the only responsibility required of the network in the user-centric QoS provision framework[3].

Assuming user i has been assigned the route $1 \to 2 \to \cdots \to r$, one specific form of optimization problem under selfish user behavior can be formulated as

$$\min_{\xi_i} \sum_{k=1}^{r} p_{\xi_i^k}^k \lambda_i^k \tag{3.2}$$

subject to $\mathbf{x}^i \leq \boldsymbol{\theta}^i$ where $\mathbf{x}^i = \mathbf{x}^i(\boldsymbol{\Xi})$ is the end-to-end QoS *received* by user i and $\boldsymbol{\theta}^i$ is its end-to-end QoS *requirement*. What remains is to analyze whether such selfish interactions converge to stable configurations, and if so, what kind of configurations they lead to in terms of resource allocation property. The stable fixed points of noncooperative games correspond to Nash equilibria which are configurations where no user—through *unilateral* actions—can improve its lot; i.e., every user finds itself at an impasse.

An interesting generalization to the above QoS provision game is in the form of *closing* the system by introducing one or more service providers explicitly who are able to set the magnitude of the ordinal relation (3.1) over switches in their administrative domain. Their behavioral mode, if selfish, would reduce to maximizing individual profit $\sum_{k\in D_\ell} \sum_{i=1}^{n} p_{\xi_i^k}^k \lambda_i^k$ where $D_\ell \subseteq [1,N]$ is the set of switches under the administrative governance of service provider $\ell \in [1,L]$. When service providers cooperative to collectively maximize joint profit, then this leads to an oligopoly. The closed system is under current investigation and mentioned here for completeness.

[2]There are further subtleties associated with this property including the fact that, in general, $\{\mathbf{c}_j : j \in [1,m]\}$ forms a partial order; we omit their discussion here due to space constraints and refer the reader to [3, 14].

[3]Implementing this relation effectively—i.e., correctly and efficiently—falls under the umbrella of *Secure QoS*.

3.2 Relevant Properties

Competitive interaction based on selfishness leads to a form of self-organization and the task before us is to investigate its properties. Game-theoretic analysis of the *many-switch* system is a difficult and challenging task due to coupling between switches as well as for complexity theoretic reasons. The *single-switch* case, however, is amenable to analysis using game-theoretic tools and following is a summary of its relevant properties.

Stability For the *unsplittable* game (i.e., $\lambda_{ij}^k \in \{0, \lambda_i^k\}$), we can show that there always exist Nash equilibria, and moreover, all initial configurations converge to Nash equilibria and convergence is fast [14]. As a side note, in the general *splittable* case we prove that Nash equilibria need not exist. This, and the practical issue of resequencing, are the main reasons why we concentrate on unsplittable systems where a flow cannot be separated across two or more service classes.

Fairness The "who should get what" problem, in the user-centric noncooperative game framework, is bypassed and made superfluous by the fact that the mechanism of competitive interaction *determines* the resource allocation and consequent end-to-end QoS achieved.

Optimality Related to the issue of fairness is optimality, in particular, Pareto optimality and system optimality. A configuration is Pareto optimal if the net utility of the system cannot be improved without sacrificing the utility of one or more users[4]. A configuration is system optimal if it is maximal with respect to total user utility. We can show that when resources are sufficient to satisfy all user QoS requirements, then competitive interaction leads to Nash equilibria that are both Pareto and system optimal [14]. In general, when resources are not "plentiful" given the current QoS requirements, then there exist gaps between all three configuration classes.

Grouping Grouping, as illustrated in Example 2.1, is also the typical tendency exhibited by stable configurations. This can be understood by noting that a selfish user subject to (3.1) will choose service classes that satisfy its QoS requirement at least cost which, in turn, partitions or sorts users by the stringency of their QoS requirements.

The single-switch system is used a building block in the construction of the many-switch architecture. The aforementioned properties are consistent with stability behavior observed from simulation of noncooperative many-switch systems [1, 2].

