An Optimization Based Approach for QoS Routing in High-Bandwidth Networks

Xiaojun Lin, *Member, IEEE* and Ness B. Shroff, *Senior Member, IEEE*

***Abstract*— In this paper, we propose an optimization based approach for Quality of Service routing in high-bandwidth networks. We view a network that employs QoS routing as an entity that distributively optimizes some global utility function. By solving the optimization problem, the network is driven to an efficient operating point. In earlier work, it has been shown that when the capacity of the network is large, this optimization takes on a *simple form*, and once the solution to this optimization problem is found, simple proportional QoS routing schemes will suffice. However, this optimization problem requires global information. We develop a distributed and adaptive algorithm that can efficiently solve the optimization online. Compared with existing QoS routing schemes, the proposed optimization based approach has the following advantages: (1) The computation and communication overhead can be greatly reduced without sacrificing performance; (2) The operating characteristics of the network can be analytically studied; and (3) The desired operating point can be tuned by choosing appropriate utility functions.**

***Index Terms*— QoS routing, high-bandwidth networks, optimization based approach.**

I. Introduction

Future telecommunication networks are expected to support applications with diverse Quality of Service requirements. Quality of Service (QoS) routing is an important component of such networks and has received considerable attention over the past decade (for a good survey, see [2] and the reference therein). The objective of QoS routing is two-fold: to find a feasible path for each incoming connection; and to optimize the usage of the network by balancing the load.

In this paper, as in the majority of studies on QoS routing, we assume a source routing model where routing decisions are made at the point where connection requests originate. In most of these studies, researchers take the following view of the QoS routing problem: The links are "dumb" and they advertise their status. The intelligence lies in the end-systems (sources or edge routers) to compute paths based on the current knowledge of the link states.

The above paradigm would have worked well if the link states were stable. However, not all link state metrics are stable. In particular, the available bandwidth metric of a link is inherently dynamic and changes frequently as connections enter and leave the network. Therefore, the link state advertisement and the QoS routing algorithm have to be executed frequently in order to keep up with the changes in link states. This leads to a significant amount of computation and communication overhead. To reduce the computation and communication burden, the frequency of the computation and the link state updates then need to be contained. This could, however, result in staleness of the link state information and inaccuracy in the routing decisions. Hence, there is a fundamental tradeoff between the amount of computation and communication resources consumed and the quality of the routing decisions. This tradeoff is usually difficult to analyze and researchers have had to resort to simulation studies [3], [4], [5], [6]. These studies reveal that the performance of existing QoS routing schemes degrades when computation and link state updates become infrequent. However, the extent to which the performance degrades depends not only on how infrequently the computation and link state updates are made, but also on a large number of other factors that include: the specifics of the path computation algorithm, the topology and the demand pattern of the network, the cost metrics assigned for each link, the link state update strategy, and the strategy to handle routing failures, etc. In general, the exact level of performance degradation is hard to predict.

In this paper, we take a different view of the QoS routing problem. We view the network (including the end-systems *and* the links) that employs QoS routing as an integral entity that jointly optimizes some global utility function. Once the solution to this optimization problem is found, the network will be driven to an efficient operating point, and the routing performance will be close to optimal. No further computation and communication are needed as long as the prevailing network condition remains essentially unchanged.*

We refer to our proposed scheme as the optimization based approach for QoS routing. When the capacity of the network is large, this optimization takes on a simple form. Our proposal is based on a known result: *simple proportional routing schemes* can approach the performance of the *optimal dynamic routing schemes* when the capacity of the network is large [7], [8], [9]. In a *proportional routing scheme*, calls are routed to alternate paths based on pre-determined probabilities. The right routing probabilities can be derived from the solution

Manuscript received xxxxxxxxx xx, 200x; revised xxxxxxxxx xx, 200x. This work has been partially supported by the NSF grant ANI-0099137 and the Indiana 21st Century Fund 651-1285-074. An earlier version of this paper has been presented at IEEE INFOCOM 2004 [1].

The authors are with the Center for Wireless Systems and Applications (CWSA) and the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (email: {linx,shroff}@ecn.purdue.edu).

*In practice, some computation and communication will still be required to track changes in the network condition. However, a nice feature of our work is that computation and communication intensive operations can be done at very long time-scales, with a negligible impact on performance.

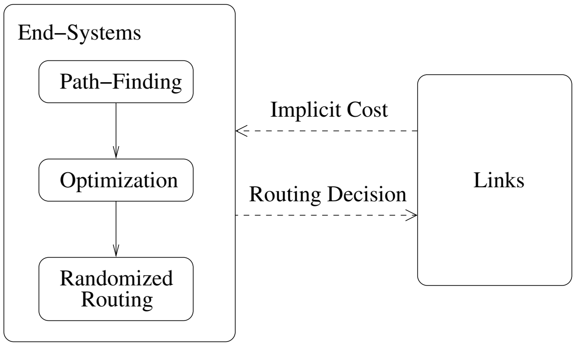


Fig. 1. Our Optimization Based Approach

of a simple optimization problem that depends only on the *average* demand and capacity of the network.

We develop an online, distributed algorithm that can efficiently solve the optimization problem. Fig. 1 provides a high-level view of the optimization based approach. Each link in the network is associated with an implicit cost. The implicit cost summarizes the congestion level at the link and can be updated by the observed demand and capacity at the link. Thus, we equip the link with only a minimal amount of intelligence (i.e., to update the implicit cost). It turns out that the implicit cost is the only information that the end-system needs to solve the optimization problem. The end-system has three components: a path-finding component that maintains a set of alternate paths; an optimization component that solves for the optimal routing probabilities; and a randomized routing component that routes each incoming connection based on the precomputed routing probabilities.

Compared with existing QoS routing schemes, our optimization based approach has the following advantages:

(1) The computation and communication overhead can be greatly reduced without sacrificing performance. Once the optimal operating point is found, the same routing parameters can be used by a large number of future arrivals, as long as the *average* network condition remains unchanged. Infrequent computation and link state updates will only affect the speed of convergence of the distributed algorithm, but not the end-result that the algorithm converges to.

In practical networks, the *average* network condition can also change gradually over time (non-stationary behavior), e.g., during the course of a day. Our distributed algorithm will track the changes in the average network condition and adjust the operating point accordingly. Note that in a control system, there has always been the issue of the right *time scale of control*. A nice feature of our proposed solution is that, the control that needs to be done at a fast time scale, i.e., the randomized routing, is very simple; while the control that requires a large amount of computation, i.e., the optimization of routing probabilities and the search for new alternate paths, can be carried out over a much slower time scale. Using the right separation of control time scales, our optimization based approach ensures near optimal performance even when the computation and communication become infrequent.

(2) The operating characteristics of the network can be analytically studied. Given the network model, we can easily predict the operating point by solving the optimization problem. In contrast, due to the complexity of the system, the analysis of existing QoS routing schemes appears to be intractable, especially under inaccurate link state information and infrequent computation.

(3) The desired operating point can be tuned by appropriately choosing the utility functions. The optimization based approach allow us not only to *predict* the operating point of the network, but also to *control* it. By choosing different utility functions for different classes and source-destination pairs, we can achieve the desired balance among the service levels offered to different groups of users. For example, when the network becomes congested, connections with a larger number of hops could suffer significantly more blocking than shorter connections. In our optimization based approach, this can be avoided by assigning longer connections a utility function that has a higher marginal utility.

A. Related Work

The optimal control of loss networks has been studied extensively in the past. Both off-line [10], [11], [12] and simulation based schemes [13] have been proposed. Our contribution is to propose an *online* solution for QoS routing. Our online scheme exploits the fact that simplicities arise in high-bandwidth networks, which we will discuss in more detail in Section II. These results lead to a much simpler and easily decomposable optimization problem.

Our proposed solution employs a proportional routing scheme. The asymptotic optimality of the proportional routing scheme in large systems has been known for some time [7], [9]. However, a major criticism of proportional routing schemes has been the following: if the demand is incorrectly estimated, the computed routing probabilities could lead to poor performance [13]. We solve this problem by using an *adaptive* algorithm that does not rely on any prior knowledge of the demand. The Adaptive Proportional Routing scheme proposed in [14], [15] is also related to our work. In their scheme, each class measures the amount of blocking along each alternate paths, and uses the inverse Erlang formula to estimate a "virtual capacity" grabbed by the class along each path. Then each class locally optimizes the routing probabilities based on the demand and these virtual capacities. Compared with the Adaptive Proportional Routing scheme, the advantage of our optimization based approach is that the optimality of the resulting operating point and the convergence of the algorithm can be rigorously shown. Further, the implicit costs provide additional information for discovering new alternate paths.

The mathematical structure of the optimization problem studied in this paper is closely related to those found in *multi-path* flow control problems [16], [17], [18], [19]. In [16], two classes of solutions to flow control problems are categorized, i.e., primal solutions and dual solutions. For the *single-path* flow control problem, both the primal and the dual solutions have been studied extensively (see [20] for a good survey). On the other hand, the *multi-path* flow control problem has received less attention. Primal solutions to the *multi-path* flow control problem were developed in [16], [18], [19]. These primal solutions are based on a penalty function approach, i.e., they replace the capacity constraints by a penalty function in the optimization objective. Primal solutions tend to produce biased approximates of the optimal operating point [16], [19], due to the fact that penalties will only be incurred when the capacity constraints are violated. (In contrast, the optimal operating point is defined to be one that *satisfies* the capacity constraints.) Further, because the penalty functions are nonlinear, an additional level of bias occurs in the primal solutions in [16], [18], [19] when there is noise in the measurements of the offered load at each link. In our application setting of QoS routing, such biases of primal solutions translate into suboptimal routing decisions and unnecessary blocking of calls, which are undesirable in practice[†].

