

TECHNICAL RESEARCH REPORT

Optimization Based Rate Control for Multipath Sessions

by Koushik Kar, Saswati Sarkar, Leandros Tassiulas

CSHCN TR 2001-2
(ISR TR 2001-1)



The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site <http://www.isr.umd.edu/CSHCN/>

Optimization Based Rate Control for Multipath Sessions

Koushik Kar[†] Saswati Sarkar* Leandros Tassiulas[†]

[†] ECE Department

University of Maryland

College Park, MD 20742, USA

{koushik,leandros}@eng.umd.edu

* EE Department

University of Pennsylvania

Philadelphia, PA 19104, USA

swati@ee.upenn.edu

Abstract

In this paper, we consider the rate control problem for multipath sessions with the objective of maximizing the total user (session) utility. This problem provides a framework in which flow control and routing are jointly optimized. We consider two cases of this problem, and develop two different rate control algorithms for these two cases. The first algorithm is an end-to-end rate control algorithm which requires, on the part of the user, explicit knowledge of the paths that the user uses. The second algorithm is a hop-by-hop rate control algorithm which does not require the user to keep track of the paths it uses. Both the algorithms are distributed and do not require the network to know the user utility functions. We analyze the convergence properties of these algorithms, and discuss how they can be implemented in a real network. Both of these algorithms are computationally simple, and have very low communication overhead.

Keywords

Traffic Management and Control, IP Routing and Congestion Control, Network Management.

I. INTRODUCTION

Effective rate control of traffic sources is required in order to control congestion in a communication network. A rate control strategy should ensure that the network is used efficiently, while guaranteeing that the traffic offered to the network by different traffic sources remain within the limits that the network can carry. Besides these, it is also desirable that the rate control algorithm would ensure that the available network resources are shared by the competing streams of traffic in some fair manner.

An optimization based approach to rate control was suggested in [7]. Here each user is associated with an utility function, which connects the bandwidth given to the user with the “value” associated with the bandwidth (note that throughout the paper, the terms “user”, “session” and “end-host” are used synonymously). The utility could be some measure of say, the perceived quality of audio/video, the user satisfaction, or even the amount paid by the user for the bandwidth allotted to it, and could be different for different users. The rate control objective is to achieve traffic rates that maximize the sum of the user utilities, subject to the link capacity constraints. This problem provides a framework for achieving a wide range of fairness objectives, by choosing the user utility functions appropriately. This utility maximization problem has received considerable attention in recent literature, and several algorithms, based on different approaches, have been proposed for this problem (see [14], [8], [12], [11], [9]).

Most of the above-mentioned work is concerned only with the case where a session sends traffic over a single path. In this paper, however, we consider a generalization of this problem, where there can be multiple paths between the source and the destination of a session. The multipath routing problem has received significant attention in recent literature (see [4], [6], [20], [21], [15]). However, most of the work done in this context is concerned with finding and establishing “good”, loop-free multipaths, and with the forwarding of packets on these paths. The problem of congestion-sensitive rate control on these multipaths has not been thoroughly explored. Note that multiple paths can be used for load balancing, thus allowing more efficient use of the network. In the multipath rate control problem, an user not only determines how much traffic to send, but also how to split the traffic amongst the multiple paths. Thus the multipath rate control problem can be viewed as one in which flow control and routing are jointly optimized.

The optimization-based multipath flow control problem has not been adequately addressed in the literature so far. Although this problem has been addressed in [8], [13], these approaches have certain limitations. The algorithm presented in [13] is only a heuristic, and may not, in general, converge to the optimum solution of the problem (see Section IV for a more detailed discussion on this). The algorithms in [8] solve an approximate version of the original problem rather than the actual problem. Moreover, the authors do not claim any convergence result for their multipath rate control algorithms. From a practical perspective, another drawback of these algorithms is their high communication overhead (we discuss these limitations in detail in Section IV, where we also compare our approach with the previous approaches). Moreover, in these algorithms, the user has to keep track of the different paths it uses, and explicitly adjust the traffic rates on these paths. Therefore, these algorithms do not scale as the number of paths increases (note that there could be an exponential number of paths between a source and a destination), and are not applicable in cases where the user does not have any explicit knowledge about the paths it uses.

In this paper, we consider two formulations of the multipath utility maximization problem, and develop two different algorithms based on them. The first formulation is the same as the one in [8], [13]. Based on this formulation, we present an end-to-end flow control algorithm that has guaranteed convergence, and has very low overhead of computation and communication. However, this algorithm, like previous approaches, assumes that the user knows the set of paths it uses, and is able to directly monitor the traffic rates on these paths. Thus it is applicable to *source routing* or similar other schemes where the source knows the set of paths and determines which path a packet will follow (such as those in [4], [6]).

The second formulation is new, and it allows us to develop a hop-by-hop flow control algorithm that achieves the optimal rates for our problem. This algorithm too has a very low overhead of computation and communication. Moreover, this algorithm does not require the user to keep track of the different paths it uses, and therefore scales with increasing number of paths. It is applicable to *destination-based routing* or similar other schemes where the each router in the path of the packet determines its next-hop node (such as those in [15], [20], [21]).

