On the Feasibility of Dynamic Congestion-Based Pricing in Differentiated Services Networks

Nan Jin and Scott Jordan, *Member, IEEE*

***Abstract*—Differentiated services can ensure that traffic on some codepoints receives higher quality of service (QoS) than traffic on other codepoints, but without additional mechanisms it cannot target any particular QoS. Congestion-based pricing has been suggested as a method to target QoS in other network architectures. Here, we investigate whether congestion-based pricing can be used to control aggregate traffic into each codepoint by motivating users to choose the codepoints appropriate for each application. We first ask what information needs to be exchanged; we assert that both price and QoS information must be available for users to make decisions. We then ask how effective congestion-based pricing in diffServ can be; we find that it is feasible only for networks with sufficiently high bandwidth to guarantee that QoS can be quickly measured.**

***Index Terms*— Differentiated services, pricing, quality of service (QoS).**

I. Introduction

DIFFERENTIATED services (diffServ) has been widely discussed in the literature as a potential method to provide a range of qualities of service (QoS) to applications. Some of this literature suggests that diffServ should not only provide a range of QoS, but should also either target particular levels of QoS or should coordinate with the applications to tune the QoS to application requirements. Such QoS targeting or coordination requires control over the aggregate flow rates into each diffServ codepoint.

Pricing has been widely discussed in the literature as a potential method to link application requirements with network resource allocation, connection access control (CAC), and flow and congestion control. Although most of the pricing literature has focussed on Integrated Services (IntServ), it is natural to ask if pricing can be used in diffServ to coordinate QoS levels with application requirements. There are relatively few papers to date that have discussed the use of pricing in diffServ. These papers, however, generally consider prices that change on the order of hours to days or months. Correspondingly they intend prices to control the network in an open loop manner and to reflect long-term average demand.

Manuscript received December 12, 2005; revised November 3, 2006 and April 9, 2007. First published March 12, 2008; current version published October 15, 2008. Approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Orda. This work was supported by the National Science Foundation and by the Defense Advanced Research Projects Agency.

N. Jin was with the Department of Electrical Engineering and Computer Science, University of California, Irvine, CA 92697 USA. She is now with Watchguard Technologies, Inc., Tustin, CA 92780 USA.

S. Jordan is with the Department of Computer Science, University of California, Irvine, CA 92697 USA (e-mail: sjordan@uci.edu).

Digital Object Identifier 10.1109/TNET.2007.908163

In this paper, we ask whether prices can be *dynamically modified on a timescale of seconds to minutes*, and whether such dynamic congestion-based pricing can be effectively used in diffServ to provide closed-loop control over flow rates into each diffServ codepoint. Our principal results suggest that diffServ congestion-based pricing based on availability of both price and QoS information is feasible only for high-bandwidth networks in which QoS can be quickly measured.

IntServ is based on a reservation paradigm. Network resources such as bandwidth and buffer are reserved for each flow or traffic class. If these reservations are properly managed by CAC or pricing mechanisms, then the corresponding requested QoS can be ensured for each flow or class. DiffServ, on the other hand, is based on a generalized priority paradigm. All packets stamped with a single codepoint are treated similarly, but per hop behaviors (PHBs) are used to differentiate the QoS obtained by different codepoints. Without additional mechanisms such as CAC or pricing, diffServ cannot even target any particular QoS for each codepoint; it merely ensures that some codepoints will obtain different QoS than others. We posit here that for many applications it is not sufficient to know that they are obtaining *better QoS than others*; they often require some *minimum QoS* to perform adequately.

An application's QoS at each hop results from two factors: 1) how much traffic is assigned to each codepoint, and 2) how each codepoint is treated by the PHB. The various levels of performance can only be achieved by jointly considering these two factors. However, the diffServ literature has almost exclusively focussed only on the second factor—the mapping from codepoints to PHBs. We believe that the first factor—controlling the aggregate traffic into each codepoint—is critical to targeting particular levels of QoS, and therefore to matching applications with codepoints.

PHBs are implemented through buffer management (dropping) and bandwidth management (scheduling) mechanisms. Many schedulers have been proposed for diffServ, e.g., priority queueing (PQ) and weighted fair queueing (WFQ). PHBs are often compared on the basis of their abilities to provide strongly differentiated qualities of service, which can correspond to high economic efficiency, and on the basis of their abilities to maintain fairness and keep lower priority codepoints from starvation.

However, *without additional control of the aggregate traffic into each codepoint, no PHB can ensure any particular level of QoS to any codepoint*, since the load on each codepoint cannot be bounded. Control over the aggregate traffic in each codepoint at each hop can be provided by any mechanism that limits congestion. Traditional mechanisms for this certainly include CAC and closed loop flow control. In recent years, congestion-based

pricing has been widely suggested as a method to signal congestion, distribute network resources, and control end-to-end service quality. Pricing works by motivating users to make appropriate decisions in a distributed fashion, and thus to instill cooperation.

*It is natural, therefore, to ask if congestion-based pricing can be used in conjunction with diffServ.*The goal is for pricing to control aggregate traffic into each codepoint. The central idea is that if applications know the QoS obtained on each codepoint and know the price charged on each codepoint, they will make intelligent decisions about how to mark their packets and how much traffic to transmit.

In this paper, we attempt to answer this question by designing a congestion-based pricing scheme for diffServ, and analyzing its ability to dynamically control diffServ traffic and provide desired QoS to each codepoint. We are not advocating that this particular congestion-based pricing scheme be used, nor even that any congestion-based pricing be used in diffServ. Our focus is on asking two questions. First, what information might need to be exchanged to design a congestion-based pricing scheme for diffServ, namely how complicated does it have to be? Second, how quickly can it react to fluctuations in network traffic, namely how effective will it be?

These questions have not been addressed in the literature on pricing. Early motivation for pricing of diffServ was provided by Cocchi [1], who argued for service-class pricing that maximizes total network utility; this early model however did not allow applications to choose between multiple codepoints or dynamically modify traffic rates. Most of the literature on pricing in diffServ addresses *static pricing* (i.e., prices changed only on the order of hours to days or months) [2]–[8]. Odlyzko [2] proposed Paris Metro Pricing (PMP), a static pricing scheme that partitions network capacity statically into logically separate channels, and uses prices to provide open loop control over traffic rates. Marbach [8] suggested pricing diffServ codepoints using PMP by representing each codepoint as a separate channel and assuming that customers maximize surplus. Orda [5] considered assignment of traffic classes to diffServ codepoints using pricing. Altmann [7] proposed a framework in which users choose a pair of price and QoS. Courcoubetis [3] brought pricing into the service level agreement (SLA); prices reflect demand for effective bandwidth over months to years. Approaches by Semret [9] and by Fulp [10] use prices changed on the order of hours to address both SLA provisioning and resource allocation to individual users.

None of this literature, however, considers the use of dynamic congestion-based pricing on the order of seconds to minutes to control aggregate diffServ traffic. Correspondingly, none suggests that pricing be used to allow for particular QoS targets or to coordinate application requirements with dynamic QoS levels. We have found only a few papers that address such issues. Gupta [11] considers dynamic pricing to track a socially optimal allocation, but QoS is restricted to delay and a strict priority scheduler is assumed. He [12] uses dynamic pricing, with QoS based on delay, to incentivise users to select codepoints in a manner that maximizes social welfare. Park [13] analyzes equilibria in dynamic multi-class QoS provisioning, but does not explicitly consider pricing.

There is a large body of work on pricing in systems other than diffServ that we draw upon. Pricing has been suggested as a method to signal congestion and distribute the allocation of bandwidth to individual flows or groups of flows. Congestion-based pricing is often used to signal such information in a distributed fashion with a minimal exchange, often using marginal cost pricing (e.g., see [14]–[17]). Utility and pricing have also been used to express the objective of congestion control schemes (e.g., see [18]–[22]). Some pricing algorithms view the price as the result of a game between users or between the users and the network (e.g., see [23] and [24]). In addition, there is literature addressing measurement of QoS (e.g., see [25] and [26]) and construction of utility functions (e.g., see [27] and [28]).

Although this literature details many uses of pricing in differentiated services, *we do not believe any of these models have answered our basic questions about the viability of dynamic congestion-based pricing for diffServ*. Our approach is to consider a network consisting of a set of autonomous systems, together offering a common set of diffServ codepoints. Each autonomous system (AS) is assumed to have control over its own PHBs. The resulting QoS on each codepoint depends on the aggregate traffic marked with that codepoint on each route. Users, representing aggregated traffic of a single application over a route, are modeled by a utility function that describes the application QoS requirements; utility is defined as a function of flow rates and QoS on each codepoint. The goal of the network is to maximize the total utility of all active users (in contrast to maximization of revenue or profit).

To answer the question regarding what information might need to be exchanged to design a congestion-based pricing scheme for diffServ, we illustrate how congestion-based pricing can be distributed among a set of user agents and a set of AS agents. Our approach requires that each AS post prices per packet for each codepoint. It also requires that the QoS obtained on each codepoint be either advertised by each AS or measured by the user or some other network entity. This price and QoS information is used by each user (representing a set of real network users) to choose a traffic rate on each codepoint. Our approach also requires the collection by an AS of additional information about the marginal user utility with respect to QoS.

The setting considered here is one in which users are assumed to freely provide private information about the sensitivity of user utilities to QoS, and both users and autonomous systems are truth-telling, i.e., users do not to attempt to influence prices and autonomous systems accurately report QoS. In this sense, users and the network are assumed to be willing to cooperate with each other through truthful exchange of information. Alternate problems worthy of consideration can be formed by changing any of these assumptions, but they are outside the scope of this paper.

To answer the question regarding how quickly such a scheme can react to fluctuations in network traffic, we assume that the user and network algorithms perform in an iterative fashion, dynamically exchanging price and flow information. We analyze the resulting time scales for convergence of QoS measurement, utility sensitivity estimation, and prices. We find that although simple feedback control policies can result in convergence in a small number of iterations, such congestion-based pricing is

feasible only for networks with sufficiently high rates to guarantee that a large number of measurements are available relatively quickly.

We study a generic priority pricing scheme that does not rely upon any specific network architecture, signalling protocol, or traffic characterization. While this framework does not model all aspects of diffServ, we believe that it captures the key aspects to answer our principal questions. A great deal of additional work would have to be done to make any such approach implementable, including development of signalling protocols, measurement techniques, and scheduling policies.

The paper is organized as follows. In Sections II and III, we propose a pricing framework, with the goal of maximizing total utility of all active users. Distributed pricing roles are assigned to each user and each autonomous system. We show that a traffic allocation is optimal if and only if all users and routers are in equilibrium. In Section IV, we investigate the time scale associated with congestion-based price updates, and show that the relative precisions of prices, traffic rates, and total utility are inversely proportional to the square root of the total number of observations of QoS measures, which in turn is determined by network bandwidth and the time required per iteration. Finally, in Section V, we present a small numerical example to demonstrate convergence rates.