Our implicit cost based solution can be viewed as a dual solution to the multi-path optimization problem. The advantage of dual solutions is that they are designed to compute the exact optimal operating point. In fact, even when there is measurement noise, it is possible to show that our dual solution converges to the exact optimal operating point when the stepsizes are driven to zero in an appropriate fashion. The algorithm proposed in [17] is also similar to our dual solution. In [17], the authors state that their algorithm is one of the Arrow-Hurwicz algorithms [22]. However, the convergence of the Arrow-Hurwicz algorithm was established in [22] only for the case when the objective function is strictly concave, which is not true for the problem at hand. In this paper, we report a new result that correctly characterizes the convergence.

Finally, our work exploits the largeness of the network to simplify QoS routing. For other works that exploit the largeness of the network, one can refer to [8], [23], [24] (on pricing) and [25] (on network decomposition).

The rest of the paper is organized as follows: In Section II, we present the asymptotic optimality of the proportional routing scheme. In Section III, we derive the distributed algorithm for computing the optimal routing probabilities and obtain the proposed QoS algorithm. We discuss implementation issues in Section IV, present simulation results in Section V, and then conclude.

II. Simplification of QoS Routing in Large Networks

A. The Model

We adopt a multi-class loss network model. There are L links in the network. Each link $l \in \{1, ..., L\}$ has capacity R^l. There are I classes of users. Each class is associated with one source-destination pair, and some given QoS requirements. Flows of class i arrive to the network according to a Poisson process with rate λ_i. Once admitted, a flow of class i will hold r_i amount of bandwidth. (For the moment we assume that bandwidth is the only QoS metric. The extension to multiple QoS metrics will be addressed in Section III-D.) The service times within a class are *i.i.d.* and independent of the arrival process. The service time distribution is general with mean $1/\mu_i$. Each admitted flow of class i generates v_i amount of revenue per unit time. The objective of the network is to maximize the revenue from all flows admitted into the network.

Such a network model could represent the backbone of an ISP serving applications with different QoS requirements. The revenue v_i could either be actual money, or simply an assigned weight that represents the network's preference for each class. The bandwidth requirement r_i could be some form of *effective bandwidth* for flows of class i. There could be multiple classes associated with each source-destination pair, differing in their bandwidth requirement r_i and revenue v_i.

In this section, we assume that each class i has set up $\theta(i)$ alternate paths using, for example, MPLS [26] (we will address how these alternate paths can be found in Section III-C). The alternate paths are represented by a matrix $[H_{ij}^l]$ such that $H_{ij}^l = 1$ if path j of class i uses link l, and $H_{ij}^l = 0$ otherwise. We denote the state of the system by a vector $\vec{n} = [n_{ij}, i = 1, ...I, j = 1, ..., \theta(i)]$, where n_{ij} is the number of flows of class i currently using path j. The bandwidth requirements and the capacity constraints then determine the set of feasible states $\mathbf{\Omega_n} = \{\vec{n} : \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} n_{ij} r_i H_{ij}^l \leq R^l \text{ for all } l\}$.

We denote the routing decision (which can be time varying) for class i by a vector $\vec{p}_i = [p_{i1}, p_{i2}, ..., p_{i,\theta(i)}]$, where

$$p_{ij} = \mathbf{Pr}\{\text{an incoming flow of class } i \text{ is routed to path } j\}.$$

An incoming flow of class i will be admitted with probability $\sum_{j=1}^{\theta(i)} p_{ij}$, and, if admitted, it will be routed to path j with probability $p_{ij}/\sum_{k=1}^{\theta(i)} p_{ik}$. Note that $\vec{p}_i \in \Omega_i$, where

$$\Omega_i \triangleq \{p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, \text{ for all } j\}.$$

Let $\vec{p} = [\vec{p}_1, ..., \vec{p}_I]$.

A *dynamic routing scheme* is one where routing decisions can adapt to the changing utilization level of the network. For example, $\vec{p}(t)$ can be a function of the current state of the network, i.e., $\vec{p}(t) = g(\vec{n}(t))$. Note that this model can characterize virtually any QoS routing proposals that select paths based on the current snapshot of the network. Alternatively, $\vec{p}(t)$ can be a function of some past history of network states $\vec{n}(s), s \in [t-d, t]$, where d is the length of the history information. The network can use the past history to predict the future, and use prediction to improve the routing decision. $\vec{p}(t)$ can also depend on the service time T of the incoming connection, if this information is available. The routing policy can then be written, in a most general form, as

$$\vec{p}(t) \quad = \quad g(\vec{n}(s), s \in [t-d, t]; T). \tag{1}$$

It can be shown that the system under any policy g will always converge to a stationary version, and the stationary version is ergodic [8].

[†] We note that for single-path flow control problems, Adaptive Virtual Queue (AVQ) algorithms have been used to construct primal solutions that compute the exact optimal operating point [21]. However, to the best of our knowledge, the generalization of the AVQ algorithms to multi-path problems has not been investigated and is beyond the scope of this paper.

Each admitted flow of class i will generate v_i amount of revenue per unit time. The dynamic routing scheme that maximizes the long term average revenue is then

$$J^* \triangleq \max_g \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \mathbf{E}_g \left[n_{ij}(t) \right] v_i,$$

where $\mathbf{E}_g$ denotes the expectation taken with respect to the stationary distribution under policy g.

Finally, in a *static scheme*, the routing policy is represented by a time-invariant vector $\vec{p}$. This corresponds to a proportional routing scheme. The performance of the static scheme is:

$$J_0 \triangleq \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij} v_i [1 - \mathbf{P}_{Loss,ij}],$$

where $\mathbf{P}_{Loss,ij}$ is the blocking probability experienced by flows of class i routed to path j.

B. Asymptotic Optimality of Static Schemes

The drawback of dynamic schemes is that the optimal schemes are difficult to find, and the implementation of these dynamic schemes will consume a large amount of computation and communication resources. It turns out that when the capacity of the system is large, *simple static schemes can approach the performance of the optimal dynamic scheme.* This has been the central theme of our earlier work [8]. Here, we rephrase the main result under the context of QoS routing. We scale the capacity and the demand proportionally by $c > 1$, i.e., in the c-scaled network, the capacity at each link l is $R^{l,c} = cR^l$, and the arrival rate of each class i is $\lambda_i^c = c\lambda_i$. In the rest of paper, when we refer to "high bandwidth networks" or "large-capacity networks," we mean that c is large. In other words, the link capacity is large compared to the bandwidth requirement of each user. The following result shows that when c is large[‡], a simple static scheme will suffice. The static scheme is constructed as follows:

Step 1: Solve the following optimization problem:

$$J_{ub} = \max_{\vec{p} \in \Omega} \quad \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{\theta(i)} p_{ij} v_i \tag{2}$$

$$\text{subject to} \quad \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \leq R^l \quad \text{for all } l,$$

where $\Omega = \bigotimes_{i=1}^{I} \Omega_i$.

Step 2: Use the optimal point $\vec{p}$ in (2) as the static policy. Let J_s be its performance.

The following proposition shows that the normalized revenue of the static scheme constructed above will approach that of the optimal dynamic scheme when $c \to \infty$.

Proposition 1: Let $J^{*,c}$ and J_s^c be the revenue of the optimal dynamic scheme and the revenue of the static scheme constructed above, respectively, in the c-scaled system, then

$$\lim_{c \to \infty} J_s^c / c = \lim_{c \to \infty} J^{*,c} / c = J_{ub}.$$

[‡]Note that here largeness does not imply over-provisioning.

Proposition 1 can be shown as in [8]. Here we sketch the main ideas of the proof. Firstly, one can show that cJ_{ub} is an upper bound of $J^{*,c}$ *under any dynamic routing policy* g [7], [9]. Secondly, the static revenue J_s^c differs from the upper bound cJ_{ub} only by the term $(1 - \mathbf{P}_{Loss,ij})$. Now since $\vec{p}$ satisfies the constraint of (2), the traffic load at each link is no greater than 1. Lemma 3 in [8] then ensures that the blocking probability goes to zero as $c \to \infty$. Finally, because $J_s^c \leq J^{*,c} \leq cJ_{ub}$, Proposition 1 then follows. The detailed proof is provided in our online technical report [27]. Readers can refer to [8] for a thorough treatment of the various simplicities that arise in the control of large-bandwidth networks.

III. The Optimization Based Approach to QoS Routing

There is a continuing trend to deploy routers with larger and larger link capacities in the Internet. Therefore, the results in the last section offer important insights on the QoS routing problem in the high-bandwidth networks of today and the future. Firstly, by solving a simple upper bound, we can obtain a simple *time-invariant* scheme that is *close to optimal*. Once we *precompute* the routing probabilities according to (2), the result can be used for a large number of future arrivals. Thus, *the computation overhead can be greatly reduced.* Secondly, the upper bound (2) replaces the instantaneous capacity constraint $\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} n_{ij} r_i H_{ij}^l \leq R^l$ by an average load constraint $\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \leq R^l$. Hence, the precomputation only needs to react to the *average congestion level* in the network rather than the *instantaneous congestion level*. The staleness of the link state information is no longer a major issue!

Therefore, if we are able to solve the upper bound (2) efficiently, we can obtain a QoS routing algorithm that is close to optimal in large networks and that can tolerate infrequent computation and infrequent link state updates. However, we still need to consider the following issues.

- The upper bound is a global optimization problem. A distributed solution is desired.
- Some parameters, such as λ_i and μ_i, could be unknown a priori and changing gradually over time. A solution is needed that can automatically adapt to these changes.