The motivation, derivation and analysis of algorithms presented in this paper are heavily based on results in non-differentiable optimization theory, mainly those by B.T. Poljak and N.Z. Shor [16] [18].

It is also worth noting here that the second algorithm presented in this paper could be used to solve the well-known network flow problem [1] (as well as its concave and multicommodity generalizations) in a distributed and scalable way.

The paper is organized as follows. In the next section, we present an end-to-end rate control algorithm for the case where the user has explicit knowledge of the set of paths it uses. In Section III, we present a hop-by-hop rate control algorithm where the user is not required to keep track of the different paths. In Section IV, we survey the previous approaches, and compare them with our approach. We conclude in Section V.

II. MULTIPATH FLOW CONTROL WITH EXPLICIT KNOWLEDGE OF PATHS

In this section, we consider the case where the set of paths used by an user is known to the user, and it is able to explicitly adjust the traffic rates on these paths. Typically, source-routing schemes satisfy these conditions, and therefore, the algorithm that we propose in this section is applicable to such schemes.

A. Problem Formulation

Next we describe the network model that we consider, and present a convex programming based formulation of the problem, which will form the basis of the algorithm that we develop.

Consider a network consisting of a set L of unidirectional links, where a link $l \in L$ has capacity c_l ($0 < c_l < \infty$). The network is shared by a set J of unicast (possibly multipath) sessions. Each session j is associated with a utility function $U_j : \mathbb{R}_+ \rightarrow \mathbb{R}$, which is assumed to be concave, continuous, bounded and increasing in $[0, \infty)$.

In the following, we assume that the the set of paths used by an user is already determined/established by some multipath route-finding algorithm (like those in [4], [6]). We are interested in the problem of finding the optimal traffic rates on these paths.

Let P_j be the set of paths used by session $j \in J$. We assume that P_j is known to session (user) j . Let $P = \cup_{j \in J} P_j$ be the set of all paths (over all sessions). For any path $p \in P$, let the set of links on the path be denoted by \tilde{L}_p . Let \tilde{P}_l denote the set of paths (of all sessions) on any link $l \in L$. Associate a rate variable y_p with each path $p \in P$. Note that the traffic rate of session j is equal to $\sum_{p \in P_j} y_p$. Our objective is to maximize the ‘‘social welfare’’, i.e., the sum of the utilities over all the sessions, subject to the link capacity constraints. The problem can be posed as:

$$\begin{aligned}
 \mathbf{P}_1 : \quad & \text{maximize} \quad \sum_{j \in J} U_j \left(\sum_{p \in P_j} y_p \right) \\
 \text{subject to} \quad & \sum_{p \in \tilde{P}_l} y_p \leq c_l \quad \forall l \in L \quad (1) \\
 & y_p \geq 0 \quad \forall p \in P \quad (2)
 \end{aligned}$$

Constraints (1) indicate that the total rate of traffic on a link cannot exceed the capacity of the link, and (2) represent the non-negativity constraints on the rate variables. Note that in the problem formulation, we have not assumed any maximum/minimum constraints on the session/path rates, apart from the obvious non-negativity constraints. However,

if some additional maximum/minimum constraints exist on the session/path rates, our algorithm can easily be modified to take those into account.

We will make the following assumption on the utility functions U_j :

Assumption 1: (Bounded slope) There exists an $A < \infty$ such that $U_j'(\tilde{y}) \leq A \quad \forall \tilde{y} \in [0, \infty)$ for all $j \in J$.

Note that the above assumption inherently assumes that the function U_j is differentiable. This is, however, not necessary. If U_j is not differentiable, the above assumption should hold for all *subgradients*¹ of U_j .

B. An Iterative Algorithm

Next we present an iterative algorithm for the problem \mathbf{P}_1 . Later we will describe how this algorithm can be implemented in a real network in a distributed way.

Let $y_p^{(n)}$ denote the value of the rate variable y_p at the n th iterative step. For each link $l \in L$, define $e_l^{(n)}$ as

$$e_l^{(n)} = \begin{cases} 0 & \text{if } \sum_{p \in \tilde{P}_l} y_p^{(n)} \leq c_l \\ 1 & \text{if } \sum_{p \in \tilde{P}_l} y_p^{(n)} > c_l \end{cases} \quad (3)$$

We will refer to the variable e_l as the “link congestion indicator” of link l . Thus a link l is considered “congested” if $e_l = 1$, and “uncongested” if $e_l = 0$.

The iterative rate update procedure, as will be stated shortly, has a very simple interpretation. In the procedure, the traffic rate on a path of a session is increased according to the derivative of the session’s utility function, while it is decreased according to the number of congested links on the path.

Now let us state the rate update procedure formally. Consider a path p of session j , i.e., $p \in P_j$. In the following, $[\cdot]_+$ denotes a projection² on $[0, \infty)$. At the n th iterative step, the rate variable y_p is updated as follows

$$y_p^{(n+1)} = [y_p^{(n)} + \lambda_n (U_j' (\sum_{p' \in P_j} y_{p'}^{(n)}) - \kappa (\sum_{l \in \tilde{L}_p} e_l^{(n)}))]_+ \quad (4)$$

where κ is a positive constant, and $\lambda_n > 0$ is the step-size at the n th iterative step. Note that if U_j is not differentiable, then U_j' in (4) should be replaced by a subgradient of U_j at that point.