II. DIFFSERV CODEPOINT SELECTION MODEL

In Sections II and III, we investigate the ability of a generic priority pricing method to dynamically control diffServ traffic and provide desired QoS to each codepoint. In this section, we introduce a model in which the network tries to maximize total utility of all users. In the next section, we propose a distributed solution in which each flow and each autonomous system on a user's route exchange information about codepoint availability and QoS, and about traffic rates and the resulting utilities.

The problem formulation consists of a description of both the users and the network. The network description includes identification of the set of diffServ codepoints, the relationship between traffic and QoS on each codepoint, and a routing matrix. The user description includes identification of a utility function that quantifies how each user selects codepoints.

A. Network Description

Consider a network consisting of a number of autonomous systems. We focus only on traffic that is sent via diffServ codepoints along fixed routes. (Routing based on utility maximization is also a topic for research, but is not considered here.) Let a link denote a *route through an AS* from a specified ingress point to a specified egress point (or destination within the AS). Index the set of links in the network by $j = 1, \ldots, J$. Index the diffServ codepoints offered by the network by $k = 1, \ldots, K$. Let the traffic [in bits per second, (bps)] on each link be denoted by a column vector $\vec{\lambda}_j$, where the kth component, λ_{jk}, denotes the traffic on link j and codepoint k. (The notation used in this paper is summarized in Table I.)

Index the set of QoS measures (e.g., delay and loss) by $l = 1, \ldots, L$, and denote the QoS on link j by a $K \times L$ matrix $\vec{\vec{\sigma}}_j$, where $\sigma_{jkl} = \sigma_j(k, l)$ represents the lth QoS measure on link j and codepoint k. We assume that QoS on each link is solely a function of the traffic on that link. Since a link denotes a route through an AS, this formulation ignores the effect of traffic on one (ingress, egress) pair within an AS on the QoS of a different (ingress, egress) pair within the same AS. This assumption is more likely to be reasonable if virtual paths are used *within each AS*. As we discuss later in the paper, use of virtual paths from edge-to-edge (not only within an AS) also simplifies calculation of sensitivity information. If virtual paths are not used, then the impact of cross-traffic should also be considered.

TABLE I
NOTATION

Notations	Descriptions
i, I	Index of the set of users
j, J	Index of the set of links in the network
k, K	Index of the set of diffServ codepoints
l, L	Index of the set of QoS measures
$\vec{\lambda}_j$, λ_{jk}	Traffic rate on link j
$\vec{\Lambda}_i$, Λ_{ik}	Traffic rate of user i's flows
$\vec{\vec{\sigma}}_j$, $\sigma_{jkl} = \sigma_j(k,l)$	QoS measures on link j
$\vec{\vec{q}}_i$, $q_{ikl} = q_i(k,l)$	QoS observed by user i
$U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i)$	User i's utility
φ_{jk}	Cost per unit traffic rate on link j codepoint k
p_{ik}	Cost user i pays per unit rate on codepoint k

B. User Description

We use the term *user* to represent a set of flows along the same route that share a set of codepoints in a consistent manner (likely representing application classes). Index the users by $i = 1, \ldots, I$. Denote the set of routes by a $I \times J$ routing matrix R, where $R_{ij} = 1$ if user i is routed over link j and $R_{ij} = 0$ otherwise. Denote user i's route by $r_i = \{j | R_{ij} = 1\}$. Similarly, denote the set of flows that utilize link j by $\hat{r}_j = \{i | R_{ij} = 1\}$.

Let the traffic (in bps) of user i's flows be denoted by a column vector $\vec{\Lambda}_i$, where the kth component, Λ_{ik}, denotes user i's traffic on codepoint k. Then, assuming losses are small, it follows that the total traffic on link j is simply the sum of the traffic from each user, namely $\vec{\lambda}_j = \sum_{i \in \hat{r}_j} \vec{\Lambda}_i$.

We assume that QoS is additive over links along a route. Denote the QoS observed by user i by the $K \times L$ matrix $\vec{\vec{q}}_i = \sum_{j \in r_i} \vec{\vec{\sigma}}_j$.

We model each user's selection of codepoints using a utility function that describes the user's satisfaction with their flows and the resulting QoS. Denote user i's utility as $U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i)$. (An example of such a utility function is given at the beginning of Section V.) Now the QoS observed by user i, $\vec{\vec{q}}_i$, is a function of the traffic on each link on that user's route, $\{\vec{\lambda}_j | \sum_{j \in r_i}\}$, which is in turn a function of the user flows, $\{\vec{\Lambda}_j\}$. Denote the set of all user flows by $\vec{\Lambda} = [\vec{\Lambda}_1^t, \ldots, \vec{\Lambda}_I^t]^t$. User i's utility can thus be represented as a function of the set of user flows, $V_i(\vec{\Lambda}) = U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i)$.

We assume that user utility V_i is a twice differential concave function with bounded Hessian. This is a significant assumption. A change in user i's traffic, $\vec{\Lambda}_i$, has two effects: a direct effect upon user i's utility through the direct dependence of U_i upon $\vec{\Lambda}_i$, and an indirect effect upon the utility of all users (including i) who share links with user i through the effect upon the QoS, $\vec{\vec{\sigma}}_j$, of each link j on user i's route. (These two effects are considered in more detail in Section III.) While the direct

effect of $\vec{\Lambda}_i$ upon U_i may be represented analytically, the effect of $\vec{\Lambda}_i$ upon $\vec{\sigma}_j$ is likely to be measured but unavailable in closed form. Concavity of V_i is therefore a complex issue that must be addressed with detailed knowledge or observation of the particular diffServ schedulers used.

C. Optimization Problem and Pricing

We consider the optimization of the total utility of all users in the network. This problem can be posed as a maximization of total user utility over flow rates of each user on each codepoint

$$\max_{\{\vec{\Lambda}_i\}} \sum_{i=1}^{I} U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i) \quad \text{s.t. } \vec{\Lambda}_i \geq 0, \ \forall i \tag{1}$$

where the inequality is taken componentwise.

Theorem 1: Any local maximum of optimization problem (1) is also a global maximum. The set $\arg\max\{\sum_{i=1}^{I} U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i) \mid \vec{\Lambda}_i \geq 0\}$ is either empty or convex.

Proof: Consider the equivalent problem

$$\max_{\vec{\Lambda}} \sum_{i=1}^{I} V_i(\vec{\Lambda}) \quad \text{s.t. } \vec{\Lambda} \geq 0. \tag{2}$$

This equivalent problem is a concave program, by definition, if the feasible region is a convex set and the optimization metric is a concave function. The feasible set in (2) is a convex set. By assumption, $V_i(\vec{\Lambda})$ is a concave function. Therefore the metric is a concave function, since the sum of concave functions is concave. For a concave program, the conclusion of the theorem follows [29, Th. 7.13]. ∎

Theorem 1 tells us that if any algorithm finds a local maximum of the optimization problem, then it must also be a global maximum. The set of optimal rates on each codepoint for each user forms a convex set. Typically, the set is not empty, since both zero and infinite rates usually correspond to zero utility, and intermediate rates typically generate positive utility.

This problem, as posed, is a centralized nonlinear optimization in which all user utility functions, all traffic rates, and all codepoint QoSs must be known by a single entity. Solving such a maximization problem for a network of moderate size can be computationally intensive, and thus pricing is often used to distribute optimization problems into separate roles for each entity, with minimal communication between them. In the next section, we outline the proposed pricing framework.

III. Distributed Solution

To answer the question regarding what information might need to be exchanged to design a distributed congestion-based pricing scheme for diffServ, it is natural to think of distributing the utility optimization between the user agents and the autonomous system agents so that the autonomous systems set the prices for each codepoint while the users determine their traffic rates on each codepoint given a set of prices.

Consider the first-order conditions required for the solution to the optimization problem given in (1)

$$\frac{d\left(\sum_{i'=1}^{I} U_{i'}\right)}{d\Lambda_{ik}} = 0. \tag{3}$$

Now

$$\frac{d\left(\sum_{i'=1}^{I} U_{i'}\right)}{d\Lambda_{ik}} = \frac{dU_i}{d\Lambda_{ik}} + \sum_{i' \neq i} \frac{dU_{i'}}{d\Lambda_{ik}}. \tag{4}$$

The first term on the right-hand side of (4) is the marginal utility of user i with respect to the flow rate on codepoint k. The second term is the total marginal utility of all other users impacted by a change in the flow rate of user i on codepoint k. These two terms combined can be rewritten as

$$\frac{\partial U_i}{\partial \Lambda_{ik}} + \sum_{j \in r_i} \sum_{k'=1}^{K} \sum_{l=1}^{L} \sum_{i' \in \hat{r}_j} \left(\frac{\partial U_{i'}}{\partial q_{i'k'l}} \frac{\partial \sigma_{jk'l}}{\partial \Lambda_{ik}} \right). \tag{5}$$

As discussed above, there are two effects upon utility from a change in user i's traffic, $\vec{\Lambda}_i$: a direct effect upon user i's utility through the direct dependence of U_i upon $\vec{\Lambda}_i$, and an indirect effect upon the utility of all users (including i) who share links with user i through the effect upon the QoS, $\vec{\vec{\sigma}}_j$, of each link j on user i's route. The direct effect is represented by the first term in (5), and the indirect effect is represented by the summation.

Since we assume that losses are small, any infinitesimal change in a user flow Λ_{ik} produces the same infinitesimal change in the link flow λ_{jk}, for any link on the user's route. (If losses were not small, downstream rates may see significantly different changes than upstream rates due to changes in losses on each link.) As a consequence, the sensitivity of QoS to both flow rates are the same, namely $\frac{\partial \sigma_{jk'l}}{\partial \Lambda_{ik}} = \frac{\partial \sigma_{jk'l}}{\partial \lambda_{jk}}$. It follows that expression (5) can be rewritten as

$$\frac{\partial U_i}{\partial \Lambda_{ik}} + \sum_{j \in r_i} \sum_{i' \in \hat{r}_j} \frac{dU_{i'}}{d\lambda_{jk}} \tag{6}$$

where

$$\frac{dU_{i'}}{d\lambda_{jk}} = \sum_{k'=1}^{K} \sum_{l=1}^{L} \frac{\partial U_{i'}}{\partial q_{i'k'l}} \frac{\partial \sigma_{jk'l}}{\partial \lambda_{jk}}. \tag{7}$$

The first term in (6) is now the marginal utility of user i with respect to the flow rate on codepoint k, keeping all QoSs constant. The term $\partial \sigma_{jk'l} / \partial \lambda_{jk}$ in (7) is the change in QoS measure l of codepoint k' on link j due to a change in traffic on codepoint k on link j. (For convenience, collect these terms for link j into an array called $D_{\vec{\lambda}_j} \vec{\vec{\sigma}}_j$.) The term $\partial U_{i'} / \partial q_{i'k'l}$ is the change in utility of user i' due to a change in user i''s QoS measure l of codepoint k'. (For convenience, collect these terms for user i' into an matrix called $\nabla_{\vec{\vec{q}}_{i'}} U_{i'}$.) $dU_{i'} / d\lambda_{jk}$ thus gives the total effect upon user i''s utility due to a change in the flow rate of codepoint k on link j. Summing these terms over all $i' \in \hat{r}_j$ similarly gives the total effect upon all users utilities due to a change in the flow rate of codepoint k on link j. Finally, summing these terms over all $j \in r_i$ gives the total marginal utility of all users due to a change in the flow rate of user i on codepoint k.