We next present an adaptive, distributed algorithm for solving the upper bound. Before we proceed, we note that in many scenarios, it is also desirable to modify the upper bound to improve fairness. We can view the upper bound (2) as a constrained optimization problem that maximizes some aggregate utility functions:

$$\max_{\vec{p} \in \Omega} \quad \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i\left(\sum_{j=1}^{\theta(i)} p_{ij}\right) v_i \tag{3}$$

$$\text{subject to} \quad \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \leq R^l \quad \text{for all } l,$$

where the utility function U_i is linear: $U_i(p) = p$. A linear utility function, however, does not possess good fairness properties: for example, connections with a larger number of hops could be completely blocked to give way to connections with fewer hops. To improve fairness, we can use a strictly concave utility function U_i, as in flow control problems [28]. The derivative $U_i'(\sum_{j=1}^{\theta(i)} p_{ij})$ represents the amount of marginal utility lost if the overall admission probability for class i is further reduced. The desired balance among different classes can be achieved by tuning the revenue v_i and the utility function U_i. Proposition 1 can be generalized to the case with concave utility functions [8], [27]. In this paper, we will use utility functions that satisfy $U_i'(1) = 1$. This choice of the utility function ensures that the revenue v_i is correctly reflected by the marginal utility when all flows of class i can be admitted, i.e., $v_i U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) = v_i$ when $\sum_{j=1}^{\theta(i)} p_{ij} = 1$. As long as the utility function follows this rule, our simulation results indicate that the revenue is usually not affected much by changing the utility functions.

A. A Distributed Algorithm

Let $\vec{p^*}$ be the maximizer of the modified upper bound (3). Because the objective function is concave and the constraint set is convex and compact, a maximizer always exists. However, it is generally not unique, since the objective function is not strictly concave. (Note that even if U_i is strictly concave, the overall problem is not, because of the linear operation $\sum_{j=1}^{\theta(i)} p_{ij}$.)

The form of the upper bound motivates us to study its dual. However, when the objective function of the primal problem is not strictly concave, the dual problem may not be differentiable. To circumvent this difficulty, we use ideas from Proximal Optimization Algorithms [29, Chapter 3.4.3]. The idea is to add a quadratic term to the objective function. We introduce an auxiliary variable y_{ij} for each p_{ij}. Let $\vec{y_i} = [y_{ij}, j = 1, ..., \theta(i)]$ and $\vec{y} = [\vec{y}_1, .., \vec{y}_I]$. The optimization becomes:

$$\begin{aligned} \max_{\vec{p}\in\Omega, \vec{y}\in\Omega} \quad & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i \\ & - \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \qquad (4) \\ \text{subject to} \quad & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \le R^l \quad \text{for all } l, \end{aligned}$$

where ν_i is some positive number chosen for each class i. For a fixed $\vec{y}$, the objective function in (4) is strictly concave. It is easy to show that the optimal value of (4) coincides with that of (3). In fact, if $\vec{p} = \vec{p^*}$ is the maximizer of (3), then $\vec{p} = \vec{p^*}, \vec{y} = \vec{p^*}$ is the maximizer of (4).

The standard Proximal Optimization Algorithm then proceeds as follows:

Algorithm $\mathcal{P}$:

At the t-th iteration,

- P1) Fix $\vec{y} = \vec{y}(t)$ and maximize the augmented objective function with respect to $\vec{p}$. To be precise, this step solves:

$$\begin{aligned} \max_{\vec{p}\in\Omega} \quad & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i \\ & - \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \qquad (5) \\ \text{subject to} \quad & \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \le R^l \quad \text{for all } l. \end{aligned}$$

  Since the objective function in (5) is now strictly concave, the maximizer exists and is unique. Let $\vec{p}(t)$ be the solution to this optimization.
- P2) Set $\vec{y}(t+1) = \vec{p}(t)$.

Step P1) can now be solved through its dual. Let $q^l, l = 1, ..., L$ be the Lagrange Multiplier for the constraints in (5). Let $\vec{q} = [q^1, ..., q^L]$. Define the Lagrangian as:

$$\begin{aligned} L(\vec{p}, \vec{q}, \vec{y}) = & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i \\ & - \sum_{l=1}^{L} q^l (\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l - R^l) \\ & - \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \\ = & \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} \left\{ U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} \sum_{l=1}^{L} H_{ij}^l q^l \right. \\ & \left. - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \right\} + \sum_{l=1}^{L} q^l R^l. \qquad (6) \end{aligned}$$

Let $q_{ij} = \sum_{l=1}^{L} H_{ij}^l q^l$, $\vec{q_i} = [q_{ij}, j = 1, ..., \theta(i)]$. The objective function of the dual problem is then:

$$D(\vec{q}, \vec{y}) = \max_{\vec{p}\in\Omega} L(\vec{p}, \vec{q}, \vec{y}) = \sum_{i=1}^{I} B_i(\vec{q_i}, \vec{y_i}) \frac{\lambda_i}{\mu_i} + \sum_{l=1}^{L} q^l R^l, \quad (7)$$

where

$$\begin{aligned} B_i(\vec{q_i}, \vec{y_i}) \quad = \quad \max_{\vec{p_i}\in\Omega_i} & \left\{ U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q_{ij} \right. \\ & \left. - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \right\}. \qquad (8) \end{aligned}$$

Note that in the definition of the dual objective function $D(\vec{q}, \vec{y})$ in (7), we have decomposed the original problem into I separate subproblems. Given $\vec{q}$, each class can solve the routing probabilities $\vec{p_i}$ via its local subproblem (8) independently. If we interpret q^l as the implicit cost per unit bandwidth at link l, then q_{ij} is the total cost per unit bandwidth for all links in the path j of class i. Thus the q_{ij} captures all the information each subproblem needs about the path class i traverses. We note that an important feature of this decomposition is that the subproblem (8) is independent of the parameters λ_i and μ_i. This makes online implementation particularly easy.

The dual problem of (5), given $\vec{y}$, is:

$$\min_{\vec{q}\geq 0} D(\vec{q},\vec{y}).$$

Since the objective function of the primal problem (5) is strictly concave, the dual is always differentiable. The gradient of D is

$$\frac{\partial D}{\partial q^l} = R^l - \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij}^0 r_i H_{ij}^l, \tag{9}$$

where p_{ij}^0 solves the local subproblem (8). Then step P1) can be solved by using the gradient descent iteration on the dual variable, i.e.,

$$q^l(t+1) = \left[q^l(t) - \alpha^l (R^l - \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} p_{ij}^0 r_i H_{ij}^l) \right]^+, \tag{10}$$

where $[.]^+$ denotes the projection to $[0,+\infty)$.

The class of distributed algorithms we will use in this paper can be summarized as follows:

Algorithm $\mathcal{A}$:

- A1) Fix $\vec{y}(t)$ and use the gradient descent iteration (10) on the dual variable $\vec{q}$. Depending on the number of times the descent iteration is executed, we will obtain a dual variable $\vec{q}(t+1)$ that either exactly or approximately minimizes $D(\vec{q},\vec{y}(t))$ (and, equivalently, solves (5)). Let K be the number of times the dual descent iteration is executed.
- A2) Let $\vec{p}(t)$ be the primal variable that maximizes, over all $\vec{p} \in \Omega$, the Lagrangian $L(\vec{p},\vec{q}(t+1),\vec{y}(t))$ corresponding to the new dual variable $\vec{q}(t+1)$. Set $\vec{y}(t+1) = \vec{p}(t)$.

From now on, we will refer to (10) as the *dual update*, and step A2) as the *primal update*.

A stationary point of algorithm $\mathcal{A}$ can be defined as a primal-dual pair $(\vec{y^*},\vec{q^*})$ such that

$$\vec{y^*} = \operatorname*{argmax}_{\vec{p}\in\Omega} L(\vec{p},\vec{q^*},\vec{y^*}),$$

$$q^{l,*} \geq 0 \text{ and } \sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} y_{ij}^* r_i H_{ij}^l \leq R^l \text{ for all } l, \text{ and}$$

$$q^{l,*}\left(\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} y_{ij}^* r_i H_{ij}^l - R^l \right) = 0 \text{ for all } l.$$

By standard duality theory, any stationary point $(\vec{y^*},\vec{q^*})$ of the algorithm $\mathcal{A}$ solves the augmented problem (4). Hence $\vec{p} = \vec{y^*}$ solves the upper bound (3).

An important question is how large K (in step A1) needs to be for algorithm $\mathcal{A}$ to converge to a stationary point. The standard proximal optimization theory [29, Chapter 3.4.3] requires $K=\infty$, i.e., at each iteration the optimization (5) has to be solved exactly. This requirement essentially corresponds to a time-scale separation between the time-scale of the primal updates and that of the dual updates. When $K<\infty$, at best an approximate solution to (5) is obtained at each iteration. If the accuracy of the approximate solution can be controlled appropriately (see [30]), one can still show the convergence of algorithm $\mathcal{A}$. However, in this case the number of dual updates K has to depend on the required accuracy and usually needs to be large.

For online implementation, one cannot carry out the dual update infinitely many times *for one iteration of algorithm $\mathcal{A}$*. It is also difficult to distributively control the accuracy of the approximate solution to (5). Hence, in this work we use a different approach. The following result is new and shows that, by appropriately choosing the stepsize α^l, the algorithm $\mathcal{A}$ converges *for any choice of $K \geq 1$*. No time-scale separation is needed! The proof is highly technical, and due to space constraints, we only include a sketch of the proof in Appendix B. Interested readers are referred to our technical report [27]. The main idea of the proof has also been reported in [31].

Proposition 2: Fix $1 \leq K \leq \infty$. As long as the stepsize α^l is small enough, the algorithm $\mathcal{A}$ will converge to a stationary point $(\vec{y^*},\vec{q^*})$ of the algorithm, and $\vec{p^*} = \vec{y^*}$ solves the upper bound (3). The sufficient condition for convergence is:

$$\max_l \alpha^l < \begin{cases} \frac{2}{\mathcal{S}\mathcal{L}} \min_i \frac{\mu_i \nu_i v_i}{\lambda_i r_i^2} & \text{if } K=\infty \\ \frac{1}{2\mathcal{S}\mathcal{L}} \min_i \frac{\mu_i \nu_i v_i}{\lambda_i r_i^2} & \text{if } K=1 \\ \frac{4}{5K(K+1)\mathcal{S}\mathcal{L}} \min_i \frac{\mu_i \nu_i v_i}{\lambda_i r_i^2} & \text{if } K>1 \end{cases},$$

where $\mathcal{L} = \max\{\sum_{l=1}^{L} H_{ij}^l, i=1,...,I, j=1,...\theta(i)\}$ is the maximum number of hops for any path, and $\mathcal{S} = \max\{\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)} H_{ij}^l, l=1,...,L\}$ is the maximum number of paths going through any link.