Note that $(\sum_{l \in \tilde{L}_p} e_l)$ is the number of congested links in path p . In the next subsection, we investigate the convergence properties of this iterative algorithm under certain conditions on the constant κ and the step-sizes.

C. Convergence Analysis

In the following, let $y = (y_p, p \in P)$ denote the vector of all path rates. Let $y^{(n)}$ denote the value of this vector at the n th iterative step. Also, let Y^* be the set of optimal solutions of \mathbf{P}_1 (note that the optimal solution can be non-unique).

¹A subgradient [18], defined in the context of convex/concave functions, can be viewed as a generalized gradient, and may exist even if the gradient does not (as is the case for non-differentiable functions).

²For any scalar \tilde{y} , $[\tilde{y}]_+ = \max(0, \tilde{y})$.

Define the overall user utility function $U : \mathfrak{R}_+^{|P|} \rightarrow \mathfrak{R}$ as $U(y) = \sum_{j \in J} U_j(\sum_{p \in P_j} y_p)$, and U^* be the corresponding optimal value. Thus $U^* = U(y^*)$ for any $y^* \in Y^*$. Also let $\rho(y, S) = \min_{z \in S} \|y - z\|$ denote the Euclidean distance of a point y from any compact set S . Now we state some convergence results under various conditions of the step-sizes.

Assume that the sequence of step-sizes $\{\lambda_n\}$ in (4) satisfies the following criteria

$$\lim_{n \rightarrow \infty} \lambda_n = 0 \qquad \sum_{n=1}^{\infty} \lambda_n = \infty \qquad (5)$$

As an example, $\lambda_n = (1/n)$ is a sequence that satisfies (5).

The following theorem shows that our algorithm converges to the optimum if the step-sizes satisfy the above condition.

Theorem 1: Consider the iterative procedure stated in (3)-(4), with the step-sizes satisfying (5). Then for all $\kappa > A$,

$$\lim_{n \rightarrow \infty} \rho(y^{(n)}, Y^*) = 0$$

The above theorem is proved in Appendix II. Note that from the continuity of U it follows that $\lim_{n \rightarrow \infty} U(y^{(n)}) = U^*$. The above theorem basically states that the distance of the rate vector from the set of optimal rates decreases to zero. In the special case where this optimum is unique, the rate vector converges to the unique optimum.

The condition $\sum_{n=1}^{\infty} \lambda_n = \infty$ in (5) can be somewhat relaxed. Our algorithm can be also be shown to converge if the step-sizes λ_n satisfy $\lambda_n = \Lambda \lambda^n$ where $0 < \lambda < 1$ and Λ is a ‘‘sufficiently large’’ constant. However, the condition $\lim_{n \rightarrow \infty} \lambda_n = 0$ is required for exact convergence. In practice, however, it may not be possible (due to precision limitations) or efficient (since it could slow down the convergence rate considerably) to decrease the step-size beyond a certain value. Next we investigate the convergence of our algorithm with constant step-sizes.

If the step-sizes are constant, we can prove a slightly weaker convergence result, as we state below. A similar result holds even in the case where the step-sizes are not constant but converge to some positive value.

For any compact set S , let $\Phi_r(S)$ be the set of all points at a distance of r or less from S , i.e., $\Phi_r(S) = \{y : \rho(y, S) \leq r\}$.

Theorem 2: Let $\{y^{(n)}(\lambda)\}$ denote the sequence of rate vectors defined by (3)-(4) with $\lambda_n = \lambda \forall n$. Then there exists a function $r(\lambda)$ such that $\lim_{\lambda \rightarrow 0^+} r(\lambda) = 0$, and for all $\kappa > A$,

$$\lim_{n \rightarrow \infty} \rho(y^{(n)}(\lambda), \Phi_{r(\lambda)}(Y^*)) = 0 \quad \forall \lambda > 0$$

Theorem 2 can be proved along the same lines as Theorem 1. The theorem states that for a constant step-size, the rate vector converges to a neighborhood around the optimum, and the size of this neighborhood becomes arbitrarily small with decreasing step-size. For a given constant step-size, the size of the neighborhood depends on the parameters of the problem \mathbf{P}_1 , including the utility functions (Appendix III shows how $r(\lambda)$ can be calculated in terms of λ). Note that the above theorem also implies that given any neighborhood around the optimum, we can choose the step-size λ to be sufficiently small so that our algorithm (with constant step-sizes) converges to that neighborhood.

D. Distributed Implementation

Now let us see how the iterative algorithm described above can be implemented in an asynchronous network environment in a distributed way.

Assume that the variable e_l is stored and updated at link l (i.e., at the node where link l originates). Also assume that the rate computation for all paths of a session (according to (4)) is carried out at the source of the session.

Now assume that the source of a session periodically sends out some rate packets (RPs), each containing a rate field R , on the paths of that session. Before sending out an RP, the source sets the R field to the current transmission rate on the path over which the packet is routed. The links on the path of the packet read the R field in order to know the current traffic rate on that path. These rate values are used to update the link congestion indicator.