Suppose that link j charges an amount per unit traffic rate on codepoint k equal to

$$\varphi_{jk} = -\sum_{i' \in \hat{r}_j} \frac{dU_{i'}}{d\lambda_{jk}}. \tag{8}$$

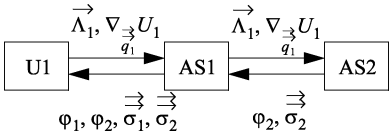


Fig. 1. Distributed solution.

(For convenience, collect these prices into a column vector $\vec{\varphi}_j$.) Suppose that each user pays the sum of all link prices along its route

$$p_{ik} = \sum_{j \in r_i} \varphi_{jk}. \tag{9}$$

(For convenience, collect these prices into a column vector $\vec{p}_i$.) It follows that the first-order conditions in (3) can be written as

$$\frac{\partial U_i}{\partial \Lambda_{ik}} = p_{ik}. \tag{10}$$

Suppose the user attempts to maximize its *surplus*, defined as utility minus cost, by choosing its rate (under the assumption that QoS is fixed)

$$\max_{\vec{\Lambda}_i} \left\{ U_i(\vec{\Lambda}_i, \vec{\vec{q}}_i) - \vec{\Lambda}_i^t \vec{p}_i \right\}. \tag{11}$$

At maximum surplus, marginal utility with respect to flow rates must equal the prices, namely (10). It follows that there exists a set of prices, $\vec{\varphi}_j$, on each link such that the first-order optimization conditions are satisfied if each user maximizes surplus.

This result suggests a distributed optimization process as illustrated in Fig. 1. Each (ingress, egress) pair in each AS advertises the prices $\vec{\varphi}_j$ and QoSs $\vec{\vec{\sigma}}_j$ for each codepoint it offers. Each user sums the prices on each link on its route to obtain the total prices $\vec{p}_i$, and similarly sums these QoSs to obtain the total QoS $\vec{\vec{q}}_i$. Each user then determines the flow rates on each codepoint in order to maximize surplus, following (11). In addition, each user advertises its marginal utility with respect to QoS on each codepoint $\nabla_{\vec{\vec{q}}_i} U_i$. Each link estimates the sensitivity of QoS on each codepoint with respect to the total flow on each codepoint $D_{\vec{\lambda}_j} \vec{\vec{\sigma}}_j$. It then sets the prices $\vec{\varphi}_j$ using (8).

The cooperation is achieved through the information exchanged. Prices signal the aggregate marginal utility with respect to QoS on each codepoint, and user gradients communicate similar user utility sensitivities. Alternate methods that attempt to achieve optimal allocation of resources without such explicit communication are not considered here.

Scalability with respect to the number of users is a potential issue here, since the complexity depends on the number of users passing through a link.[1] It is desirable to aggregate any pieces of information that are associated with individual flows, namely any items with a user i subscript. Rather than storing the flow rates and sensitivities of each individual user, an AS could simply calculate and store the aggregate flow rates and sensitivities by summing corresponding aggregates from each upstream AS. Disaggregation of flow rates into each downstream AS can be directly estimated. A simple approach to disaggregation of sensitivities is to estimate each downstream sensitivity based on the traffic split. This reduces the complexity to the order of the number of codepoints on a link. However, this approach to disaggregation of sensitivities may be inaccurate if users on each route do not have similar utility functions. If virtual paths are used edge-to-edge, then disaggregation of sensitivities is simple, since each outgoing link would contain different VPs. This reduces to complexity to the order of the number of combinations of virtual paths and codepoints on a link. Use of edge-to-edge VPs would also allow for aggregation on the reverse path of link prices $\vec{\varphi}_j$ and QoSs $\vec{\vec{\sigma}}_j$. As the VP passes through each AS, it could keep a running total of link prices and QoSs, so that the user receives only the sums $\vec{p}_i$ and $\vec{\vec{q}}_i$. However, either type of aggregation and disaggregation is likely to produce smoothing effects on all of these quantities, and these effects require further study.

To summarize the above discussion, we have the following theorem.

Theorem 2: The user traffic $\{\vec{\Lambda}_i\}$ solves optimization problem (1) if and only if it is an equilibrium of user algorithm (11) for each user i and network algorithm (8) for each autonomous system j.

Proof: The equivalent problem (2) is a concave program. It follows from the Kuhn–Tucker theorem under convexity [29, Th. 7.16] that a set of user flows $\{\vec{\Lambda}_i\}$ solves the problem if and only if there exist a set, $\{\mu_{ik}\}$, of nonnegative constraint shadow costs such that $\mu_{ik}\Lambda_{ik} = 0, \forall i, k$ and

$$\frac{d \sum_i V_i(\vec{\Lambda})}{d\Lambda_{ik}} + \mu_{ik} = 0, \ \forall i, k. \tag{12}$$

For any strictly positive flow Λ_{ik}, (12) implies that the shadow costs $\{\mu_{ik}\}$ must be zero. If prices are set using (8) and (9), then it follows from (4) and (6) that the first term in (12) is simply $\frac{\partial U_i}{\partial \Lambda_{ik}} - p_{ik}$. Consequently, (12) is satisfied if any only if (10) holds. The theorem follows. ■

Theorem 2 establishes that if all users act as individual optimizers (but do not try to game the system) and if the network sets prices according to marginal utilities, then the system optimum can be cooperatively achieved in a distributed fashion. As discussed above, alternate problems models could be considered in which users are noncooperative, non-truth-telling, or use game theory to increase their surplus further.

When flows are positive, the user algorithm (11) is an unconstrained optimization problem. Due to the cooperation imposed by the exchange of information, this can be solved using any appropriate ascent direction computational method, e.g., a gradient method or Newton's method. In our results below, user traffic is updated iteratively using a gradient method as follows:

$$\vec{\Lambda}_i^{k+1} = \vec{\Lambda}_i^k + s^k \nabla U_i(\vec{\Lambda}_i^k, \vec{\vec{q}}_i^k) \tag{13}$$

where s^k is a positive step size, and $\nabla U_i(\vec{\Lambda}_i^k, \vec{\vec{q}}_i^k) = [\partial U_i/\partial \Lambda_{i1} - p_{i1}^k; \ldots; \partial U_i/\partial \Lambda_{iK} - p_{iK}^k]$.

In each iteration, after each user advertises its new rates and its marginal utility with respect to QoS on each codepoint, each autonomous system updates its prices using (8). Since this use of marginal cost pricing accurately reflects externalities, such gradient methods guarantee convergence of rates and prices. (If exchange of marginal cost pricing was absent, gradient methods do not necessarily guarantee convergence [30].)

[1]However, a *user* can represent a set of flows along the same route that share a set of codepoints in a consistent manner, probably on the basis of application classes.

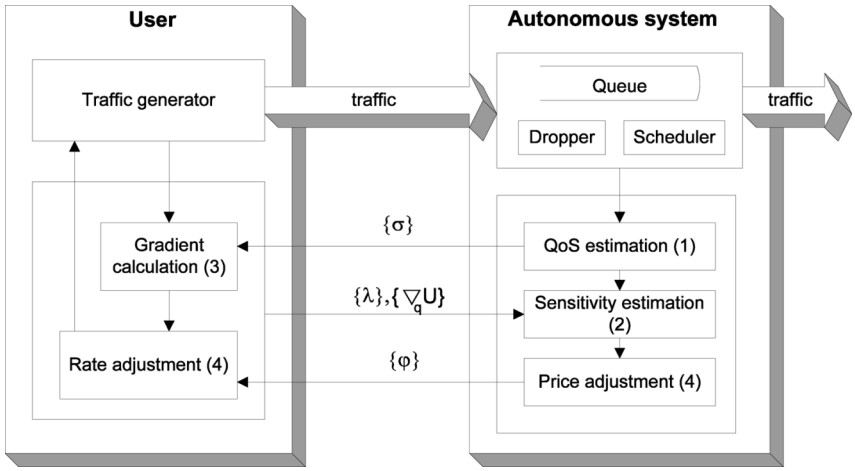


Fig. 2. Information exchange in each iteration.

Theorem 3: Let user traffic rates $\{\vec{\Lambda}_i^k\}$ be a sequence generated by (13), and prices $\{\varphi_j^k\}$ a sequence generated by (8). Then there exists a positive step size $s^k = s > 0$ such that the limit point of $\{\vec{\Lambda}_i^k\}$ is stationary and solves optimization problem (1).

Proof: Denote the objective function by $f(\vec{\Lambda}) = \sum_{i=1}^{I} V_i(\vec{\Lambda})$, and denote the combination of the rate vectors by $x^k = \vec{\Lambda}^k$. Then

$$\nabla f(x^k) = \left[\frac{\partial U_i}{\partial \Lambda_{ik}} + \sum_{j \in r_i} \sum_{i' \in \hat{r}_j} \frac{dU_{i'}}{d\lambda_{jk}}\right] = \left[\frac{dU_i}{d\Lambda_{ik}} - p_{ik}\right].$$

It follows that the update $x^{k+1} = x^k + s^k \nabla f(x^k)$ is equivalent to the synchronous set of updates given by (13). By assumption $V_i(\vec{\Lambda})$ is twice differentiable with bounded Hessian. As a consequence, ∇f is Lipschitz continuous. Therefore, every limit point of $\{x^k\}$ is a stationary point of f [31, Prop. 1.2.3]. By Theorem 2, any such stationary point solves optimization problem (1). ■

Our first principal question posed above asked what information might need to be exchanged to design a congestion-based pricing scheme for diffServ, namely how complicated does it have to be? Key requirements of our approach are: 1) the scheme be distributed among a set of user agents and a set of AS agents; 2) each AS post prices per packet for each codepoint; 3) the QoS obtained on each codepoint be either advertised by each AS or measured by the user or some other network entity; 4) users be aggregated in some efficient manner into groups that can be represented by a manageable number of user agents; 5) user agents use price and QoS information to adaptively choose traffic rates on each codepoint; and 6) the sensitivity of users with respect to QoS either be advertised by user agents or measured by some other network entity.

With this information exchange, Theorems 2 and 3 tell us that, under certain conditions, the pricing process proposed above is guaranteed to converge to an equilibrium among all users and autonomous systems that maximizes the total utility. This finding demonstrates the ability of pricing to dynamically control diffServ traffic and provide desired QoSs to diffServ codepoints.

Our second question, however, asks how quickly such a process can react to fluctuations in network traffic. In the next section, we explore this issue by analyzing how quickly prices should be updated to achieve statistical stability.

IV. Time Scales

Our goal in this section is to understand how quickly a congestion-based pricing scheme can react to fluctuations in network traffic. To be specific, we want to understand the relationship between convergence time, measurement accuracy, and system parameters such as bandwidth and round-trip time.

The algorithms introduced in the previous section adopt an iterative approach, as shown in Fig. 2. In each iteration: 1) the autonomous systems measure QoS; 2) the autonomous systems estimate the sensitivities of QoS with respect to traffic rates; 3) the users compute the gradients of utilities with respect to QoS; and 4) the autonomous systems compute the prices and the users update their rates.