3.3 Hardness of Many-Switch QoS Control

There are two important reasons for alleviating the computational responsibility of a user and moving away from the user-centric framework. First, expecting a user to know the internal state of routers in WANs is unrealistic. Second, the optimization problem faced by each user in the form of (3.2)—even for a *single* user keeping the service class assignments of all other users fixed—is NP-hard [1]. The reduction is to multiple choice knapsack. This also trivially implies that the original n-user optimization problem is NP-hard. The hardness result has a broader scope in the sense that it states that consuming resources just enough to satisfy one's QoS requirements—but not more—is a difficult problem, even when complete network state is available and only one user is making decisions.

3.4 Selfishness Emulation

The aforementioned problem, in the *single-switch* case, however, has a simple solution. In fact, there is a user optimal (in the Nash sense) procedure for selecting service class j which, in the unsplittable

[4]It is also the stability notion for cooperative games as Nash equilibrium is to noncooperative games.

case, is given by

(i) $\mathbf{c}_j \leq \boldsymbol{\theta}_i$,

(ii) p_j is minimal among all service classes satisfying condition (i).

A generalization thereof applies in the splittable case [1]. A consequence of this is that the network can render user-optimal decisions *on behalf* of a user as well as the user would if presented with the same information and option of exercising direct control. Thus it obviates the need for the user to know the router state and affect direct control. We call this *selfishness emulation* and it is the main tool that allows us to transform the user-centric architecture into a network-centric architecture without sacrificing the properties described in Section 3.2.

4 Network-Centric QoS Provision Architecture

4.1 Distributed QoS Control

Transition to a network-centric QoS provision architecture—by selfishness emulation—is simple if there is only a single switch ($\tau = 1$) in the system. A QoS management module—SBSP (SBS Protocol)—is installed at the router which intercepts incoming packets whose headers are enscribed with the flow's QoS QoS requirement vector $\boldsymbol{\theta}^i$. SBSP inspects $\boldsymbol{\theta}^i$, computes the service class index j using the user optimal decision procedure, writes j into the header and forwards the packet to the GPS packet scheduler proper. If the entering packet is part of a guaranteed service flow, it is forwarded untouched. SBSP is transparent to the GPS switch and the guaranteed service signaling protocol (e.g., RSVP) resident on the router. This is illustrated in Figure 4.1 (left).

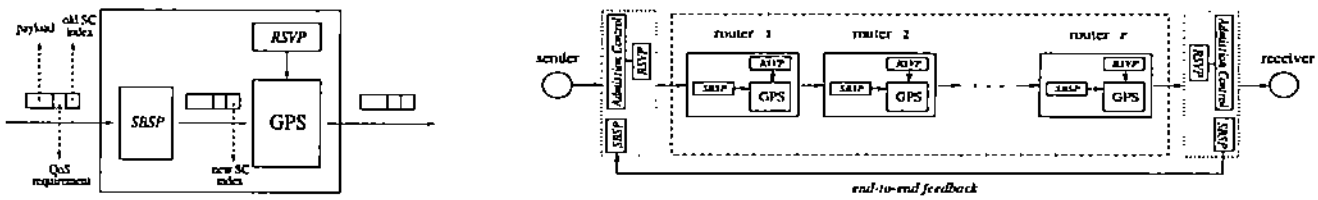


Figure 4.1: Left: SBSP-augmented GPS router. Right: Distributed QoS control over an τ-hop path comprising of SBSP-augmented GPS routers with end-to-end feedback loop.

Difficulties arise in the many-switch case due to the hardness of the service class assignment problem over an τ-hop path. We employ two methods to attack this problem: single-switch reduction [1] and Lagrangian Method [2].