Note that in order to update the path rates, a source needs to know only the total number of congested links on its path and not the exact set of congested links. Now assume that the receiver of a session periodically sends out some congestion packets (CPs), each containing a rate field C , on the paths (in the backward direction) of that session. The receiver sets the C field to 0 before sending out the CP. Subsequently, when the CP goes through a link on its path, the link adds its congestion indicator to the entry in the C field of the CP. Thus when the CP reaches the source node, the field C of the CP contains the number of congested links on that particular path, which is used in the computation of the new path rates at the source.

Note that although we have described the RPs and the CPs as separate packets, in practice, they can simply be piggybacked on the data and acknowledgement (ACK) packets (in an ACK based protocol). Thus the R field can be a part of the data packet, and the C field a part of the ACK packet.

The link and session algorithms are described below (the step-size is assumed to be constant). In the algorithms, the rates and the link congestion indicators are updated periodically. However, in practice, they can also be updated on the arrivals of RPs/CPs.

Link l 's algorithm:

On receiving an RP of path p :

Read the R field of the RP to know the new value of y_p , and forward the RP onto the next link.

Periodically :

Update e_l as

$$e_l \leftarrow \begin{cases} 0 & \text{if } \sum_{p \in \hat{P}_l} y_p \leq c_l \\ 1 & \text{if } \sum_{p \in \hat{P}_l} y_p > c_l \end{cases}$$

On receiving a CP :

Add e_l to the C field of the CP and forward the CP onto the next link.

Session j 's algorithm:

On receiving a CP of path p :

Read the C field of the CP to know $(\sum_{l \in \bar{L}_p} e_l)$, the current number of congested links in p .

Periodically :

1. For each $p \in P_j$, update y_p as

$$y_p \leftarrow [y_p + \lambda (U'_j(\sum_{p' \in P_j} y_{p'}) - \kappa(\sum_{l \in \bar{L}_p} e_l))]_+$$

2. For each $p \in P_j$, send an RP on p , setting the field R to y_p .

E. Discussion

One drawback of the algorithm described above is that the actual rates need to be communicated from the users to the links. This not only results in a communication overhead, but also requires the links to maintain states on a per-path basis. In practice, however, the traffic on a link can be estimated, and this estimated rate can be used to update the link congestion indicator. Note that only the *total* traffic rate on a link needs to be estimated, and not the individual path rates. Therefore, with estimation of traffic rates at the links, we do not require per-path or per-session information to be maintained at the links. This also removes the overhead of communicating the rates from the users to the links.

Now consider the overhead of communicating to the user the number of congested links on the user's path. Note that the value in the C field of the CP can be at most \bar{L} , the maximum number of links on the path of any session. Thus the congestion field C needs to be $\log_2(\lfloor \bar{L} \rfloor + 1)$ bits long. Therefore for most real networks, including the internet, allocating just one byte to the C field should be sufficient (note that one byte would allow 255 links on a sessions path). Thus the overhead of the network congestion feedback to the users is quite small.

Note that in the algorithm described above, the storage and processing complexity at the end-host is linear in the number of paths it uses. Therefore, this algorithm does not scale as the number of paths between a source-destination pair increases.

III. MULTIPATH FLOW CONTROL WITHOUT EXPLICIT KNOWLEDGE OF PATHS

In this section, we will present a rate control algorithm which does not require the user to keep track of the different paths it uses. This algorithm applies to cases where the user does not know the exact set of paths, or is unable to directly control the traffic rates on these paths. In this algorithm, the storage/processing complexity at the network/end nodes does not depend significantly on the number of paths used. In this case, however, the nodes in the network need to keep track of the sessions whose paths pass through that node. This algorithm, therefore, requires per-session information to be maintained at the network nodes. It is also important to note that while the algorithm previously presented is an end-to-end flow control algorithm, the algorithm that we present in this section works on a hop-by-hop basis.

A. Problem Formulation

Next we provide an alternative formulation of the multipath flow control problem, which forms the basis of the algorithm that we develop. In the following, we assume that each node on a session path maintains a set of *input links*

(the links through which traffic of that session arrives at that node) and a set of *output links* (the links through which traffic of that session departs from that node) for the session. Note that the set of input links of a session at a node is a subset of the set of all incoming links at that node; similarly, the set of output links of a session at a node is a subset of the set of all outgoing links at that node. We assume that set of paths used by a session, and therefore, the sets of input and output links of the session at a node, are already determined by some multipath route-finding algorithm (like those in [20], [21]). We are interested in determining the traffic rates on these links/paths so that the total user utility is maximized.