When a gradient ascent method is used to maximize a deterministic function, the rate of convergence depends on the step size and on the shape of the function. In congestion-based pricing, however, the function itself is random. This randomness is caused by two factors. First, measurement of QoS and estimation of sensitivities induce measurement error, which propagates through each step of each iteration. Second, users arrive and depart from the system, causing a change in the number of terms in the summation of utility.

These two sources of randomness collude to limit the effectiveness of any congestion-based pricing scheme. Measurement error can be reduced by collecting more data per iteration. This data collection, however, takes time and therefore slows down the rate of convergence. In addition, each iteration requires an exchange of information between users and autonomous systems, and therefore additional iterations also require additional time. The result is a tradeoff between convergence time and measurement accuracy (and therefore the size of the resulting fluctuations). Since congestion-based pricing can only be effective if prices and rates can adapt quickly enough to keep up with arrivals and departures of users, this limits congestion-based pricing to networks in which this is possible.

In the next subsection, we briefly review standard techniques for determination of confidence intervals for correlated time series measurements. We then trace the series of steps (1)–(4) above in each iteration. In Section IV-B, we analyze the relation between the relative precision of QoS measures and the total number of observations, which in turn is a function of network bandwidth and the time required per iteration. In Section IV-C, we analyze the effect of measurement errors upon estimates of the sensitivities of QoS with respect to traffic rates. In Section IV-D, we analyze the effects of these errors upon the corresponding sensitivity of utility with respect to QoS. In Section IV-E, we carry through the accumulated effect of these errors upon prices, rates, and utilities. Finally, in Section IV-F, we relate these confidence intervals to basic system parameters such as throughput, round-trip time, and number of autonomous systems. These results together will establish that although simple feedback control policies can result in convergence in a small number of iterations, such congestion-based pricing is feasible only for networks with sufficiently high rates to guarantee that a large number of measurements are available relatively quickly.

A. Brief Review of Confidence Intervals in Correlated Time Series

Suppose we are interested in a performance measure X. Typically, we estimate its mean value, based on observations $x_1, \ldots, x_n$, using an average $\bar{X}(n) = \sum_{i=1}^{n} x_i/n$ of the observations. If the observations are independent and identically distributed (which in our system they are not!), as the number of observations n tends toward infinity, the Law of Large Numbers guarantees that the estimate $\bar{X}(n)$ converges (in multiple senses) toward the unknown true mean. Furthermore, the Central Limit Theorem states that distribution of the error between the estimate and the true mean approaches a Normal distribution with a standard deviation inversely proportional to $\sqrt{n}$.

These results can be used to estimate the confidence interval of the estimate as follows. First, the unbiased estimate of the variance of $\bar{X}(n)$ is calculated from the observations using

$$\hat{\sigma}_n^2 = \sum_{i=1}^{n} \frac{[x_i - \bar{X}(n)]^2}{n(n-1)}. \tag{14}$$

A confidence level $1-\alpha$ is chosen, and the upper critical point is determined by $z = \Phi^{-1}(1-\alpha/2)$.[2] (In numerical results below, we use a 95% confidence level, i.e., $\alpha = 0.05$ and $z = 1.96$.) The half-width of the confidence interval is then given by $\Delta_x = z\hat{\sigma}_n$. The $1-\alpha$ confidence interval for $\bar{X}(n)$ is given by $[\bar{X}(n) - \Delta_x, \bar{X}(n) + \Delta_x]$. The *relative* precision of the estimate is given by the half-width of the confidence interval normalized by the estimate itself, $\epsilon = \Delta_x/\bar{X}(n)$. As a result, the relative precision is inversely proportional to $\sqrt{n}$, i.e., $\epsilon \sim 1/\sqrt{n}$ [32].

However observations are often not independent. Indeed in networks we expect observations to be highly correlated. With correlated observations, the same procedure is used, except that a new method must be used in place of (14) to create an unbiased estimate of the variance of $\bar{X}(n)$. A typical such method is the Method of Batch Means. This method divides the n original observations $x_1, \ldots, x_n$ into b nonoverlapping batches $(x_{1,1}, \ldots, x_{1,m}), \ldots, (x_{b,1}, \ldots, x_{b,m})$ of size m (where $n = bm$). The batch means $(\bar{X}_1(m), \ldots, \bar{X}_b(m))$ are then treated as an approximately i.i.d. sequence of data, based on the assumption that observations that are separated in time are less correlated, as will typically be true in networks. The estimate of the variance of $\bar{X}(n)$ is calculated from the observations using

$$\hat{\sigma}_n^2 = \sum_{i=1}^{b} \frac{[\bar{X}_i(m) - \bar{X}(n)]^2}{b(b-1)}$$

and the confidence interval is then calculated as before. Many papers have suggested methods for determining the batch size m and number of batches b [33]; we choose the Fixed Number of Batches rule due to its simplicity.

We start our analysis in the next subsection by expressing the relationship between the the number of measurements n of QoS collected in each iteration and the resulting relative precision of the QoS estimates.

B. Measurement of QoS

Consider a sequence of n_{jk} measurements of σ_{jkl}, the lth QoS measure on link j and codepoint k. The sequence forms a weakly stationary processes, and $\lim_{n_{jk}\to\infty} n_{jk}\hat{\sigma}^2_{n_{jk}}$ exists and is finite. It follows that the relative precision of the estimate is inversely proportional to the square root of the number of batches, and hence to the number of observations

$$\epsilon[\sigma_{jkl}] \sim \frac{1}{\sqrt{n_{jk}}}.$$

The rate at which $\epsilon[\sigma_{jkl}]$ decreases depends on the amount of correlation between the QoS measures experienced by consecutive packets. To illustrate this, consider a single M/M/1/N queue operating under a dynamic weight scheduler (described in more detail in the next section). This scheduler supports four codepoints denoted 00, 01, 10, and 11. Two QoS measures—delay and loss—are measured. We consider the relative precision for the estimator of delay of the lowest priority codepoint, 00.[3] In Fig. 3, we plot the relative precision as a function of the number of measurements of delay of 00 packets. The relative precision of QoS measure σ_{jkl} follows $\epsilon[\sigma_{jkl}] \approx 4.3/\sqrt{n_{jk}}$ where n_{jk} is the number of measurements of delay of 00 packets. If codepoint 00 packet delays were independent, then the relative precision would have been $\epsilon[\sigma_{jkl}] \approx 1.6/\sqrt{n_{jk}}$. Therefore, in this example it requires approximately seven correlated codepoint 00 measurements to get the equivalent of one independent measurement. The relative precisions for the estimators of delay on other codepoints and for the estimators of losses follow similar $1/\sqrt{n}$ patterns.

The time required to collect a sufficient number of QoS measurements, therefore, depends on the throughput on each codepoint. During a time T, we can collect QoS measurements from approximately $\lambda_{jk}T$ codepoint k packets. It follows that the

[2] Φ denotes the C.D.F. of the standard Normal distribution.

[3] The parameters used are: $\rho = 1.212$, buffer size $= 11$ packets, packet size $= 4000$ bits, 30% of packets each are marked as 00 and 10, and 20% each as 01 and 11.

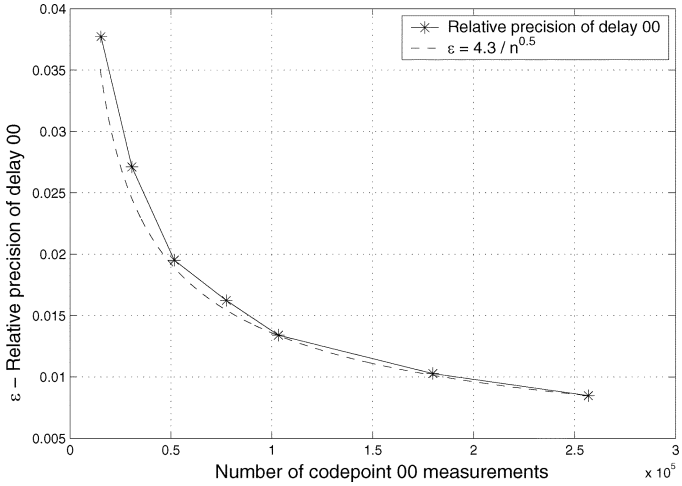


Fig. 3. Relative precision of delay of codepoint 00 versus number of measurements.

relative precision $\epsilon[\sigma_{jkl}] \approx 4.3/\sqrt{\lambda_{jk}T}$. Note that the time required to achieve a specified level of relative precision is inversely proportional to the rate on that codepoint. This relationship places a fundamental limit on the efficacy of congestion-based pricing. For example, a relative precision of 1% in the measurement of codepoint 00 delay can be obtained with approximately 2×10^5 measurements. If the transmission rate of the queue described above is 10 Gbps, then collection of these measurements takes only 0.4 s; however, if the transmission rate is only 10 Mbps, then collection requires 400 s, which is likely to render congestion-based pricing ineffective.

C. Measurements of Sensitivities of QoS

In this subsection, we analyze the effect of measurement errors upon estimates of the sensitivities of QoS with respect to traffic rates. These sensitivities $\partial\sigma_{jk'l}/\partial\lambda_{jk}$, which form a $K \times K$ matrix, are required to set the prices in (8). They should be computed or measured by each autonomous system once each iteration, in conjunction with the measurement of QoS on each codepoint.

We adopt an approach based on sample path observation and perturbation. For simpler queues, there are closed-form expressions that relate sensitivity to perturbations in flow rates. For diffServ schedulers, however, there are no such closed-form expressions. However, such perturbations can be estimated by direct observation. The idea is to observe the packet flows through the autonomous system, track the variation in sample paths that would occur if small changes were made to the rates, and transform these variations into small changes in the resulting QoS. We construct a virtual queue for each codepoint at each router within the autonomous system. A virtual queue does not actually store or transmit packets; it merely estimates the perturbations in QoS from small changes in the rate on its codepoint. Specifically, virtual queue k within autonomous system j considers the effect of a small increase in rate λ_{jk} to $\lambda_{jk}(1+PR)$, where $0 < PR \ll 1$ is the *perturbation ratio*. The virtual queue tracks the arrival times of each packet in the queue, makes corresponding increases to the packet size, and calculates the corresponding perturbed qualities of service on codepoint k, denoted by σ^+_{jkl}. The virtual queues k' thus estimate the resulting QoS sensitivity of $\sigma^+_{jk'l}$ with respect to rate of codepoint k by

$$\frac{\partial\sigma_{jk'l}}{\partial\lambda_{jk}} \approx \frac{\sigma^+_{jk'l} - \sigma_{jk'l}}{\lambda_{jk}PR}. \tag{15}$$