4.1.1 Single-Switch Reduction

Single-switch reduction is predicated on the fact that if *optimal local QoS responsibilities* $\boldsymbol{\theta}_1^i, \boldsymbol{\theta}_2^i, \ldots, \boldsymbol{\theta}_\tau^i$ for a given end-to-end QoS requirement $\boldsymbol{\theta}^i$ were known, then by applying the user optimal decision procedure at every switch k with $\boldsymbol{\theta}_k^i$ as input, the many-switch service class assignment problem would be solved. Since these quantities, however, are not known, the single-switch reduction version of SBSP uses *uniform* local QoS responsibility as the starting point and iteratively improves the solution with respect to (3.2). End-to-end QoS is *measured* at the receiver and fed back to the sender via an end-to-end feedback loop. In conjunction with the *target* QoS requirement vector, it is made available to switches along the control path.

SBSP modules on the switches *actively* shape rendered end-to-end into the target end-to-end QoS using local decisions. In so doing, SBSP lessens the QoS responsibility of highly loaded switches at the expense of amplifying the QoS responsibility of lightly loaded ones. SBSP exercises per-flow QoS control, however, with zero per-flow state at routers and constant size headers independent of hop count. A cascade of SBSP-augmented switches is shown in Figure 4.1 (right).

4.1.2 Lagrangian Method

The Lagrangian method starts from an abstract optimization formulation. It transforms the constrained optimization 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 common multiplicative factor, the Lagrangian multiplier. The latter has a simple interpretation: the larger the multiplier's value the more stringent the QoS rendered by the system. Nonuniformity is introduced via local optimization at every switch and modulates the influence of the coupling constant. The decoupled form lends itself to efficient distribution implementation: the same local optimization parameterized by the multiplier is run at all switches.

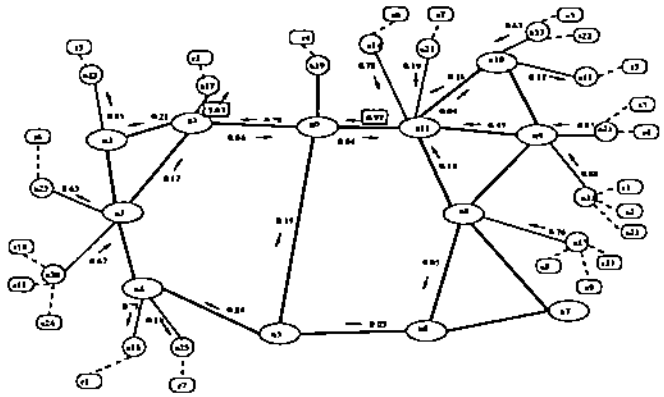
4.2 Desirable Properties

Following is a listing of SBS' desirable properties:

- *Zero Per-Flow State at Routers* SBSP—for both the single-switch reduction and Lagrangian versions—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.
- *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.
- *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* SBSP 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.

4.3 Simulation Results

Figure 4.2 (left) shows a benchmark instance for a vBNS-like network carrying a diverse set of traffic flows. Our simulations are based on the LBNL *ns* simulator. Table 4.2 (right) shows performance results of SBS with respect to the end-to-end QoS requirements of 11 SBS application flows and the QoS delivered by the network. There are also 4 guaranteed traffic flows whose QoS requirements are handled via resource reservation and admission control. They are not shown here. The QoS requirements of the application flows are given by packet loss rate and end-to-end delay. This is an easy instance—there are only two sets of QoS requirements (0.05/0.1 and 0.01/0.08)—and we find that the QoS requirements of all 11 application flows are satisfied.



flow	pls. req.	delay req.	SBS		
			pls.	delay	cost
1	.05	.100	.0365	.031	49.64
2	.05	.100	.0341	.019	39.66
3	.05	.100	.0338	.041	59.66
4	.01	.080	.0002	.040	60.00
5	.01	.080	.0003	.045	60.00
6	.05	.100	.0341	.034	49.66
7	.01	.080	.0004	.008	30.00
8	.05	.100	.0363	.032	49.65
9	.01	.080	.0000	.043	50.00
10	.05	.100	.0264	.023	29.74
11	.01	.080	.0000	.045	60.00

Figure 4.2: Left: Prototype SBS architecture tested using simulation on NSF's vBNS WAN topology. Right: Performance results for SBS with 11 application flows and dual QoS requirements.