Consider a network consisting of a set L of unidirectional links, where a link $l \in L$ has capacity c_l ($0 < c_l < \infty$). Let the set of nodes of the network be denoted by K . The network is shared by a set J of unicast (possibly multipath) sessions. Let s_j and d_j respectively denote the source and destination nodes of a session $j \in J$. Let $K_j \subseteq K$ denote the set of nodes which session j traverses (including s_j and d_j). We will refer to the nodes in the set $K_j \setminus \{s_j, d_j\}$ as the “intermediate nodes” of session j . Let $J_l \subseteq J$ denote the set of sessions that use link $l \in L$. Also let $L_j \subseteq L$ denote the set of links used by session $j \in J$. Let I_k and O_k respectively denote the set of incoming and outgoing links at node k . Let $I_{kj} \subseteq I_k$ and $O_{kj} \subseteq O_k$ respectively denote the set of input and output links of session j at node k . Note that $I_{s_j j} = O_{d_j j} = \phi$. As before, each session j is associated with a utility function $U_j : \mathfrak{R}_+ \rightarrow \mathfrak{R}$, which is assumed to be concave, continuous, bounded and increasing in $[0, \infty)$.

Now for each link $l \in L_j$ for each session $j \in J$, associate a variable x_{lj} denoting the traffic rate of session j on link l . Then the utility maximization problem can be posed as:

$$\mathbf{P}_2 : \quad \text{maximize} \quad \sum_{j \in J} U_j \left(\sum_{l \in O_{s_j j}} x_{lj} \right)$$

$$\text{subject to} \quad \sum_{l \in I_{kj}} x_{lj} = \sum_{l \in O_{kj}} x_{lj} \quad \forall k \in K_j \setminus \{s_j, d_j\} \quad \forall j \in J \quad (6)$$

$$\sum_{j \in J_l} x_{lj} \leq c_l \quad \forall l \in L \quad (7)$$

$$x_{lj} \geq 0 \quad \forall l \in L_j \quad \forall j \in J \quad (8)$$

Each constraint in (6) states the flow constraint of a session at a node, i.e., the total input flow of a session at an intermediate node is equal to the total output flow of that session at that node. Constraints (7) are the link capacity constraints, while (8) are the non-negativity constraints on the rates.

B. An Iterative Algorithm

Next we present an iterative algorithm for the problem \mathbf{P}_2 . The algorithm is developed using techniques similar to those used in deriving the algorithm of Section II. We will describe a distributed implementation of this algorithm in Section III-D.

Let $x_{lj}^{(n)}$ denote the value of the rate variable x_{lj} at the n th iterative step. For each link $l \in L$, define $\varepsilon_l^{(n)}$ as

$$\varepsilon_l^{(n)} = \begin{cases} 0 & \text{if } \sum_{j \in J_l} x_{lj}^{(n)} \leq c_l \\ 1 & \text{if } \sum_{j \in J_l} x_{lj}^{(n)} > c_l \end{cases} \quad (9)$$

The variable ε_l is the ‘‘link congestion indicator’’ of link l (ε_l is similar to the variable e_l defined in the Section II-B).

Now for each node $k \in K_j \setminus \{s_j, d_j\}$ for each session $j \in J$, define $\nu_{kj}^{(n)}$ as

$$\nu_{kj}^{(n)} = \begin{cases} 0 & \text{if } \sum_{l \in I_{kj}} x_{lj}^{(n)} = \sum_{l \in O_{kj}} x_{lj}^{(n)} \\ 1 & \text{if } \sum_{l \in I_{kj}} x_{lj}^{(n)} > \sum_{l \in O_{kj}} x_{lj}^{(n)} \\ -1 & \text{if } \sum_{l \in I_{kj}} x_{lj}^{(n)} < \sum_{l \in O_{kj}} x_{lj}^{(n)} \end{cases} \quad (10)$$

We will refer to the variable ν_{kj} as the ‘‘node congestion indicator’’ of node k for session j . For a session j , an intermediate node k is considered ‘‘balanced’’ if $\nu_{kj} = 0$, ‘‘congested’’ if $\nu_{kj} = 1$, and ‘‘underutilized’’ if $\nu_{kj} = -1$.

In the following, we will refer to the node where a link originates as the *start node* of the link. Similarly, we will refer to the node where a link ends as the *end node* of the link. The rate update algorithm, stated below, has a simple intuitive interpretation. In the algorithm, the rate of a session on a link decreases if the start node of the link is underutilized or if the end node of the link is congested. Similarly, the rate of a session on a link increases if the opposite conditions hold, i.e., if the start node of the link is congested or if the end node of the link is underutilized. Also, the rate of a session on a link decreases if the link itself is congested. In addition, if the start node of the link is the source node of the session, then the rate of the session on the link increases according to the derivative of the session utility.

Now we state the rate update procedure formally. Let π_l and θ_l respectively denote the start node and end node of link $l \in L$. Consider a session $j \in J$, and a link $l \in L_j$. In the following, $[\cdot]_+$ denotes a projection on $[0, \infty)$, as before. The update procedure of x_{lj} at step n is

$$x_{lj}^{(n+1)} = \begin{cases} [x_{lj}^{(n)} + \lambda_n (U_j'(\sum_{l \in O_{s_j j}} x_{lj}^{(n)}) - \kappa(\varepsilon_l^{(n)} + \nu_{\theta_l j}^{(n)}))]_+ & \text{if } l \in O_{s_j j} \\ [x_{lj}^{(n)} + \lambda_n (-\kappa(\varepsilon_l^{(n)} - \nu_{\pi_l j}^{(n)}))]_+ & \text{if } l \in I_{d_j j} \\ [x_{lj}^{(n)} + \lambda_n (-\kappa(\varepsilon_l^{(n)} + \nu_{\theta_l j}^{(n)} - \nu_{\pi_l j}^{(n)}))]_+ & \text{otherwise} \end{cases} \quad (11)$$

where κ is a positive constant, and $\lambda_n > 0$ is the step-size at the n th iterative step.