Suppose the measurement of QoS $\sigma_{jk'l}$ has a confidence interval $(E[\sigma_{jk'l}] - \Delta\sigma_{jk'l}, E[\sigma_{jk'l}] + \Delta\sigma_{jk'l})$, where $\Delta\sigma_{jk'l} = E[\sigma_{jk'l}]\epsilon(\sigma_{jk'l})$. The confidence interval of the QoS sensitivity depends on the correlation between $\sigma_{jk'l}$ and $\sigma^+_{jk'l}$. An upper bound can be found by assuming these are independent. The confidence interval of the QoS sensitivity is then given by

$$\Delta\left[\frac{\partial\sigma_{jk'l}}{\partial\lambda_{jk}}\right] = \frac{\sqrt{\Delta^2[\sigma_{jk'l}] + \Delta^2[\sigma^+_{jk'l}]}}{\lambda_{jk}PR} \approx \frac{\sqrt{2}\cdot\Delta[\sigma_{jk'l}]}{\lambda_{jk}PR}.$$

The corresponding relative precision is

$$\epsilon\left[\frac{\partial\sigma_{jk'l}}{\partial\lambda_{jk}}\right] = \frac{\sqrt{2}\Delta[\sigma_{jk'l}]/(\lambda_{jk}PR)}{\partial\sigma_{jk'l}/\partial\lambda_{jk}}. \tag{16}$$

Define

$$\rho[\sigma_{jk'l}, \lambda_{jk}] \equiv \frac{\partial\sigma_{jk'l}/\partial\lambda_{jk}}{\sigma_{jk'l}/\lambda_{jk}}.$$

This term, called the *rate elasticity of QoS*, is a measure of the increment in QoS per increment in rate, relative to the ratio of the two. The rate elasticity is the slope in direction k of the QoS versus rate surface divided by the average QoS per unit rate, and is related to the curvature of the QoS surface. If the QoS is a convex function of rate, then $0 < \rho[\sigma_{jk'l}, \lambda_{jk}] < 1$. Typically, the rate elasticity will not be very small at operating points. Equation (16) can be thus written as

$$\epsilon\left[\frac{\partial\sigma_{jk'l}}{\partial\lambda_{jk}}\right] = \frac{\sqrt{2}\epsilon[\sigma_{jk'l}]}{\rho[\sigma_{jk'l}, \lambda_{jk}]PR}. \tag{17}$$

Equation (17) tells us that the relative precision of sensitivity of a QoS measure is proportional to the relative precision of the corresponding QoS measure. The proportionality factor is inversely proportional to the perturbation ratio and to the rate elasticity of QoS. Since both of these terms are likely to be less than 1, the relative precision of QoS sensitivity is likely to be greater than the relative precision of the corresponding QoS measure.

Since the perturbation is one-sided, the estimate of the sensitivity in (15) is biased, i.e., the average estimate does not equal the true value. As a consequence, there is a tradeoff in the selection of the perturbation ratio PR. While larger values of PR results in smaller relative precision of sensitivities, this also results in increased error in the estimate itself. This bias leads to error in prices, and thus to a suboptimal resource allocation. There is therefore a tradeoff between the speed of convergence and the efficiency of the allocation.

To illustrate the relationship between the relative precision of sensitivity of QoS and the relative precision of the corresponding QoS measure, Fig. 4 shows the relative precision of sensitivity of delay of codepoint 00 with respect to the rate to codepoint 00 versus the number of measurements per iteration, using a perturbation ratio $PR = 0.1$, for the queue considered in the previous subsection. Compared with Fig. 3, we can see that the relative precision of QoS sensitivity is much larger than

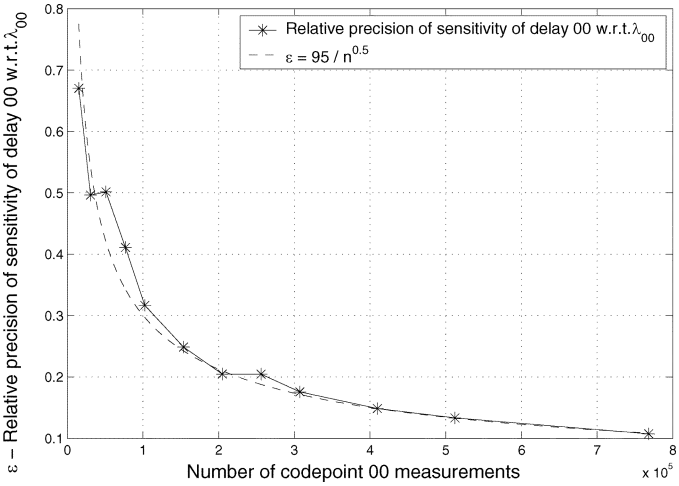


Fig. 4. Relative precision of sensitivity of delay of codepoint 00 with respect to λ_{00} versus number of measurements.

the relative precision of the corresponding QoS measure. For instance, at 2×10^5 measurements per iteration, the rate elasticity $\rho[\sigma_{jk'l}, \lambda_{jk}] \approx 0.52$ and $\epsilon\left[\frac{\partial \sigma_{jk'l}}{\partial \lambda_{jk}}\right] \approx 20\epsilon[\sigma_{jk'l}]$, where k is codepoint 00. This increase in the magnitude of the relative precision represents the cost of this measurement-based approach to estimating the QoS sensitivities.

D. Measurements of Sensitivities of Utility

In this subsection, we analyze the effect of errors in QoS measurement upon the sensitivity of utility with respect to QoS. First, we consider the effect of the confidence interval on QoS upon the confidence interval of utility sensitivity. Consider measurement of $\frac{\partial U_i}{\partial q_{ikl}}(\vec{q})$. Now the QoS $\vec{q}$ will have measurement error; suppose the measured value is $\vec{q}+\Delta\vec{q}$. Then the measurement of the sensitivity will be

$$\frac{\partial U_i}{\partial q_{ikl}}(\vec{q}+\Delta\vec{q}) = \frac{\partial U_i}{\partial q_{ikl}}(\vec{q}) + \sum_{k',l'} \frac{\partial}{\partial q_{ik'l'}}\left(\frac{\partial U_i}{\partial q_{ikl}}\right)\Delta q_{ik'l'}.$$

To get an upper bound, we assume that $\{q_{ik'l'}\}$ forms an independent set of random variables. It follows that

$$var\left(\frac{\partial U_i}{\partial q_{ikl}}\right) = \sum_{k',l'} \frac{\partial^2 U_i}{\partial q_{ikl}\partial q_{ik'l'}} var\left(\Delta q_{ik'l'}\right)$$

and thus that

$$\Delta\left[\frac{\partial U_i}{\partial q_{ikl}}\right] = \sqrt{\sum_{k',l'}\left(\frac{\partial^2 U_i}{\partial q_{ikl}\partial q_{ik'l'}}\Delta[q_{ik'l'}]\right)^2}.$$

In order to relate this to the relative precisions calculated above, we express the route QoS in terms of individual link QoS: $q_{ikl} = \sum_{j\in r_i}\sigma_{jkl}$. We assume that link qualities of service are independent to estimate $\Delta q_{ikl} = \sqrt{\sum_{j\in r_i}\Delta^2[\sigma_{jkl}]}$. The corresponding route QoS thus has a relative precision of

$$\epsilon[q_{ikl}] = \frac{\Delta[q_{ikl}]}{q_{ikl}} = \frac{\sqrt{\sum_{j\in r_i}\sigma_{jkl}^2\cdot\epsilon^2[\sigma_{jkl}]}}{\sum_{j\in r_i}\sigma_{jkl}}. \tag{18}$$

Consequently the confidence interval for the utility sensitivity can be written as

$$\Delta\left[\frac{\partial U_i}{\partial q_{ikl}}\right] = \sqrt{\sum_{k',l'}\left(\left(\frac{\partial^2 U_i}{\partial q_{ikl}\partial q_{ik'l'}}\right)^2\sum_{j\in r_i}\Delta^2[\sigma_{ik'l'}]\right)}.$$

Finally, we can write a corresponding expression for the relative precision

$$\epsilon\left[\frac{\partial U_i}{\partial q_{ikl}}\right] = \sqrt{\frac{\sum_{k',l'}\left(\left(\frac{\partial^2 U_i}{\partial q_{ikl}\partial q_{ik'l'}}\right)^2\sum_{j\in r_i}\Delta^2[\sigma_{ik'l'}]\right)}{\left(\frac{\partial U_i}{\partial q_{ikl}}\right)^2}}.$$

Define

$$\rho\left[\frac{\partial U_i}{\partial q_{ikl}}, q_{ik'l'}\right] \equiv \frac{\frac{\partial}{\partial q_{ik'l'}}\left(\frac{\partial U_i}{\partial q_{ikl}}\right)}{\frac{\partial U_i}{\partial q_{ikl}}/q_{ik'l'}}.$$

This term describes how the gradient of user utility to QoS measure l of codepoint k responds to a change in QoS measure l' of codepoint k', and is determined by the shape of the utility function. Then the expression for the relative precision reduces to

$$\epsilon\left[\frac{\partial U_i}{\partial q_{ikl}}\right] = \sqrt{\sum_{k',l'}\rho^2\left[\frac{\partial U_i}{\partial q_{ikl}}, q_{ik'l'}\right]\epsilon^2[q_{ik'l'}]}. \tag{19}$$

The result establishes a relation between the relative precision of the gradients of user utilities with respect to QoS and the relative precision of the QoS.

E. Effect on Prices, Rates, and Utilities

In this subsection, we analyze the accumulated effect of measurement errors on QoS, QoS sensitivity, and utility sensitivity upon prices, rates, and utilities. The price that user i experiences on codepoint k is given by (9), which is a function of both utility sensitivities and QoS sensitivities. A conservative estimate of the confidence interval can be written as

$$\Delta[p_{ik}] = \sqrt{\sum_S\left(\frac{\partial\sigma_{jk'l}}{\partial\Lambda_{ik}}\right)^2\Delta^2\frac{\partial U_{i'}}{\partial q_{i'k'l}} + \left(\frac{\partial U_{i'}}{\partial q_{i'k'l}}\right)^2\Delta^2\frac{\partial\sigma_{jk'l}}{\partial\Lambda_{ik}}}$$

where $S = \{j\in r_i, k', l, i'\in\hat{r}_j\}$. Using (17) and (19), we can establish the relation between the relative precision of prices and the relative precision of QoS measures, in the following form:

$$\epsilon[p_{ik}] = \frac{\sqrt{\sum_{j\in r_i,k',l,i'\in\hat{r}_j}\left(\frac{\partial\sigma_{jk'l}}{\partial\Lambda_{ik}}\frac{\partial U_{i'}}{\partial q_{i'k'l}}\right)^2 f\left[\{\epsilon^2(\sigma_{j'k''l'})\}\right]}}{\left[\sum_{j\in r_i,k',l,i'\in\hat{r}_j}\left(\frac{\partial\sigma_{jk'l}}{\partial\Lambda_{ik}}\frac{\partial U_{i'}}{\partial q_{i'k'l}}\right)\right]} \tag{20}$$

where

$$f\left[\{\epsilon^2(\sigma_{j'k''l'})\}\right] = \sum_{k'',l'}\rho^2\left[\frac{\partial U_{i'}}{\partial q_{i'k'l}}, q_{i'k''l'}\right]\epsilon^2[q_{i'k''l'}] + \frac{2\epsilon^2[\sigma_{jk'l}]}{\rho^2[\sigma_{jk'l},\lambda_{jk}]PR^2}.$$

Since the route QoS relative precision can be expressed in terms of the link QoS relative precisions using (18), $f[\cdot]$ can be ex-

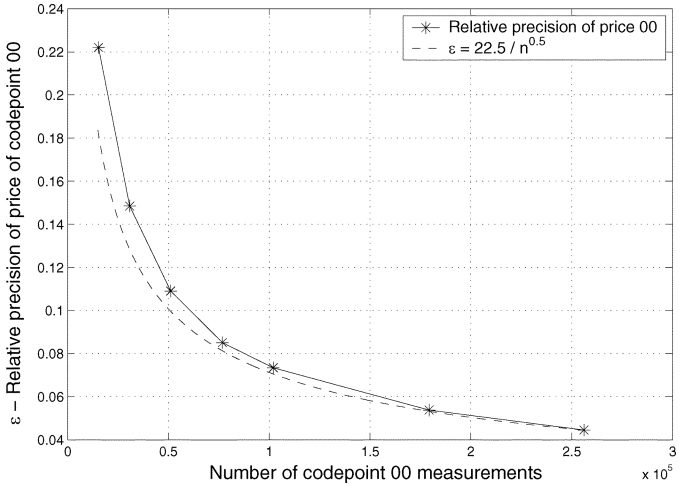


Fig. 5. Relative precision of price of codepoint 00 versus number of measurements.

pressed as a linear combination of $\{\epsilon^2[\sigma_{j'k''l'}], \forall k'', l', \forall j' \in r_{i'}\}$.