Remark: The sufficient condition for $K=1$ differs from that of $K=\infty$ only by a constant factor. For $K>1$, our result requires that the stepsizes decrease on the order of $O(1/k^2)$. This is probably not the tightest possible result, and we conjecture that stepsizes of order $O(1)$ would work for any K. However, we leave this for future work. Note also that ν_i appears on the right hand side of the condition. Hence, by making the objective function more concave, we also relax the requirement on the stepsize α^l. Finally, Proposition 2 does not require the routing matrix $[H_{ij}^l]$ to be of full rank.

B. Distributed Implementation

Algorithm $\mathcal{A}$ lends naturally to online distributed implementation. The ingress router for each class is responsible for determining the routing probabilities for this class. To do so, the ingress router only needs to solve the local subproblem (8) using the implicit costs q^l at all core routers that class i traverses. An efficient algorithm can solve (8) in at most $O[\theta(i)\log\theta(i)]$ steps (see Appendix A). The core routers bear the responsibility to update the implicit costs q^l according to the simple dual update rule (10). After every K dual updates, the ingress router executes the primal update.

We have mentioned earlier that the solution of each local subproblem (8) does not require knowledge of the demand parameters λ_i and μ_i. Next, we show that the dual update can also be carried out using online measurement at each link, again without prior knowledge of the demand parameters of each class. We then obtain an *adaptive* algorithm that can track changes in the network conditions.

Note that in the dual gradient (9), $\sum_{i=1}^{I}\sum_{j=1}^{\theta(i)}\frac{\lambda_i}{\mu_i}r_i p_{ij}H_{ij}^l$ is the average load per unit time at link l. This motivates us to estimate the gradient as follows: over a certain time window W, each link l collects the information of flow connection requests *from all classes* that arrive at the link. Let w be the total number of flow arrivals during W. Let $r_k, T_k, k = 1, ...w$ denote the bandwidth requirement and the service time, respectively, of the k-th arrival. (This information can be carried along with the connection requests.) Then we can use

$$G_t = R^l - \frac{\sum_{k=1}^{w} r_k T_k}{W} \tag{11}$$

to estimate the gradient. The interpretation is immediate: $\sum_{k=1}^{w} r_k T_k$ is the total amount of load brought to link l. One can verify that this estimate is unbiased, i.e., $\mathbf{E}[G_t] = \partial D/\partial q^l$. We can then update the implicit costs by

$$q^l(t+1) = \left[q^l(t) + \alpha^l\left(\frac{\sum_{k=1}^{w} r_k T_k}{W} - R^l\right)\right]^+ \tag{12}$$

When W is not large, the stepsize α^l has to be small to "average out" the noise in the estimate. This algorithm has the flavor of *stochastic approximation algorithms* [32] that have been used in many engineering problems. Our simulations with this algorithm demonstrate good convergence properties when a small fixed stepsize is used. That is, according to the simulations, the stochastic approximation algorithm converges to a small neighborhood of the solution to the upper bound. Further, when the stepsize α^l is away from zero, our algorithm can track the nonstationary behavior of the network. As the demand (i.e., λ_i, μ_i) changes, it is reflected in the gradient estimate G_t. The network will then move towards the new optimal operating point.

C. How to Generate Alternate Paths

The set of alternate paths, denoted by the matrix $[H_{ij}^l]$, could potentially be the enumeration of all possible paths for each class. In practice, however, a much smaller set of alternate paths suffices. Maintaining this set of alternate paths is the role of the path-finding component in Fig. 1. There are several options to generate the candidate paths.

Option 1: Use paths that appear to be "heuristically good." For example, given a source-destination pair, we can use the set of minimum-hop paths, or, paths whose number of hops is no greater than h plus that of the minimum-hop path. Obviously, h should be small to avoid an explosion in the number of candidate paths.

Option 2: A better approach is to discover new paths online. The implicit costs q^l, which arise naturally as the Lagrangian Multipliers of the dual problem, give us guidelines on discovering potentially better alternate paths. Given a configuration of the alternate paths, we can easily verify the following properties that characterize any stationary point $(\vec{p^*}, \vec{q^*})$ of algorithm $\mathcal{A}$ (for details, see [27]): (1) when the utility functions are strictly concave, the admission probability $\sum_{j=1}^{\theta(i)} p_{ij}^*$ for each class i can be uniquely determined; (2) only paths that have the minimum cost see positive routing probabilities. The cost of a path is the sum of the implicit costs for all links along the path. Hence, if we let $q_{i,0}$ denote the minimum cost among all alternate paths for class i, i.e., $q_{i,0} \triangleq \min_j \sum_{l=1}^{L} H_{ij}^l q^{l,*}$, then for all j,

$$p_{ij}^* > 0 \Rightarrow q_{ij}^* = q_{i,0}.$$

This is consistent with the concept of the *minimum first derivative path* discussed in [29, p417]. Therefore, adding paths whose costs are larger than the minimum cost will not yield any gain.

We can use the above properties to iteratively generate the candidate paths online. Starting from any initial set of candidate paths, we execute the distributed algorithm $\mathcal{A}$ to solve the upper bound. Then based on the implicit costs at the (possibly approximate) stationary point, we can run *any* minimal cost routing algorithm using the implicit costs as the cost metric for each link. If the minimal cost is smaller than the minimal cost among the current set of candidate paths by a certain threshold, we add this new path into the set, and continue. Otherwise, we can conclude that no further alternate paths need to be added.

Option 3: Use historical data. This can be viewed as a traffic engineering step. We first take measurements of typical traffic demands at different times of the day. For each demand pattern, we can use the above procedure in Option 2 *offline* to find the optimal alternate paths. The union of the alternate paths under all demand patterns can then be used as the set of candidate paths. The role of the distributed algorithm $\mathcal{A}$ is to shift the traffic load among these candidate paths automatically as the network condition changes.

D. Extensions to Multiple QoS Constraints

So far we have assumed that the bandwidth constraint is the only QoS constraint. We now address the extension to multiple QoS metrics and constraints. We can argue that link-state metrics other than the available bandwidth (e.g., delay and overflow probabilities, etc.) could be more stable in future high-bandwidth networks. When the link capacity of the network is large, the network can support a large number of connections at the same time. Due to the complexity in maintaining per-flow information, Quality of Service is likely to be provisioned on an aggregate basis. Each node in the network will provide a QoS guarantee on delay and/or packet loss probabilities *for all flows* belonging to the same class, rather than *for each individual flow*. Such guarantees will stay unchanged as new flows arrive at or old flows depart from the network.

Let each class be given some QoS requirements on *both* the bandwidth constraint *and* some other constraints such as delay or packet loss probabilities. We now assume that each link will provision certain QoS guarantees on these other QoS metrics. Such guarantees are constant over time and can be advertised to the entire network. The alternate paths for each class must now be constrained to those that satisfy these other QoS requirements. Given a set of alternate paths, the distributed algorithm in Section III-A can be used unchanged to find the optimal routing probabilities. In order to generate the alternate paths, we can use the options in Section III-C, except that now we have to consider other constraints too. For example, in Option 2, we can still use the implicit cost as the cost metric for each link and execute any *constrained minimal cost* QoS routing algorithm to search for new alternate paths.

It is important to note that the path-finding step does not deal with the available bandwidth constraint directly. Instead, it is based on the implicit cost, which is a more stable parameter that depends on the average congestion level of the network. Hence, the path-finding step can be carried out infrequently. Note that the computation of optimal paths under multiple QoS constraints is usually a NP-complete problem. Hence, for any practical implementation of QoS routing solutions, the computation overhead has always been a key issue. Our optimization based approach does not directly reduce the computational complexity. Rather, it reduces the *frequency* of the computation. We emphasize that *the optimal performance is still preserved even though computation becomes infrequent*. This, as mentioned in the Introduction, is again due to the separation of control time-scales: the set of candidate paths needs to change only when the *average* demand and capacity of the network changes significantly. Hence, the intensive computations only need to be carried out infrequently.

IV. Implementational Issues

In this section we address some implementational issues.

A. Communicating the Implicit Costs

The distributed algorithm requires communicating the implicit costs back to the ingress routers. There are two alternatives. One is to use the connection request packets sent by the ingress router. Each link can insert its own implicit costs when processing the connection request packets. When the response is sent back to the ingress router, the implicit costs are piggy-backed for free. The other approach is to periodically advertise the implicit costs throughout the network. In the latter case, even when the implicit costs are updated infrequently, while the speed of convergence of the distributed algorithm will be affected, the optimal routing probabilities that the algorithm converges to will remain the same.

B. Gradient Estimates at the Link Algorithms

For the link algorithm, the gradient estimate in (11) requires the information from all flow arrivals, including those that could have been rejected by the upstream links. In some network systems, once an intermediate link along the path rejects a connection request, the request will not be passed on to downstream links. Let $\mathbf{P}_{i,j}^{B,l}$ be the probability that a connection request of class i routed to path j is rejected by links that are upstream to link l. The true connection arrival rate of class i at a link l will be $\lambda_i p_{ij}(1-\mathbf{P}_{i,j}^{B,l})$. In this case, the gradient estimate constructed in (11), by counting only actual arrivals, will be biased. However, when the system is large, this error will be small. This is due to two factors: (1) as long as the load at each link is less than or equal to 1, $\mathbf{P}_{i,j}^{B,l}$ will be close to zero (see Lemma 3 in [8]); (2) if some links have load greater than 1, the implicit costs at these links will be increased until the loads become less than or equal to 1. Therefore, in the end $\mathbf{P}_{i,j}^{B,l}$ will be close to zero and will have a minimal impact on the gradient estimate.