C. Convergence Analysis

In this section, we investigate the convergence of the iterative algorithm outlined in the last subsection.

In the following, let $x = (x_{lj}, l \in L_j, j \in J)$ denote the vector of all rates. Let $x^{(n)}$ denote the value of this vector at the n th iterative step. Also, let X^* be the set of optimal solutions of \mathbf{P}_2 .

Now we state some convergence results under various conditions of the step-sizes, similar to those stated for the algorithm in Section II. As before, we assume that the utility functions satisfy Assumption 1.

Theorem 3: Consider the iterative procedure stated in (9)-(11), with the step-sizes satisfying (5). Then there exists a $\kappa_1 < \infty$, such that for all $\kappa > \kappa_1$,

$$\lim_{n \rightarrow \infty} \rho(x^{(n)}, X^*) = 0$$

Theorem 4: Let $\{x^{(n)}(\lambda)\}$ denote the sequence of rate vectors defined by (9)-(11) with $\lambda_n = \lambda \forall n$. Then there exists a $\kappa_1 < \infty$ and a function $r(\lambda)$ such that $\lim_{\lambda \rightarrow 0^+} r(\lambda) = 0$, and for all $\kappa > \kappa_1$,

$$\lim_{n \rightarrow \infty} \rho(x^{(n)}(\lambda), \Phi_{r(\lambda)}(X^*)) = 0 \quad \forall \lambda > 0$$

In general, κ_1 can be obtained as follows. Note that the set of lagrange multiplier vectors for problem \mathbf{P}_2 is non-empty (from Proposition 5.2.1 of [5]). Let $\mu^* = \{\mu_l^*, l \in L\}$ be any lagrange multiplier vector for \mathbf{P}_2 . Let $\mu_{\max}^* = \max_{l \in L} \mu_l^*$. Then the result of Theorems 3 and 4 can be shown to hold for $\kappa_1 = \mu_{\max}^*$. Under an additional ‘‘interior point assumption’’, upper bounds on μ_{\max}^* can be easily calculated in terms of the parameters of the problem \mathbf{P}_2 (see [5] (pp. 450)). If the set of links L_j , for every session $j \in J$, forms a directed acyclic graph (DAG), then it can be shown that $\kappa > A$ is sufficient to guarantee the convergence results stated in the above theorems. Note that in practice, the routing algorithms usually establish the multipaths in the form of a DAG so as to ensure ‘‘loop-free’’ routing [20], [21].

D. Distributed Implementation

Next we describe how the algorithm described in Section III-B can be implemented in a distributed way in an asynchronous network environment. Let us assume that the link congestion indicator variable is updated at the start node of the corresponding link. Also assume that the node congestion indicator variable is updated at the corresponding node. Thus node π_l is responsible for keeping track of ε_l , while node k is responsible for keeping track of ν_{kj} . Assume that the rate variable x_{lj} is updated at node π_l . In the optimization process, a node has to communicate with the previous-hop and next-hop nodes. In this case too, we assume that this communication is carried out using rate packets (RPs) and congestion packets (CPs). However, unlike the algorithm described in Section II, the RPs and CPs in this case are exchanged only between adjacent network nodes and not between end-hosts. A node on a session path communicates the rate variable updated by it to the session’s next-hop node through the R field of a rate packet (RP) (note that this rate variable is required for the update of the node congestion indicator variable at the next-hop node). The node communicates the node congestion indicator variable updated by it to the session’s previous-hop node through the C field of congestion packet (CP) (note that this node congestion indicator variable is required for the update of the rate variable at the previous-hop node).

The algorithms for the source and intermediate nodes of a session j are stated below. Note that the destination node does not take part in the optimization process. In the algorithms described below, the step-size for rate updates is kept constant at λ .

Source node s_j 's algorithm:**On receiving a CP from θ_l :**

Read the C field of the CP to know the new value of $\nu_{\theta_l j}$.

Periodically :

1. For each $l \in O_{s_j j}$, update ε_l as

$$\varepsilon_l \leftarrow \begin{cases} 0 & \text{if } \sum_{j \in J_t} x_{lj} \leq c_l \\ 1 & \text{if } \sum_{j \in J_t} x_{lj} > c_l \end{cases}$$

2. For each $l \in O_{s_j j}$, update x_{lj} as

$$x_{lj} \leftarrow [x_{lj} + \lambda (U_j'(\sum_{l \in O_{s_j j}} x_{lj}) - \kappa(\varepsilon_l + \nu_{\theta_l j}))]_+$$

3. Send RPs to the next-hop nodes of session j , setting the R field to the updated value of the appropriate rate variable x_{lj} .

Intermediate node k 's algorithm ($k \notin I_{d_j j}$) :**On receiving a CP from θ_l :**

Read the C field of the CP to know the new value of $\nu_{\theta_l j}$.