To conclude this part of the analysis, we focus on relationship between the relative precisions and the number of measurements. Above we established that the relative precision in the QoS measurement is inversely proportional to the square root of the number of measurements: $\epsilon[\sigma_{jkl}] \sim 1/\sqrt{n_{jk}}$. We then found in (17) that the relative precision of the QoS sensitivity is proportional to the relative precision in the corresponding link QoS measurement, so it follows that $\epsilon\left[\frac{\partial \sigma_{jk'l}}{\partial \lambda_{jk}}\right] \sim 1/\sqrt{n_{jk'}}$. Carrying the analysis from the link to route level, we must consider the number of codepoint k packets generated by user i during one iteration, denoted by N_{ik}. For links on user i's route, namely $\forall j \in r_i$, the number of codepoint k packets generated by user i during one iteration, N_{ik}, is linearly proportional to the number of measurements of σ_{jkl}, n_{jk}, with the proportionality constant depending on the ratio of user i's traffic to link j traffic on codepoint k. Hence (18) reveals that the relative precision of the route QoS measurement follows a similar pattern $\epsilon[q_{ikl}] \sim 1/\sqrt{N_{ik}}$.

In (19), we established that the relative precision in the utility sensitivity depends on the relative precisions of the route QoS measurements on each codepoint. Denote the total number of packets generated by user i during one iteration by N_i. Assuming that the number of codepoints is small, $N_i \sim N_{ik}, \forall k$, so it follows that $\epsilon\left[\partial U_i/\partial q_{ikl}\right] \sim 1/\sqrt{N_i}$.

Finally, we gave an expression for the relative precision in price in (20). The relative price error depends on the relative precisions in route QoS measurement and in link QoS measurement. These can be combined to show that $\epsilon[p_{ik}] \sim 1/\sqrt{N_{ik}}$.

As an example, for the queue considered in the previous subsections, Fig. 5 shows the relative precision of the price for codepoint 00 packets versus the number of measurements per iteration. The relative price precision is proportional to the weighted quadratic mean of a number of link QoS relative precisions and link QoS sensitivities relative precisions. Compared with the previous figures, we can see that the relative price precision is intermediate between the QoS relative precision and the QoS sensitivity relative precision. For instance, at 2×10^5 measurements per iteration, $\epsilon[p_{ik}] \approx 5\%$.

Finally, we note that relative precisions of rates and utilities will demonstrate similar dependencies upon the number of measurements. The errors in prices will in turn affect users' adjustment of rates and the obtained utility; consequently it can be shown that $\epsilon[\lambda_{ik}] \sim 1/\sqrt{N_{ik}}$ and $\epsilon[U_i] \sim 1/\sqrt{N_i}$.

F. Feasibility of Congestion-Based Pricing

We now return to the second principal question that we address in this paper: how quickly can congestion-based pricing react to fluctuations in network traffic, namely how effective will it be? In particular, we now consider how errors in price, rate, and utility are affected by network topology and network bandwidth through the number of autonomous systems, the throughput on each codepoint, and the round-trip time.

First consider the effect of the number of autonomous systems upon the relative precisions. QoS and prices are summed along each route. However, in (18) we found that the relative precisions of the route QoS measurements are proportional to the weighted quadratic means of the relative precisions of link QoS measurements. The relative precisions in prices are also proportional to weighted quadratic means. It follows that although an increasing number of autonomous systems along a route may increase the relative precisions, the relationship is fairly weak. For instance, if the link QoS relative precisions $\epsilon[\sigma_{jkl}]$ are bounded for all links j along route i, i.e., $\epsilon[\sigma_{jkl}] \leq \epsilon_{\sigma_{kl}}, \forall j \in r_i$, then the route QoS relative precision will be similarly bounded $\epsilon(q_{ikl}) \leq \epsilon_{\sigma_{kl}}$, and (19) is reduced to

$$\epsilon\left[\frac{\partial U_i}{\partial q_{ikl}}\right] \leq \sqrt{\sum_{k',l'} \rho^2\left[\frac{\partial U_i}{\partial q_{ikl}}, q_{ik'l'}\right] \epsilon^2_{\sigma_{k'l'}}}$$

and the upper bound is independent of the number of links in user i's route.

Next consider the combined effect of the throughput and the round-trip time on relative precisions. As noted above, the time T_{AS} required to collect a sufficient number of QoS measurements within an autonomous system depends on the throughput on each codepoint, $n_{jk} = \lambda_{jk} T_{AS}$, and so the relative precision of QoS measurements $\epsilon[\sigma_{jkl}] \sim 1/\sqrt{\lambda_{jk} T_{AS}}$. The same relationships hold for the relative precision of the QoS sensitivities within an autonomous system. Therefore within an autonomous system, the minimum time per iteration T_{AS} is proportional to square of the desired relative precision and inversely proportional to the throughput $T_{AS} \sim \epsilon^2\left[\frac{\partial \sigma_{jk'l}}{\partial \lambda_{jk}}\right]/\lambda_{jk}$. When moving from link to route level, however, any exchange of information requires a round-trip time (RTT). It follows that the minimum time per iteration at the route level T_R is $T_R = T_p + RTT$ where the time required for price estimation $T_p \sim \epsilon^2[p_{ik}]/\lambda_{ik}$.

The minimum time per iteration therefore strongly depends on the network bandwidth. While T_p is much larger than RTT, increases in network bandwidth result in proportional decreases in the iteration time. Returning to the example used in this section, if a relative precision of 5% or less is desired in the price of codepoint 00, this requires approximately 2×10^5 measurements. If network bandwidth is 10 Gbps, then collection of these measurements takes only 0.4 s, which is likely to be on the same order as round-trip time. On the other hand, if the network bandwidth is only 10 Mbps, then collection requires 400 s,

which greatly dominates the round-trip time. The proportionality of the relative precisions of price, rates, and utility to $1/\sqrt{n}$ means that the relative precision achieved in the 10 Mbps system within 0.4 s is over 31 times larger than the relative precision in the 10-Gbps system within the same time. The network bandwidth is thus likely to determine the viability of congestion-based pricing more so than the particular details of the method of estimation.

This minimum time per iteration places an important limitation on the effectiveness of congestion-based pricing in differentiated services networks. Congestion-based pricing is considered effective only if prices can be be updated quickly enough to track congestion in the network. For networks with a small number of users, congestion changes on the same time scale as fluctuations in a single user's traffic. In these situations, congestion-based pricing is often too slow to respond, and open-loop flow control is used instead. For networks with a large number of users, congestion changes on a time scale related to the arrival and departure of users. In these situations, congestion-based pricing is often considered as a flow control or connection admission control mechanism. Our results here indicate that in differentiated services networks, *congestion-based pricing may only be effective only if the network bandwidth is high enough so that the minimum time per iteration is smaller than the time scale on which congestion changes*.

In the next section, we consider this issue further using small network examples.

V. Numerical Analysis

We now consider a simple network with two users and one autonomous system. The network bandwidth is set to 10 Mbps. While we have analyzed much larger examples, this simple network more effectively illustrates the main ideas. Furthermore, as discussed in the last section, the relative precisions are only weakly dependent upon the number of autonomous systems. As before, denote user i's rates on each codepoint by $\vec{\Lambda}_i$. Suppose user i's QoS is represented by delay and loss on each codepoint, $\vec{D}_i$ and $\vec{L}_i$ respectively. We adopt a utility function that is additive with respect to the utility earned on each codepoint $U_i(\vec{\Lambda}_i, \vec{D}_i, \vec{L}_i) = \sum_k w_{ik} U_{ik}$ where w_{ik} is the weight associated with codepoint k. U_{ik} is the utility earned on codepoint k, and is modeled as

$$U_{ik}(\Lambda_{ik}, D_{ik}, L_{ik}) = N_i\,(\Lambda_{ik}/N_i)^{0.85}(1-d_i D_{ik}^{1,1})(1-l_i L_{ik}^{1,1})$$

where N_i is the number of individual flows within user i that share the same route and the same set of codepoints, and where $d_i > 0$ and $l_i > 0$ are user-specific constants. The utility earned on codepoint k is increasing and concave with respect to the rate (under fixed delay and loss), and decreasing and concave with respect to delay and loss on that codepoint (under fixed rate).[4] The sensitivity of a user to delay (respectively, loss) is determined by d_i (respectively, l_i), and higher values of d_i (respectively, l_i) indicate a lower tolerance for delay (respectively, loss). In the numerical example, we set $N_i = 50$, $\forall i$, $w_{ik} = 0.25$, $\forall i, k$, $d_1 = 1$, $l_1 = 6$, $d_2 = 200$, and $l_2 = 1$. Hence, user 1 is more sensitive to loss than to delay, while user 2 is more sensitive to delay than to loss. Many other utility functions could be chosen as long as the assumptions are satisfied. The choice of utility function is likely to be strongly related to how an application perceives quality of service.

[4]Note, however, that this is not sufficient to guarantee that the corresponding user utility V_i is a twice differential concave function with bounded Hessian, as assumed in the model. This can only be verified experimentally, since we have no analytical expressions for delay and loss.

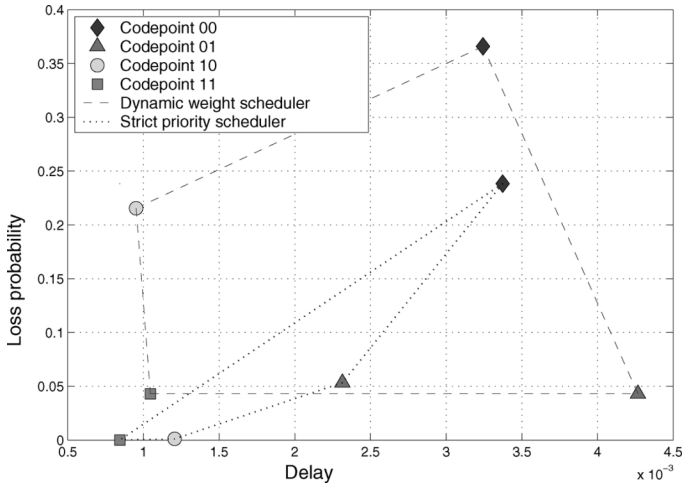


Fig. 6. QoS on each codepoint.