The gradient estimate in (11) needs knowledge of the service time T_k of an incoming flow. When this information is not known at the time of connection arrival, it can also be replaced by the time average of the service time of past flows. This time average can be calculated at the ingress router by measuring the service time of the flows that have completed service. The unbiasness of (11) is not affected by such changes.

C. Adaptive Stepsizes

The transient behavior of the distributed algorithm is sensitive to the choice of the stepsize α^l. A smaller stepsize will result in a smaller *misadjustment* (overshoot or undershoot) around the optimal solution, but takes a longer time to converge. A larger stepsize expedites the convergence at the cost of larger misadjustment. This tradeoff between misadjustment and speed of convergence is a fundamental one for stochastic approximation algorithms with constant stepsizes. A better approach is to use an adaptive stepsize scheme: a larger stepsize is used initially (or when sudden changes occur) to expedite convergence, followed by a smaller stepsize to reduce the misadjustment. This idea of stepsize adaptation has been used in many other applications, especially in adaptive filtering. Here we illustrate one such approach, borrowed from the idea in [33]:

Fix a link l. Let G_t be the estimate of the gradient at the t-th iteration. Let E_t be a weighted average of the past samples of G_t, i.e., upon a new sample G_t, let

$$E_{t+1} = \epsilon^l G_t + (1-\epsilon^l)E_t,$$

where ϵ^l is a small positive constant. Let α_t^l denote the stepsizes at the t-th iteration. We can update the stepsize based on the correlation between E_t and G_t, i.e.,

$$\alpha_{t+1}^l = \min\{[\alpha_t^l + \beta^l E_t G_t]^+, \alpha_{\max}\}, \tag{13}$$

where β^l is a small positive constant, and $\alpha_{\max}$ is a maximum allowable stepsize chosen to ensure the stability of the system. We will demonstrate in the simulation results in Section V that the distributed algorithm with such an adaptive stepsize scheme can swiftly track the changes in the network condition, and it can also effectively reduce the misadjustment when the network condition is stable.

V. Simulation Results

In this section, we present simulation results that illustrate our optimization based approach for QoS routing. We use an event-driven flow-level simulator written in C++. Our simulator can simulate different flow arrival patterns and holding-time distributions. We can also simulate different strategies for updating the information across the network, e.g., changes in link implicit costs can either be updated at the source nodes immediately (i.e., synchronous updates), or

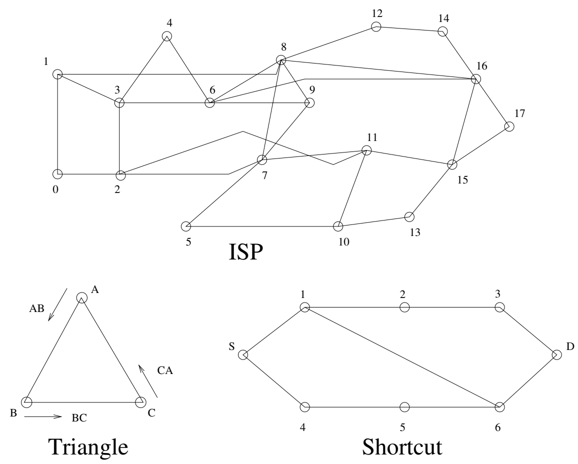


Fig. 2. Network Topologies

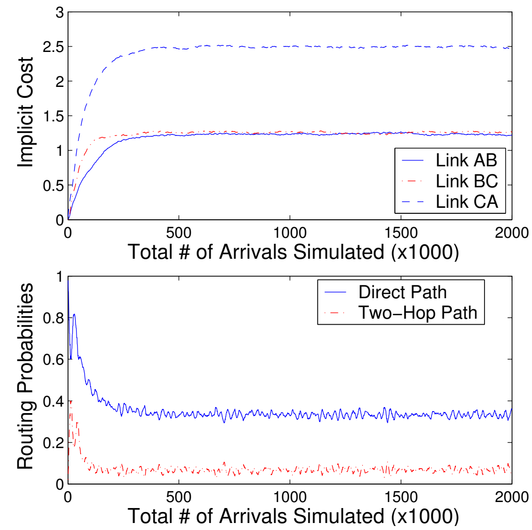


Fig. 3. Evolution of the implicit costs (top) and the routing probabilities of class CA (bottom) with respect to the number of arrivals simulated. The unit of x-axis is 1000 arrivals. The solution to the upper bound is the following: the implicit costs are 1.25, 1.25, and 2.5, respectively, for link AB, BC and CA. The routing probability for class CA are 0.33 for the direct path and 0.067 for the two-hop path.

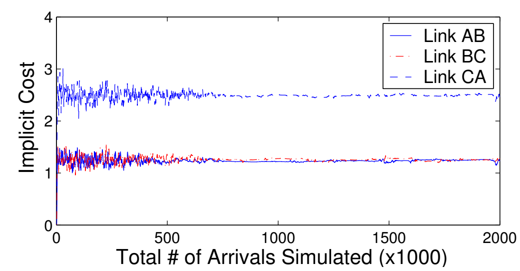


Fig. 4. Evolution of the implicit costs when the adaptive stepsize scheme is used.

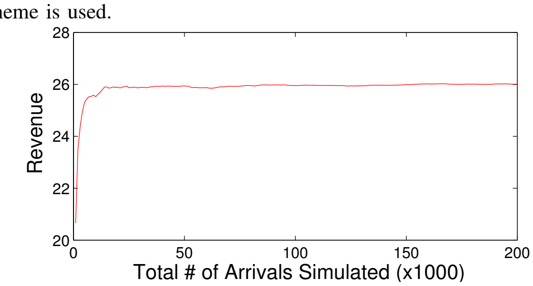


Fig. 5. Evolution of the average revenue when the adaptive stepsize scheme is used.

they can be updated after an arbitrary delay (i.e., asynchronous updates). We implement the distributed algorithm following the online measurement based scheme in Section III-B. Note that although the convergence of the algorithm is shown assuming global synchronized updates, in our simulation the local subproblem (8) is solved only when a new call arrives. Hence, the algorithm is executed in an asynchronous fashion.

The topologies that we use are shown in Fig. 2. We first demonstrate the convergence of the distributed algorithm using the "triangle" network in Fig. 2. There are three classes of flows (AB, BC, CA). For each class of flows, there are two alternate paths, i.e., a direct one-link path, and an indirect two-link path. The arrival rates for classes AB, BC, CA are 1, 1 and 3 *flows per time unit*, respectively. Each flow consumes one *bandwidth unit* along the path(s) and holds the resources for a time that is exponentially distributed with mean of 100 units. Let the capacity of all links be 100 *bandwidth units*. For all classes the revenue v_i is 1 and the utility function is $U_i(p) = \ln p$. Both the revenue and the implicit cost are chosen to be unitless.

Fig. 3 demonstrates the evolution over time of the implicit costs at all links and the evolution of the routing probabilities of class CA. The x-axis corresponds to the total number of arrivals simulated. Readers can verify that all quantities of interest converge to a small neighborhood of the solution to the upper bound. The parameters we use for the distributed algorithm are: $\alpha^l = 0.0001$ *per bandwidth unit*, $\nu_i = 1$, $K = 1000$ and $W = 1$ *time unit*.

Fig. 4 demonstrates the convergence of the implicit costs when we use the adaptive stepsize scheme in Section IV. The parameters we use are: $\epsilon^l = 0.001$, $\alpha_{\max} = 0.1$ *per bandwidth unit*, $\beta^l = 0.0001$ *per cubic bandwidth unit* and $\alpha_0^l = 0$. The initial convergence is almost immediate: the implicit costs quickly jump to a small neighborhood of the solution to the upper bound, thanks to an increase in the stepsize initially. The evolution of the routing probabilities (not shown) follows the same trend. While the misadjustment takes time to die out (as the stepsize becomes smaller), Fig. 5 shows that the convergence of the revenue to its stationary value is achieved must faster (note that the range on the x-axis is smaller). As far as the overall revenue is concerned, the fluctuations of the implicit costs appear to cancel themselves out.

We have also simulated the case when the network condition changes over time, i.e., when the system is non-stationary. Fig. 6 and Fig. 7 demonstrate the evolution of the implicit costs when the average inter-arrival time of class CA changes according to a square wave and a triangle wave, respectively. We observe that the distributed algorithm with adaptive stepsizes can track the changes in the network condition swiftly.

We next simulate a larger network, i.e., the "ISP" topology in Fig. 2, which is reconstructed from an ISP network and has

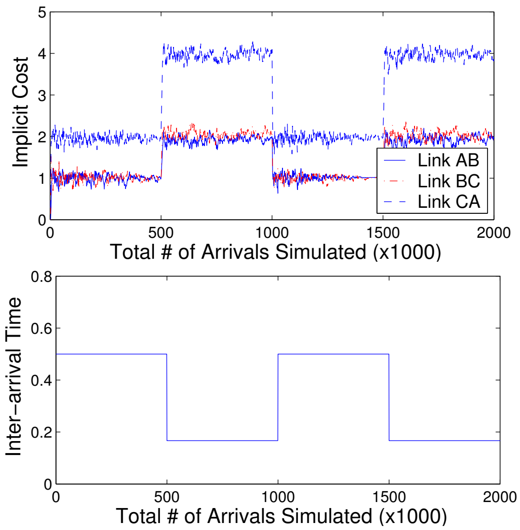


Fig. 6. Evolution of the implicit costs (top) when the average inter-arrival time of class CA changes according to a square wave (bottom).