On receiving an RP from π_l :

Read the R field of the RP to know the new value of x_{lj} .

Periodically :

1. For each $l \in O_{k_j}$, update ε_l as

$$\varepsilon_l \leftarrow \begin{cases} 0 & \text{if } \sum_{j \in J_t} x_{lj} \leq c_l \\ 1 & \text{if } \sum_{j \in J_t} x_{lj} > c_l \end{cases}$$

2. For each $l \in O_{k_j}$, update x_{lj} as

$$x_{lj} \leftarrow [x_{lj} + \lambda (-\kappa(\varepsilon_l + \nu_{\theta_l j} - \nu_{\pi_l j}))]_+$$

3. Update ν_{k_j} as

$$\nu_{k_j} \leftarrow \begin{cases} 0 & \text{if } \sum_{l \in I_{k_j}} x_{lj} = \sum_{l \in O_{k_j}} x_{lj} \\ 1 & \text{if } \sum_{l \in I_{k_j}} x_{lj} > \sum_{l \in O_{k_j}} x_{lj} \\ -1 & \text{if } \sum_{l \in I_{k_j}} x_{lj} < \sum_{l \in O_{k_j}} x_{lj} \end{cases}$$

4. Send RPs to the next-hop nodes of session j , setting the R field to the updated value of the appropriate rate variable x_{lj} .

5. Send CPs to previous-hop nodes of session j , setting the C field to the updated value of the node congestion indicator ν_{k_j} .

Intermediate node k 's algorithm ($k \in I_{d_j j}$) :**On receiving an RP from π_l :**

Read the R field of RP to know the new value of x_{lj} .

Periodically :

1. For each $l \in O_{k,j}$, update ε_l as

$$\varepsilon_l \leftarrow \begin{cases} 0 & \text{if } \sum_{j \in J_t} x_{lj} \leq c_l \\ 1 & \text{if } \sum_{j \in J_t} x_{lj} > c_l \end{cases}$$

2. For each $l \in O_{k,j}$, update x_{lj} as

$$x_{lj} \leftarrow [x_{lj} + \lambda (-\kappa(\varepsilon_l - \nu_{\pi_{lj}}))] _+$$

3. Update $\nu_{k,j}$ as

$$\nu_{k,j} \leftarrow \begin{cases} 0 & \text{if } \sum_{l \in I_{k,j}} x_{lj} = \sum_{l \in O_{k,j}} x_{lj} \\ 1 & \text{if } \sum_{l \in I_{k,j}} x_{lj} > \sum_{l \in O_{k,j}} x_{lj} \\ -1 & \text{if } \sum_{l \in I_{k,j}} x_{lj} < \sum_{l \in O_{k,j}} x_{lj} \end{cases}$$

4. Send CPs to previous-hop nodes of session j , setting the C field to the updated value of the node congestion indicator $\nu_{k,j}$.

E. Discussion

Note that since the node congestion indicator variables $\nu_{k,j}$ takes only three values (namely, 0, 1 and -1), the C field of the CP needs to be allocated only 2 bits. Moreover, note that with measurement-based traffic rate estimation at the nodes, the overhead of rate communication (from a node to its next-hop node) can be avoided. Thus the total communication overhead of this algorithm is quite small.

As mentioned before, since the storage and processing complexity of this algorithm does not depend significantly on the number of paths, this algorithm scales with increasing number of session paths. However, note that a node has to maintain a congestion indicator for each of the sessions going through it. Moreover, even with measurement-based rate estimation, a node has to estimate the traffic rate for each session going through it (and not just the total rate of traffic at that node). This implies that the storage and processing complexity at an intermediate node is proportional to the number of sessions going through it. While maintaining per-session states is usually not feasible in backbone routers (where there can be thousands of sessions going through the router), it can be feasible in Virtual Private Networks (VPNs) and intranets. In backbones, state aggregation could be used to reduce the overhead of these additional flow states, thus making the algorithm more feasible. Comparing the algorithm of Section II with the algorithm just presented, we see that some of the storage/processing complexity at the end-host in the former algorithm has been transferred to the intermediate network nodes in the case of the latter.

An interesting feature of the algorithm presented in this section is that it can be used in solving the multicommodity flow problem [1] with concave utility functions (note that the multicommodity flow problem is usually defined with linear objective functions) in a distributed way. To see this, in problem \mathbf{P}_2 , set $K_j = K \quad \forall j \in J$, so that traffic of a session could pass through all possible nodes in the network. Also set $I_{k,j} = I_k \setminus \{d_j\}$, $O_{k,j} = O_k \setminus \{s_j\} \quad \forall k \in K_j \quad \forall j \in J$, so that the traffic of an session could pass through all possible links (except the links going into the source node or coming out of the destination node, which can obviously be excluded). With these settings, all possible paths between the source and destination nodes of a session are included in the problem formulation. Now \mathbf{P}_2 represents

a generalized multicommodity flow problem which can be solved in a distributed way using the algorithm described above.

IV. RELATED WORK

In this section, we mention some of the recent work on the optimization based flow control problem. We then compare our approach to the multipath utility maximization problem with two other approaches for the same problem, presented in [8], [13].