Table 1 (left) shows comparative performance results on three classes of problems: resource plentiful ("easy"), in-between, and resource scarce ("difficult"), with 6 instances in each category. The 18 problem instances were generated randomly—thus the QoS requirements are different among all 11 application flows—and the problem instances were made easy, in-between, or difficult by shifting the QoS requirement thresholds downward. We compare the performance of SBSP against a *fixed* service class assignment scheme which uses a priori information about *all* flows—their traffic characteristics and QoS requirements—to compute a static schedule over all flows and switches in the system. The last two columns show the corresponding results for FIFO and random (but fixed) service class assignment schemes. The numbers in the table show the number of flows (among the 11 nonreserved flows) whose QoS requirements were *violated*. We observe that SBS, on average, outperforms the other three schemes. When resources become scarcer, however, the service class assignment problem becomes intrinsically difficult and satisfying all application flows becomes accordingly difficult.

4.4 Approximations

A practical consideration of import is the size of the header needed to represent service class assignment related information. In SBSP, header size is constant and does not increase with hop count nor the number of application flows. This is a nontrivial property since SBS renders per-flow QoS control inside the network at *routers* based on each flow's QoS requirement without maintaining per-flow state. To facilitate practical deployment, it is conducive to have constant size headers whose bit count is small. Reducing the absolute bit count without significantly affecting SBS' behavioral properties is not a simple problem. The QoS requirement vector in the packet header allows intimate per-flow QoS control to be exercised inside the network and the removal of such information comes at the cost of weakened ability to render refined QoS commensurate with user needs.

Inst.	Resource Plentiful				In-between				Resource Scarce			
	sbs	fix.	fifo	ran.	sbs	fix.	fifo	ran.	sbs	fix.	fifo	ran.
1	0	1	3	4	1	5	3	6	4	5	3	6
2	0	1	2	1	0	6	2	4	1	6	2	4
3	0	1	4	1	2	5	4	2	2	5	4	4
4	0	0	2	3	3	0	2	3	3	0	3	5
5	0	0	2	4	3	1	3	4	4	2	4	4
6	0	1	2	4	2	2	2	5	3	2	4	5

Inst.	Plentiful		In-between		Scarce	
	lag.	app.	lag.	app.	lag.	app.
1	0	0	0	2	5	6
2	0	0	1	1	2	2
3	0	0	1	1	4	4
4	0	0	1	3	3	3
5	0	0	3	3	4	5
6	0	0	2	2	3	3

Table 1: Left: Comparative performance evaluation with jitter QoS indicator. Right: Comparative performance evaluation of Lagrangian and Approximate Lagrangian SBS protocol.

We have investigated an approximate form of the Lagrangian version of SBSP which uses the multiplier to exercise control over QoS stringency but omits the QoS requirement vector which is needed in the full formulation. This is tantamount to exercising "differentiated services"—flows belonging to one service class receive superior service than flows in another service class—however, with degradation in power to render QoS that is attuned to each flow's individual QoS needs.

Table 1 (right) shows the performance of the full Lagrangian SBSP and Approximate Lagrangian SBSP. The performance difference can be amplified by choosing specially designed nonrandom configurations. For random configurations, we observe that there is a slight overall performance drop of Approximate Lagrangian SBSP over full SBSP. Approximate Lagrangian SBSP is similar, in some sense, to a modified and adaptive version of MacKie-Mason and Varian's *Smart Market* [9] where pricing is used to resolve scheduling conflicts of packets at switches inside a network implementing priority queues.

References

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

[2] S. Chen and K. Park. An architecture for noncooperative QoS provision in many-switch systems. To appear in *Proc. IEEE INFOCOM '99*, 1999.

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

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

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

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

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

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

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

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

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

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

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

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

Note: The Reference section has been shortened due to space constraints.