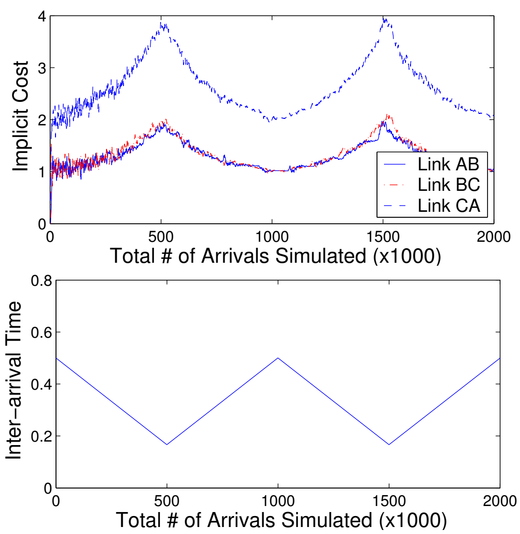


Fig. 7. Evolution of the implicit costs (top) when the average inter-arrival time of class CA changes according to a triangle wave (bottom).

been used in many simulation studies [3], [4], [5], [6], [14]. It has 18 nodes and 30 links. We simulate the case with a uniform demand matrix: flows arrive at each node according to a Poisson process with rate λ, and the destinations are chosen uniformly among all other nodes. The bandwidth requirement of each connection is one *bandwidth unit*. Revenue v_i is 1. We use a Pareto service time distribution with shape parameter 2.5, to capture the heavy-tailed characteristic of the traffics on the Internet. The mean service time is 100 *time units*. The capacity of each link is 1000 *bandwidth units*.

There are a total of $18 \times 17 = 306$ source-destination pairs (i.e., classes). When the simulation is initialized, the set of alternate paths for each source-destination pair consists of all minimum-hop paths. Once simulation starts, new paths can be added following Option 2 in Section III-C. To simplify the simulation, we adopt an upper limit of 10 on the number of alternate paths for each source-destination pair: when a new path is found, if there are already 10 alternate paths, the old path with the smallest routing probability will be replaced by the new path.

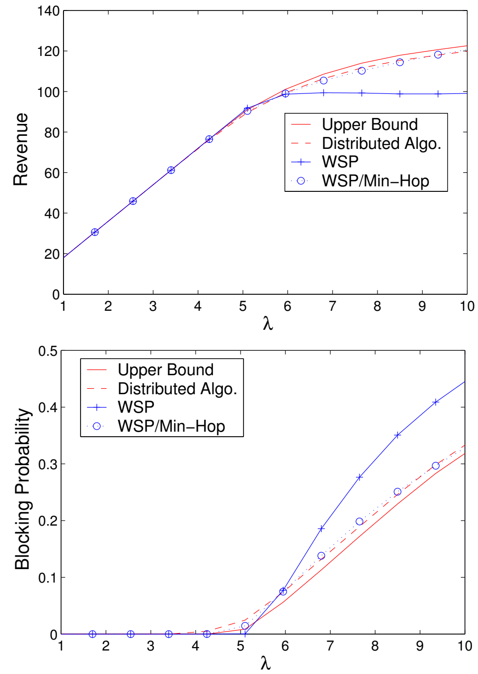


Fig. 8. The revenue (top) and the blocking probability (bottom) of the distributed algorithm compared with the upper bound and WSP.

We choose the utility function to be of the following form

$$U_i(p) = h_i \ln p - (h_i - 1)p,$$

where h_i is the minimal number of hops between source-destination pair i. This utility function improves the admission probability for flows that traverse a larger number of hops. (At the same level of admission probability $p < 1$, the marginal utility $\frac{dU_i}{dp} = h_i/p - (h_i - 1)$ is larger for flows that traverse a long path.)

We simulate the optimization based approach using the distributed algorithm and compare, in Fig. 8, the revenue and the total blocking probability over all classes against the values determined by the upper bound. We vary the per-node flow arrival rate λ from 1.0 to 10.0 *flows per time unit*. As we can see from these figures, our distributed algorithm tracks the upper bound consistently over all loads. With a network of this size (each link can hold 1000 flows) the difference between the upper bound and the simulation of our distributed algorithm is already small.

We also compare the performance of the Widest-Shortest-Path (WSP) algorithm. WSP has been used in many simulation studies [3], [4], [14]. Among all feasible paths, the WSP algorithm will first choose paths that have the smallest number of hops. If there are multiple such paths, the WSP algorithm will choose the one with the largest available bandwidth. However, as shown in Fig. 8, the performance of a faithful implementation of WSP starts to taper off at $\lambda = 5.0$ *flows per time unit*. The performance degradation of WSP is due to its selection of non-minimal hop paths, which could result in sub-optimal configurations for the whole network. If we constrain WSP to minimum-hop paths only, the performance degradation will disappear in this example, as shown by the curve labeled "WSP/Min-Hop." However, from this, we should not draw the conclusion that such a practice is always better. By constraining WSP to minimum-hop paths, one also reduces the capability of WSP to use other potentially less congested paths. The end result depends on the topology of the network and the demand pattern. For example, in the "shortcut" topology in Fig. 2, assume that the capacities of all links are the same. If flows from S to D is to only use the minimum-hop path (S-1-6-D), once this path is full, no more flows can be admitted. However, if the flows use the non-minimum-hop paths S-1-2-3-D and S-4-5-6-D, twice as many flows can be admitted. Hence it is not always better to restrict on minimum-hop paths.

Our distributed algorithm, on the other hand, will always be able to find the right balance by solving the upper bound. It *consistently* tracks the upper bound under all load conditions. This provable optimality is an attractive feature of our optimization based approach as it ensures that the routing decision will always be close to optimal.

The strength of the optimization based approach is even more evident when the computation and link state updates become infrequent. To show this, we pick $\lambda = 6.0$ *flows per time unit* and simulate both the distributed algorithm and the WSP (with minimum-hop path only) when we vary the interval between link-state updates. To ensure a fair comparison of the performance and the overhead of the two schemes, we adopt the following settings for our simulation. For the distributed algorithm, we choose to simulate the case where the implicit costs are advertised with each link state update, and computation is carried out after each link state update. (That is, we are not simulating the "piggy-back" approach in Section IV-A.) For WSP, in contrast to the suggestion given in [5], we do not allow WSP to *recompute* paths when a connection routed to a precomputed path is later rejected. The reason is that one cannot reduce the computational overhead too much if such recomputation is allowed: for example, when the blocking probability is around 10%, on average 1 out of 10 arrivals will trigger recomputation! For a similar reason, we also do not use the *triggered* link state update strategy of [5] for WSP. When the triggered strategy is used, changes in available bandwidth that exceed certain percentage of the past advertised available bandwidth will trigger a new link state update. When the network operates at a high utilization level, the available bandwidth is small. Even small changes in available bandwidth will trigger frequent updates. Hence, one can not reduce the communication overhead too much using a triggered update strategy.

Simulation results are presented in Fig. 9. The performance of the distributed algorithm changes little as the link state update interval becomes larger and larger, while the performance of WSP decreases significantly. (The unit time on the x-axis is the mean inter-arrival time of flows at each node.) In the worst case, WSP blocks *twice as many connections* compared to the case when it has perfect link states. We have also simulated the case when the network condition changes over time (i.e., when the system is non-stationary). In Fig. 10, we change the average inter-arrival time at each node according to the triangle wave in Fig. 10(b), and plot the overall blocking probability in Fig. 10(a) when we vary the interval between link-state updates. The performance of the distributed algorithm is again insensitive to the link state update interval, while the performance of WSP decreases significantly as the link state update interval increases. Note that the exact level of this performance degradation for WSP is a complex function that depends on many factors, such as the topology and the demand of the network, etc. Again, the strength of the optimization based approach is that it consistently achieves near optimal performance, even when the computation and communication overhead are greatly reduced.

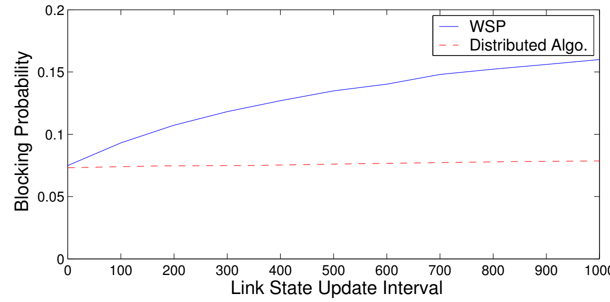
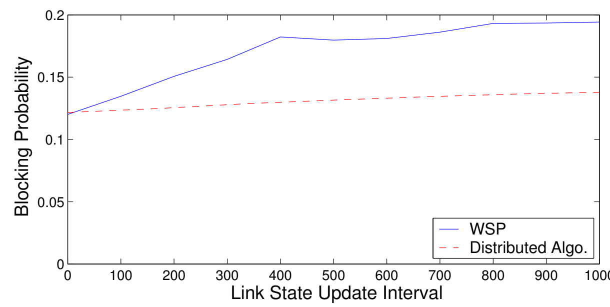
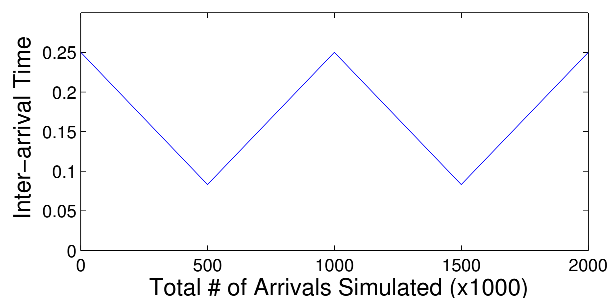


Fig. 9. The blocking probability as the link state update interval increases. The unit on the x-axis is the mean inter-arrival time of flows at each node. The arrival rate at each node is fixed at $\lambda = 6.0$ *flows per unit time*.