We consider two generic diffServ schedulers that offer four diffServ codepoints. The first one, here called a *strict priority scheduler*, operates as follows: a packet is inserted into the transmission buffer behind previous packets of the same codepoint but ahead of lower codepoint packets. The scheduler transmits the packet at the head of the buffer. If the buffer is full, the scheduler drops the packet at the tail of the buffer. Clearly, under this scheduler each codepoint will experience both lower delay and lower loss than codepoints with lower priority. In Fig. 6, we show the quality of service obtained on each codepoint under this scheduler, using the rates for each user that maximize total utility. As expected, each codepoint experiences both lower delay and lower loss than codepoints with lower priority, in the order 11, 10, 01, and 00.

Although this scheduler does provide differentiated service among the four codepoints, delay and loss are strongly correlated. None of the codepoints provides relatively low loss and high delay (or vice versa). For this reason, more advanced schedulers often consider priority for delay and loss separately. An example of such a *dynamic weight scheduler* operates as follows. Let the first bit of the codepoint be used to indicate priority for low delay and the second bit be used to indicate priority for low loss, e.g., codepoint 10 should obtain low delay but may obtain moderate loss. When the scheduler is ready to serve a packet, the oldest 11 or 10 packet in the buffer is served if available; if not then the oldest 00 or 01 packet is served. This ensures these codepoints obtain a lower delay than codepoints 00 and 01. If the scheduler needs to drop a packet from the queue, then the newest 00 or 10 packet is dropped if available; if not then the newest 11 or 01 packet is dropped. As illustrated in Fig. 6, under the dynamic weight scheduler the codepoints for which delay is a high priority (10 and 11) achieve low delay, while the codepoints for which loss is a high priority (01 and 11) achieve low loss. As a result, the region of achievable

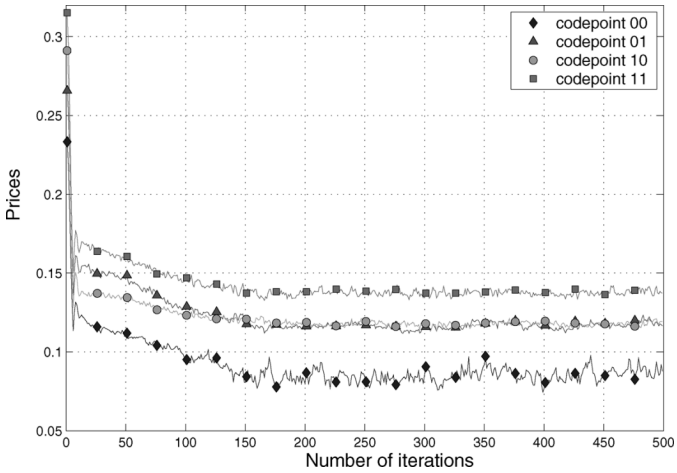


Fig. 7. Convergence of prices under a dynamic weight scheduler.

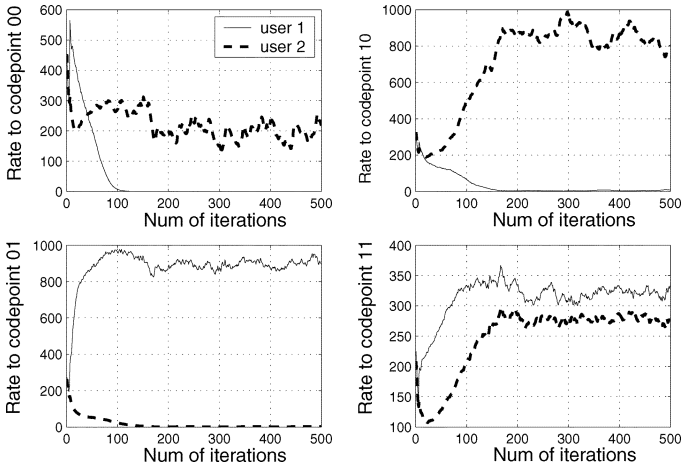


Fig. 8. Dynamic allocation of traffic rates on each codepoint for both users.

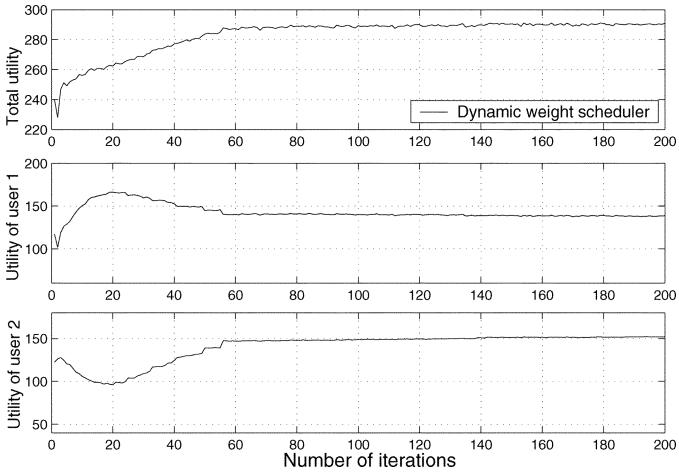


Fig. 9. Utilities versus number of iterations.

combined delay and loss is larger than under the static priority scheduler. Actually, under the dynamic weight scheduler, the pricing algorithm and the scheduling policy interplay in a more desired way to match different user QoS requirements, and consequently corresponds to higher economic efficiency. However, although the dynamic weight scheduler can provide clearly differentiated QoS to each of the four codepoints, as the total flow rate increases, the QoS for all four codepoints degrades, with variations in the QoS that can exceed an order of magnitude even for the highest priority codepoint.

A. Convergence of Prices, Rates, and Utilities

We now turn to examining the convergence of prices, rates, and utilities. We set the initial demands on each codepoint of user 1 and user 2 to $\Lambda_i = \{445, 320, 268, 215\}$, $i = 1, 2$, in terms of number of packets per second (where the packet size is set to 4000 bits). The convergence of prices on each codepoint is shown in Fig. 7. Prices converge to close to their optimal values within approximately 150 iterations. At the optimal, $p_{11} > p_{10} \approx p_{01} > p_{00}$.

The convergence of traffic rates on each codepoint of user 1 and user 2 is shown in Fig. 8. Under the dynamic weight scheduler, the QoS experienced by each codepoint is widely differentiated. The loss probability on codepoint 00 and 10 are 30% and 16% respectively, which are too large for user 1 to mark any packets with these two codepoints. Thus user 1, who desires low loss, marks almost all its packets with codepoints 01 and 11, the codepoints for which loss is a high priority. In contrast, user 2, who desires low delay, marks packets with codepoints 10 and 11, the codepoints for which delay is a high priority, and 00, which is relatively inexpensive and still has acceptable latency. In Fig. 9, we show the convergence of utility of each user, and the total utility of both users. The utility depends on both the rate and QoS on each codepoint. The congestion-based pricing algorithm quickly increases the total utility to the maximum achievable.

The result suggests that the availability of both price and QoS information is crucial for diffServ congestion-based pricing schemes to target particular levels of QoS. Also the estimation of both sensitivities of QoS with respect to traffic rates and marginal utilities with respect to QoS is beneficial for diffServ congestion-based pricing to coordinate with the user applications to tune the QoS to application requirements.

B. Effect of Different Time Scales

In this subsection, we consider how the pricing algorithm behaves under different time scales on which congestion changes. To investigate the behavior, we keep the number of individual flows within user 2, N_2, constant, and vary the number of flows within user 1, N_1. By varying N_1 at different rates, we observe when prices and demands can keep up with varying traffic and when they cannot. In each scenario, we use the dynamic weight scheduler introduced above, but we add pricing smoothing: instead of using (9) to calculate the new prices in each iteration, we use a moving average $\vec{p}_i^{k+1} = 0.7\vec{p}_i^{k} + 0.3\vec{p}_i$, where $\vec{p}_i$ is the price vector calculated from (9), and $\vec{p}_i^{k}$ is the price vector in previous iteration.

The first scenario is meant to represent relatively slow changes in user load. We start with $N_1 = 50$, $N_2 = 50$ under the optimal allocation. We then vary N_1 between 50 to 100 in a sawtooth pattern, by increasing or decreasing it by one every three iterations. After 450 iterations, N_1 is held constant at 100. The dynamics of prices and of arrival rates on selected

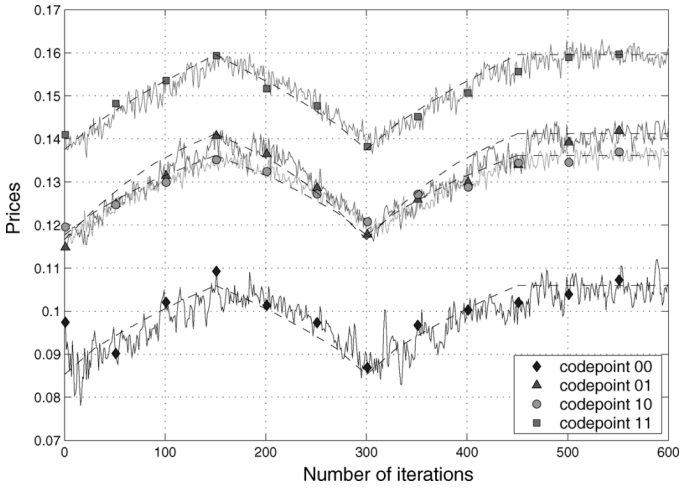


Fig. 10. Dynamic prices when N_1 varies by one in every three iterations.

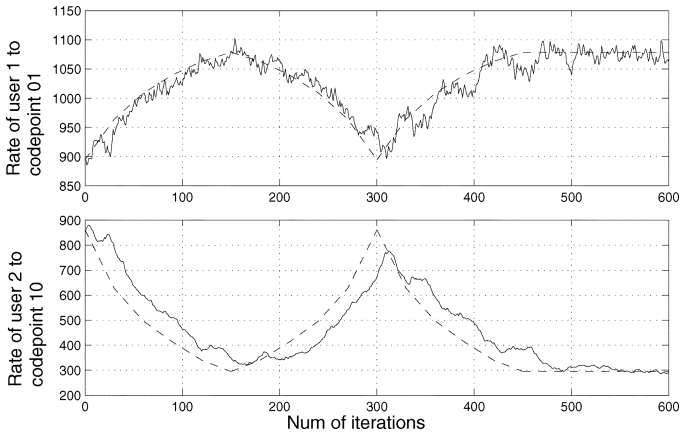


Fig. 11. Dynamic allocation of rates when N_1 varies by one in every three iterations.