(a)



(b)

Fig. 10. The blocking probability versus the link state update interval (top) when the average inter-arrival time at each node changes according to the triangle wave (bottom). The unit on the x-axis of Figure (a) is the mean inter-arrival time of flows at each node. The unit on the x-axis of Figure (b) corresponds to 1000 total arrivals simulated. The average arrival rate at each node is $\lambda = 8.0$ *flows per unit time*.

When our optimization based approach to QoS routing is used, designers can predict the operating point of the network by analytically solving the upper bound. This is shown in

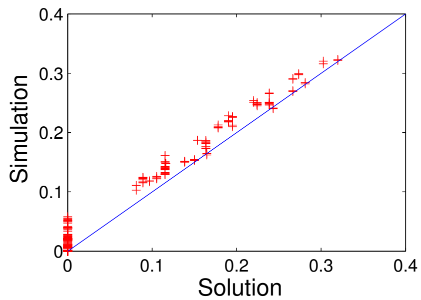


Fig. 11. The blocking probability predicted by the upper bound compared with that collected from the simulation of the distributed algorithm. The arrival rate at each node is fixed at $\lambda = 6.0$ *flows per time unit*.

Fig. 11 where each point represents the blocking probability of one source-destination pair computed by the upper bound (along the x-axis) and that collected from the simulation of the distributed algorithm (along the y-axis). The points follow the diagonal line, which indicates that the simulation matchs the theory. In contrast, the analysis of dynamic QoS routing schemes (such as WSP) appears to be an intractable problem, especially when the computation becomes infrequent and the link state information becomes inaccurate. One usually has to resort to simulation to find out the operation of a QoS routing algorithm.

VI. Conclusion and Future Work

In this paper, we developed an optimization based approach for Quality of Service routing in high-bandwidth networks. We view a network that employs QoS routing as an entity that carries out a distributed optimization. By solving the optimization problem, the network is driven to an efficient operating point. When the capacity of the network is large, this optimization takes on a simple form. We develop a distributed and adaptive algorithm that can efficiently solve the optimization online. The proposed optimization based approach has several advantages in reducing the computation and communication overhead, and in improving the predictability and controllability of the operating characteristics of the network.

We now briefly outline directions for future work: (1) In this paper we propose to update the implicit costs by measuring the arrived load. Other methods are possible, for example, by taking into account the utilization levels of the links. (2) A deeper understanding of the transient behavior of the distributed algorithm is important. The adaptive stepsize scheme in Section IV that improves the speed of the convergence is of particular interest. (3) We assume that the capacity of the network is uniformly large. If some part of the network is not so large (for example, at the network edge), one then has to study a finer level of dynamics in these parts of the network. It would be interesting to study hybrid schemes that combine our results with some further details of the dynamics of smaller links. (4) In this paper we take a source routing model. Adapting our result to the distributed routing or hierarchical routing paradigms is also a possible direction for future work. A related issue is how to deal with the case when routers do not allow arbitrary splitting of traffic among multiple paths. (5) Finally, from a theoretical viewpoint, it would be important to prove the convergence of the distributed algorithm under more general settings, such as with asynchronous computation.

Appendix

A. An Efficient Algorithm for Solving the Local Subproblem (8)

Given the implicit costs $\vec{q}$, each class i solves its local subproblem (8) to obtain the routing probabilities. Recall that the local subproblem is:

$$B_i(\vec{q}_i, \vec{y}_i) = \max_{\vec{p}_i \in \Omega_i} \left\{ U_i(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i \sum_{j=1}^{\theta(i)} p_{ij} q_{ij} - \sum_{j=1}^{\theta(i)} \frac{\nu_i}{2} (p_{ij} - y_{ij})^2 v_i \right\}, \tag{14}$$

where

$$\Omega_i \triangleq \{p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, \text{ for all } j\}.$$

Let L_j be the Lagrangian multiplier for the constraint $p_{ij} \geq 0$, and let L_0 be the Lagrangian multiplier for the constraint $\sum_{j=1}^{\theta(i)} p_{ij} \leq 1$. Then, the Karush-Kuhn-Tucker condition becomes:

$$\begin{aligned} & L_0 \geq 0, L_j \geq 0, j = 1, ..., \theta(i), \\ & p_{ij} \geq 0, \sum_{j=1}^{\theta(i)} p_{ij} \leq 1, j = 1, ..., \theta(i), \\ & L_j p_{ij} = 0, L_0(\sum_{j=1}^{\theta(i)} p_{ij} - 1) = 0, j = 1, ..., \theta(i), \\ & U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - r_i q_{ij} - \nu_i (p_{ij} - y_{ij}) v_i + L_j - L_0 = 0. \end{aligned}$$

Let $Q_{ij} = \nu_i y_{ij} v_i - r_i q_{ij}$. The last equation becomes:

$$U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - \nu_i v_i p_{ij} + Q_{ij} + L_j - L_0 = 0. \tag{15}$$

Without loss of generality, assume that the alternate paths are ordered such that $Q_{i,1} \geq Q_{i,2} \geq ... \geq Q_{i,\theta(i)}$. Then it is easy to show that $p_{i,1} \geq p_{i,2} \geq ... \geq p_{i,\theta(i)}$ (see [27] for the details). Therefore, there must exist an integer J such that:

$$p_{ij} > 0 \text{ for any } j \leq J, \text{ and } p_{ij} = 0 \text{ for any } j > J.$$

Note that this number J is important because once J is known, the routing probabilities p_{ij} can be easily found. To see this, sum (15) for all $j \leq J$. Since $L_j = 0$ for all $j \leq J$, we have,

$$J U_i'(\sum_{j=1}^{\theta(i)} p_{ij}) v_i - \nu_i v_i \sum_{j=1}^{\theta(i)} p_{ij} + \sum_{j=1}^{J} Q_{ij} - J L_0 = 0.$$

Let $f(x) = J U_i'(x) v_i - \nu_i v_i x$. We first find the value of $\sum_{j=1}^{\theta(i)} p_{ij}$, which is equivalent to solving x and L_0 such that

$$f(x) + \sum_{j=1}^{J} Q_{ij} - J L_0 = 0, \tag{16}$$

where either $x = 1$ and $L_0 \geq 0$, or $L_0 = 0$ and $0 \leq x \leq 1$. Note that $f(x)$ is decreasing in x due the concavity of U_i. If

$$f(1) + \sum_{j=1}^{J} Q_{ij} < 0,$$

then the solution to (16) should be some $x < 1$ with $L_0 = 0$, in which case x is the solution of $f(x) + \sum_{j=1}^{J} Q_{ij} = 0$. Otherwise, x should be equal to 1, and $L_0 = \frac{f(1)+\sum_{j=1}^{J} Q_{ij}}{J}$. In both cases, we can find the values of $x = \sum_{j=1}^{\theta(i)} p_{ij}$ and L_0 easily. Once these values are found, we can solve the routing probabilities p_{ij} via (15), i.e.,

$$p_{ij} = \begin{cases} \frac{U_i'(x)v_i + Q_{ij} - L_0}{\nu_i v_i} & \text{if } j \leq J \\ 0 & \text{if } j > J \end{cases} . \quad (17)$$

We have just shown that, once the number J is know, the routing probabilities p_{ij} can be easily computed. It remains to find the correct value of J. We use a linear search for finding J. We start the search by assuming $J = \theta(i)$. We then verify whether the current value of J is correct by solving p_{ij} via the procedure described earlier. If the values of p_{ij} are all non-negative, then J is correct. In fact, since the solution for p_{ij} computed in (17) is decreasing in j, we only need to ensure that $p_{i,J}$ is non-negative. On the other hand, if the verification fails, we reduce J by 1, and verify again; until either a correct value of J is found, or $J = 0$ and hence all p_{ij} should be zero.

We summarize below the algorithm for solving the subproblem (8):

1) Sort the index j such that Q_{ij} is in decreasing order.
2) Let $J = \theta(i)$ and $Q = \sum_{j=1}^{\theta(i)} Q_{ij}$.
3) If $JU_i'(1)v_i - \nu_i v_i + Q < 0$, then solve

$$JU_i'(x)v_i - \nu_i v_i x + Q = 0$$

for x[§] and let $L_0 = 0$. Otherwise, let $x = 1$ and

$$L_0 = \frac{JU_i'(1) - \nu_i v_i + Q}{J}.$$

4) Compute

$$p_{i,J} = \frac{U_i'(x)v_i + Q_{i,J} - L_0}{\nu_i v_i}.$$

a) If $p_{i,J} \geq 0$, then the correct value of J is found. Compute p_{ij} as

$$p_{ij} = \begin{cases} \frac{U_i'(x)v_i + Q_{ij} - L_0}{\nu_i v_i} & \text{if } j \leq J \\ 0 & \text{if } j > J \end{cases} ,$$

and the algorithm terminates.

b) Otherwise, let $J \Leftarrow J-1$ and let $Q \Leftarrow Q - Q_{i,J+1}$. If $J \geq 1$, go to step 3. If $J = 0$, set $p_{ij} = 0$ for all j and terminate.

We now summarize the complexity of the above algorithm. All steps except Step 1 and Step 4(a) are $O(1)$, and they may need to be executed $\theta(i)$ times in the worst case. The Step 4(a) is $O(\theta(1))$ but it only needs to be executed once. Sorting $Q_{i,j}$ in Step 1 can be executed in $O(\theta(i) \log \theta(i))$ time using an efficient sorting algorithm such as quicksort. Hence, the overall complexity is at most $O(\theta(i) \log \theta(i))$.

[§] With the choice of the utility function in Section V, the solution can be written explicitly.

B. Sketch of Proof of Proposition 2

Here we provide a brief sketch of the proof of Proposition 2. Interested readers can find the full proof in our technical report [27]. The convergence for $K = \infty$ follows the standard results on Proximal Optimization Algorithms [29]. Hence, we focus on $K = 1$. Let $\vec{x}(t) = \operatorname{argmax}_{\vec{p} \in \Omega} L(\vec{p}, \vec{q}(t), \vec{y}(t))$. The dual update is given by

$$q^l(t+1) = \left[q^l(t) + \alpha^l \left(\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} x_{ij}(t) r_i H_{ij}^l - R^l \right) \right]^+ .$$

Let $(\vec{y^*}, \vec{q^*})$ be any stationary point of algorithm $\mathcal{A}$. Let

$$V(t) = \sum_{l=1}^{L} \frac{1}{\alpha^l} (q^l(t) - q^{l,*})^2 + \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} v_i \nu_i (y_{ij}(t) - y_{ij}^*)^2.$$

We will show that $V(t)$ is non-increasing. Using the property of the projection mapping ([29, Proposition 3.2(b), p211], we have,

$$(q^l(t+1) - q^{l,*}) \Big\{ (q^l(t+1) - \Big[q^l(t) + \alpha^l \left(\sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} x_{ij}(t) r_i H_{ij}^l - R^l \right) \Big] \Big] \Big\} \leq 0.$$

Hence,

$$\begin{aligned} & (q^l(t+1) - q^{l,*})^2 \\ = \; & (q^l(t) - q^{l,*})^2 - (q^l(t+1) - q^l(t))^2 \\ & +2(q^l(t+1) - q^{l,*})(q^l(t+1) - q^l(t)) \\ \leq \; & (q^l(t) - q^{l,*})^2 - (q^l(t+1) - q^l(t))^2 \\ & +2(q^l(t+1) - q^{l,*})\alpha^l \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i H_{ij}^l (x_{ij}(t) - y_{ij}^*). \end{aligned}$$

Further, noting that $\vec{y}(t+1) = \operatorname{argmax}_{\vec{p} \in \Omega} L(\vec{p}, \vec{q}(t+1), \vec{y}(t))$ when $K = 1$, hence,

$$\begin{aligned} V(t+1) - V(t) \leq & -\sum_{l=1}^{L} \frac{1}{\alpha^l} (q^l(t+1) - q^l(t))^2 \\ & +2 \sum_{l=1}^{L} (q^l(t+1) - q^{l,*}) \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} r_i H_{ij}^l (x_{ij}(t) - y_{ij}^*) \\ & + \sum_{i=1}^{I} \sum_{j=1}^{\theta(i)} \frac{\lambda_i}{\mu_i} v_i \nu_i [(y_{ij}(t+1) - y_{ij}^*)^2 - (y_{ij}(t) - y_{ij}^*)^2]. \end{aligned}$$

When the stepsizes α^l are small, $q^l(t+1)$ and $q^l(t)$ are close for all l. Hence, the difference between $x_{ij}(t)$ and $y_{ij}(t+1)$ is also small. Using this fact and the concavity of $U_i(\cdot)$, we can show that the sum of the second term and the third term on the right hand side is no greater than $\sum_{l=1}^{L} b^l (q^l(t+1) - q^l(t))^2$, where the parameters b_l depend on the stepsizes α^l. Hence, if $b^l \leq 1/\alpha^l$, $V(t)$ is non-increasing and thus must have a limit. From here it is not hard to show that $\vec{q}(t)$ and $\vec{y}(t)$ will then converge to a stationary point of algorithm $\mathcal{A}$. The convergence proof for $K > 1$ follows a similar argument.

REFERENCES

[1] X. Lin and N. B. Shroff, "An Optimization Based Approach for Quality of Service Routing in High-Bandwidth Networks," in *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.

[2] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, vol. 12, no. 6, pp. 64–79, November/December 1998.

[3] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of Service Based Routing: A Performance Perspective," in *Proceedings of ACM SIGCOMM*, Vancouver, Canada, September 1998, pp. 17–28.

[4] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees," in *IEEE ICNP*, 1997.

[5] A. Shaikh, J. Rexford, and K. Shin, "Efficient Precomputation of Quality-of-Service Routes," in *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video*, Cambridge, United Kingdom, July 1998.

[6] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the Impact of Stale Link State on Quality-of-Service Routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 162–176, April 2001.

[7] P. B. Key, "Optimal Control and Trunk Reservation in Loss Networks," *Probability in the Engineering and Informational Sciences*, vol. 4, pp. 203–242, 1990.

[8] X. Lin and N. B. Shroff, "Simplification of Network Dynamics in Large Systems," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 813–826, August 2005.

[9] R. J. McEliece and K. N. Sivarajan, "Maximizing Marginal Revenue in Generalized Blocking Service Networks," in *Proc. 30th Annual Allerton Conference on Communication, Control, and Computing*, 1992, pp. 455–464.

[10] F. P. Kelly, "Routing in Circuit Switched Networks: Optimization, Shadow Prices and Decentralization," *Advances in Applied Probability*, vol. 20, pp. 112–144, 1988.

[11] F. P. Kelly, "Routing and Capacity Allocation in Networks with Trunk Reservation," *Mathematics of Operations Research*, vol. 15, no. 4, pp. 771–793, 1990.

[12] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan, "ATM Network Design and Optimization: a Multirate Loss Network Framework," *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 531–543, August 1996.

[13] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, "Call Admission Control and Routing in Integrated Service Networks Using Neuro-Dynamic Programming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 2, pp. 197–208, February 2000.

[14] S. Nelakuditi, Z.-L. Zhang, R. P. Tsang, and D. H. C. Du, "Adaptive Proportional Routing: a Localized QoS Routing Approach," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 790–804, December 2002.

[15] S. Nelakuditi, S. Varadarajan, and Z.-L. Zhang, "On Localized Control in QoS Routing," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 1026–1032, June 2002.

[16] F. P. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[17] W. H. Wang, M. Palaniswami, and S. H. Low, "Optimal Flow Control and Routing in Multi-Path Networks," *Performance Evaluation*, vol. 52, no. 2-3, pp. 119–132, April 2003.

[18] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization Based Rate Control for Multipath Sessions," *Technical Report No. 2001-1, Institute for Systems Research, University of Maryland*, 2001.

[19] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for Multi-Path Routing and Congestion Control," *Presented at the ENS-INRIA ARC-TCP Workshop, Paris, France, Nov. 2003 and at the IMA Workshop on Measurements and Modeling of the Internet, January 2004, available at http://tesla.csl.uiuc.edu/∼srikant/pub.html.*

[20] S. H. Low and R. Srikant, "A Mathematical Framework for Designing a Low-Loss Low-Delay Internet," *Network and Spatial Economics*, vol. 4, no. 1, pp. 75–102, March 2004.

[21] S. Kunniyur and R. Srikant, "An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 286–299, April 2004.

[22] K. J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Nonlinear Programming*, Stanford University Press, Stanford, CA, 1958.

[23] I. Ch. Paschalidis and J. N. Tsitsiklis, "Congestion-Dependent Pricing of Network Services," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 171–184, April 2000.

[24] I. Ch. Paschalidis and Y. Liu, "Pricing in Multiservice Loss Networks: Static Pricing, Asymptotic Optimality, and Demand Substitution Effects," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 425–438, June 2002.

[25] D. Y. Eun and N. B. Shroff, "Simplification of Network Analysis in Large-Bandwidth Systems," in *Proceedings of IEEE INFOCOM*, San Francisco, April 2003.

[26] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *RFC 3031*, January 2001.

[27] X. Lin and N. B. Shroff, " An Optimization Based Approach for Quality of Service Routing in High-Bandwidth Networks," *Technical Report, Purdue University, http://min.ecn.purdue.edu/∼linx/papers.html*, 2003.

[28] S. H. Low and D. E. Lapsley, "Optimization Flow Control–I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.

[29] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, New Jersey, 1989.

[30] R. T. Rockafellar, "Monotone Operators and the Proximal Point Algorithm," *SIAM J. Control and Optimization*, vol. 14, pp. 877–898, August 1976.

[31] X. Lin and N. B. Shroff, "The Multi-path Utility Maximization Problem," in *41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2003.

[32] H. J. Kushner and G. Yin, *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, New York, 1997.

[33] H. J. Kushner and J. Yang, "Analysis of Adaptive Step-Size SA Algorithms for Parameter Tracking," *IEEE Transactions on Automatic Control*, vol. 40, no. 8, pp. 1403–1410, August 1995.

Xiaojun Lin (S'02 / M'05) received his B.S. from Zhongshan University, Guangzhou, China, in 1994, and his M.S. and Ph.D. degrees from Purdue University, West Lafayette, Indiana, in 2000 and 2005, respectively. He is currently an Assistant Professor of Electrical and Computer Engineering at Purdue University.

Dr. Lin's research interests are resource allocation, optimization, network pricing, routing, congestion control, network as a large system, cross-layer design in wireless networks, mobile ad hoc and sensor networks. He received the 2005 best paper of the year award from *Journal of Communications and Networks*. His paper was also one of two runner-up papers for the best-paper award at IEEE INFOCOM 2005.

Ness B. Shroff (S'91 / M'93 / SM'01) received his Ph.D. degree from Columbia University, NY in 1994. He has been at Purdue University since 1994, and is currently a Professor of Electrical and Computer Engineering at Purdue.

Dr. Shroff's research interests span the areas of wireless and wireline communication networks. He is especially interested in fundamental problems in the design, performance, control, security, and pricing of these networks.

Dr. Shroff is an editor for the IEEE/ACM Trans. on Networking and the Computer Networks Journal, and past editor of IEEE Communications Letters. He was the Technical Program co-chair for IEEE INFOCOM'03 (San Francisco, CA), the panel co-chair for ACM Mobicom'02 (Atlanta, GA), program co-chair for the symposium on high-speed networks, Globecom 2001 (San Francisco, CA), and conference chair for the 14th Annual IEEE Computer Communications Workshop (Estes Park, CO). He was the co-organizer of the NSF Workshop on "Fundamental Research in Networking" in April 2003. He has received numerous awards including the NSF CAREER award in 1996, the *Computer Network Journal's* best paper award of 2003, and the 2005 best paper of the year award from *Journal of Communications and Networks*.