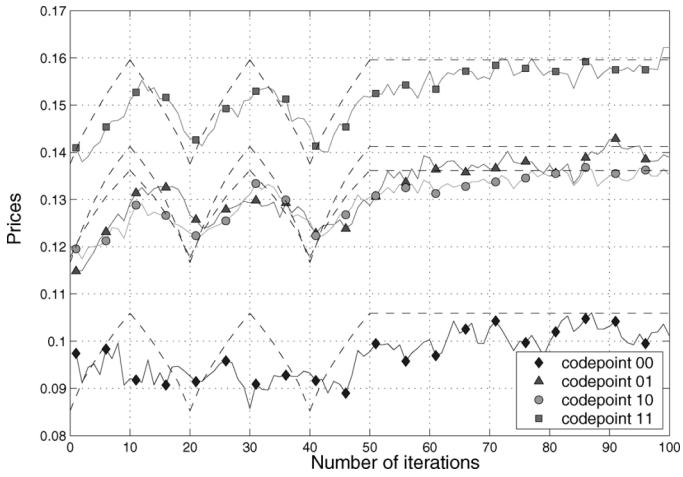


Fig. 12. Dynamic prices when N_1 varies by five in every iteration.

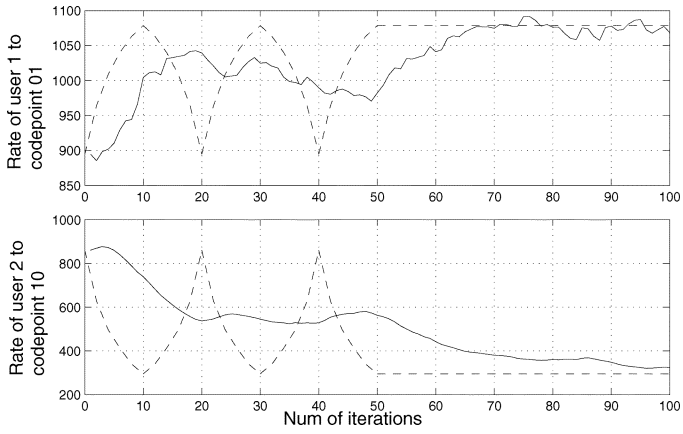


Fig. 13. Dynamic allocation of rates when N_1 varies by five in every iteration.

codepoints are shown in Figs. 10 and 11. For comparison, the dashed curves depict the optimal prices and rates corresponding to the current traffic mix. The prices follow the general trend of the traffic dynamics well, with a lag of five to six iterations due to the price smoothing. The rates also follow the general trend well, with a lag of about 20 iterations due to our selection of step size s^k in (13) and to price smoothing. (Increasing the step size reduces the lag but also reduces stability.)

The second scenario is meant to represent relatively fast changes in user load. We again start with $N_1 = 50$ and $N_2 = 50$, but then vary N_1 in a sawtooth pattern by increasing or decreasing it by five every one iteration. After 50 iterations, N_1 is held constant at 100. The dynamics of prices and rates are shown in Figs. 12 and 13. The prices now have considerable difficulty following the traffic dynamics, since they do not have time to converge to the optimum before the number of users changes direction. The rates have even greater difficulty keeping up with the changing traffic. As a result, congestion-based pricing is ineffective.

In both scenarios, the price on codepoint 00 fluctuates more than prices on other codepoints. This occurs because the rate elasticities of QoS, $\rho[\sigma_{jk'l}, \lambda_{jk}]$ with respect to the traffic on codepoint 00 is significantly lower than the rate elasticities with respect to the traffic on other codepoints. Such low rate elasticity increases the relative precision of price on codepoint 00 compared to other codepoints.

The numerical results tell us that congestion-based pricing can be effective only if prices can be updated quickly enough to track congestion in the network. For example, in the experiments we presented above, the network bandwidth is only 10 Mbps (so that every iteration takes 400 s to achieve a relative precision of 5% in the price of codepoint 00), and if users arrive or depart at least every 80 s (i.e., five arrivals per iteration), then congestion-based pricing will be ineffective as depicted in Figs. 12 and 13. In contrast, if the network bandwidth is 10 Gbps (so that every iteration requires only 0.4 s to achieve a relative precision of 5% in the price of codepoint 00) and if users arrive or depart no more often than every 1.2 s (three iterations), then congestion-based pricing can be fairly effective.

A great deal of additional research is required to make any such congestion-based pricing approach practical, including design of signalling protocols, measurement algorithms, user agents to automate user response, scheduling policies, and feedback algorithms to guarantee convergence. In particular, there may be more efficient methods to produce the required sensitivity estimates. Autonomous systems might derive analytical formulae that estimate QoS sensitivities as a function

of rates and QoS. More likely, autonomous systems might observe traffic over many days, calculate QoS sensitivities for a set of rate vectors, and in real time interpolate between these calculated QoS sensitivity matrices to estimate the sensitivities for the current rate vector. Such an approach requires further study, but has the potential to significantly reduce the measurement error in QoS sensitivity and thus reduce the error in prices. Noncooperative approaches that do not rely on release of private information also merit further study, as they may require less information than the approach studied here.

REFERENCES

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

[2] A. Odlyzko, "Paris metro pricing for the internet," in *Proc. ACM Conf. Electronic Commerce (EC'99)*, 1999, pp. 140–147.

[3] C. Courcoubetis and V. Siris, "Managing and pricing service level agreements for differentiated services," in *7th Int. Workshop on QoS*, 1999, pp. 165–173.

[4] P. Marbach, "Pricing priority classes in a differentiated services network," presented at the Allerton Conf. Communication, Control, and Computing, Monticello, IL, 1999.

[5] A. Orda and N. Shimkin, "Incentive pricing in multiclass systems," *Telecommun. Syst.*, vol. 13, pp. 241–267, 2000.

[6] P. Marbach, "Pricing differentiated services networks: Bursty traffic," in *Proc. IEEE INFOCOM*, 2001, pp. 650–658.

[7] J. Altmann, H. Daanen, H. Oliver, and A. S.-B. Suárez, "How to market-manage a QoS network," in *Proc. IEEE INFOCOM*, 2002, pp. 284–293.

[8] P. Marbach, "Priority service and max-min fairness," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 733–746, Oct. 2003.

[9] N. Semret, R. Liao, A. Campbell, and A. A. Lazar, "Pricing, provisioning and peering: Dynamic markets for differentiated internet services and implications for network interconnections," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2499–2513, Dec. 2001.

[10] E. W. Fulp and D. S. Reeves, "Optimal provisioning and pricing of differentiated services using QoS class promotion," in *Proc. INFORMATIK: Workshop on Advanced Internet Charging and QoS Technology*, 2001.

[11] A. Gupta, D. O. Stahl, and A. B. Whinston, "Priority pricing of integrated services networks," *Internet Economics*, pp. 323–352, 1997.

[12] L. He and J. Walrand, "Pricing differentiated Internet services," in *Proc. IEEE INFOCOM*, 2005, pp. 195–204.

[13] K. Park, M. Sitharam, and S. Chen, "Quality of service provision in noncooperative networks: Heterogenous preferences, multi-dimensional QoS vectors, and burstiness," in *Proc. Int. Conf. Information and Computation Economies*, 1998, pp. 111–127.

[14] J. K. MacKie-Mason and H. Varian, "Pricing congestible resources," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 7, pp. 1141–1149, Sep. 1995.

[15] H. Jiang and S. Jordan, "The role of price in the connection establishment process," *Eur. Trans. Telecommun.*, vol. 6, no. 4, pp. 421–429, Jul.-Aug. 1995.

[16] C. Courcoubetis and R. Weber, *Pricing Communication Networks: Economics, Technology, and Modelling*. Hoboken, NJ: Wiley, 2003.

[17] N. Jin, G. Venkitachalam, and S. Jordan, "Dynamic congestion-based pricing of bandwidth and buffer," *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1233–1246, Dec. 2005.

[18] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.

[19] S. Low and D. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–875, Dec. 1999.

[20] R. J. La and V. Anantharam, "Utility-based rate control in the internet for elastic traffic," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 272–286, Apr. 2002.

[21] R. Srikant, *The Mathematics of Internet Congestion Control*. Cambridge, MA: Birkhauser, 2004.

[22] T. Alpcan and T. Basar, "A globally stable adaptive congestion control scheme for Internet-style networks with delay," *IEEE/ACM Trans. Netw.*, vol. 13, no. 6, pp. 1261–1274, Dec. 2005.

[23] H. Yaiche, R. R. Maxumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

[24] X.-R. Cao, H.-X. Shen, R. Milito, and P. Wirth, "Internet pricing with a game theoretical approach: Concepts and examples," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 208–216, Apr. 2002.

[25] J. Qiu and E. W. Knightly, "Measurement-based admission control with aggregate traffic envelopes," *IEEE/ACM Trans. Netw.*, vol. 9, no. 2, pp. 199–210, Apr. 2001.

[26] C. A. Courcoubetis, A. Dimakis, and G. D. Stamoulis, "Traffic equivalence and substitution in a multiplexer with applications to dynamic available capacity estimation," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 217–231, Apr. 2002.

[27] L. Breslau and S. Shenker, "Best-effort versus reservations: A simple comparative analysis," in *SIGCOMM Symp. Communications Architectures and Protocols*, 1998.

[28] L.-T. Park, J.-W. Baek, and W.-K. J. Hong, "Management of service level agreements for multimedia Internet service using a utility model," *IEEE Commun. Mag.*, vol. 39, no. 5, pp. 100–106, May 2001.

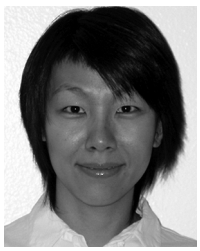
[29] R. K. Sundaram, *A First Course in Optimization Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[30] J. Rosen, "Existence and uniqueness of equilibrium points for concave N-person games," *Econometrica*, vol. 33, pp. 520–534, 1965.

[31] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[32] K. Pawlikowski, "Steady-state simulation of queueing processes: A survey of problems and solutions," *ACM Comput. Surv.*, vol. 22, no. 2, pp. 123–170, Jun. 1990.

[33] G. S. Fishman and L. S. Yarberry, "An implementation of the batch means method," *INFORMS J. Comput.*, vol. 9, pp. 296–310, 1997.



Nan Jin received the B.S degree from Nanjing University, Nanjing, China, in 1997, the M.S. degree from the Chinese Academy of Sciences, Beijing, in 2000, and the Ph.D. degree from the University of California, Irvine, in 2005.

She is currently a Senior Software Engineer at Watchguard Technologies, Tustin, CA, in the area of QoS and traffic management in large-scale networking systems.

Scott Jordan (S'86–M'90) received the B.S./A.B., M.S., and Ph.D. degrees from the University of California, Berkeley, in 1985, 1987, and 1990, respectively.

From 1990 until 1999, he served as a faculty member at Northwestern University. Since 1999, he has served as a faculty member at the University of California, Irvine. During 2006, he served as an IEEE Congressional Fellow, working in the U.S. Senate on Internet and telecommunications policy issues. His research interests currently include pricing and differentiated services in the Internet, resource allocation in wireless multimedia networks, and telecommunications policy.