In recent literature, several different algorithms have been proposed for the single-path case of the problem addressed in this paper. In [14], Low et al. propose an algorithm based on the dual approach for the same problem. In [2], the authors suggest a randomized marking based implementation of the algorithm in [14], that uses only one bit for the network congestion feedback. In [8], the authors propose both primal and dual algorithms that solve approximate versions of the same problem. Another related, but different, approach is proposed in [11], in which the user adjusts its rate based on the proportion of marked packets or end-to-end (measurable) losses. In [12], the authors present a window-based based flow control approach for this problem. Here the users choose some weights and the window-based flow control scheme, on convergence, allocates rates that are proportionally fair with respect to those weights. The weights are updated in such a way that the algorithm finally converges to the optimal rates. In [9], the authors present a simple algorithm for the same problem where the user adjusts its rate based on the number of congested links. Like the approach taken in this paper, the algorithm in [9] is also based on non-differentiable optimization methods, and has certain similarities with the single-path version of the first algorithm presented in this paper. It is important to note that all of the above-mentioned algorithms are end-to-end flow control algorithms.

As already mentioned, the multipath case of the utility maximization problem have not been adequately addressed in the literature. Most of the approaches for the single-path case of the problem, as mentioned above, require strict concavity of the objective function in order to guarantee convergence. However, in the multipath case, the overall objective function may not be strictly concave, even if the individual user utility functions are strictly concave (consider the objective function of \mathbf{P}_1). This is one of the reasons why extending these approaches to the multipath case becomes difficult, and in fact, direct extensions of these algorithms do not provide convergence guarantees.

In [13], the author presents a dual-based algorithm for the problem, which is in fact a generalization of the algorithm presented in [14] for the single-path case of the problem. Here, the network conveys to the user the “congestion prices” for each of the paths it uses (the “congestion price” of a path is the sum of the dual variables of the capacity constraints for all links on the path). The user then sends all its traffic on the path with the smallest congestion price. This algorithm, however, does not guarantee convergence to the optimal rates. Since the dual in this case is non-differentiable, it does not converge to the optimum under a constant step-size gradient projection method. However, it is possible to make the dual converge to the optimum with decreasing step-sizes (see [10] where a case similar to this has been addressed). However, in this case, due to the lack of strict concavity of the primal objective function, it is difficult

to obtain the optimal primal variables (the traffic rates) from the optimal dual variables (the congestion prices). In this case, maximizing the Lagrangian (with the dual variables set to their optimal values) can yield multiple primal solutions, and some of these solutions can be infeasible (see [5] (Chp. 6) for a more detailed discussion on this). In Appendix IV, we provide a counterexample to show that in the algorithm presented in [13], the path rates could constantly fluctuate between two infeasible rates for the problem.

In [8], the authors present both primal and dual algorithms that attempt to solve approximations of the multipath utility maximization problem. These are also generalizations of the algorithms for the single-path case of the problem, presented in the same paper. However, no formal convergence result is stated in the paper for the multipath case of the problem. The authors do show that the value of the approximate objective function (that they are interested in maximizing) is increasing in time, but that does not guarantee convergence to the optimal solution.

Both of the above-mentioned approaches are end-to-end flow control algorithms based on the same formulation that we have used in developing the end-to-end flow control algorithm presented in this paper. However, our algorithm provides guaranteed convergence, and significantly less overhead of communication compared to the previous algorithms (in these algorithms, the “congestion prices” communicated between the network and end-hosts are real numbers, and this poses a difficulty in conveying the prices using a small number of bits without sacrificing precision).

As mentioned before, the previous approaches do not scale with increasing number of paths, and require the user to keep track of the different paths it uses. This necessitated the development of the second algorithm presented in this paper, based on the multicommodity flow formulation. Distributed solutions of network flow and multicommodity flow problems have been proposed in the network optimization literature (see [3]). The problems addressed in this regard typically fall into two categories. The first one is the standard network flow problem. However, this problem does not have the multi-user aspect that we have in our problem. The second kind of problems are flow routing problems based on multicommodity flow formulations. However, these problems are only concerned with the routing of flows, and do not have the flow control aspect that we have in our problem. Moreover, they are concerned with minimizing a cost function of the link load, whereas we are concerned with maximizing the user utility. These factors make our problem considerably different from the previously-addressed problems. Also note that the techniques used to develop the algorithms in this paper are also significantly different from those used in the previous approaches, resulting in a lower overhead of computation and communication.

V. DISCUSSION AND CONCLUSION

In this paper, we addressed the problem of optimization based rate control for multipath sessions. We presented two rate control algorithms, one end-to-end and the other hop-by-hop, that converge to the rates that maximize the total user utility. The algorithms require very simple computations on the part of the users/nodes of the network, and the communication overhead is also small.

The two algorithms presented in this paper are applicable in different scenarios, and each has certain advantages over

the other. The end-to-end rate control algorithm is applicable to schemes where the source of the session can specify the path a packet takes, and therefore can directly monitor the traffic rates on the different paths. This algorithm has the advantage that per-flow state need not be maintained at the network routers. However, in this case, the storage/computation complexity at an end-user is proportional to the number of paths used by the user. The hop-by-hop flow control algorithm is applicable to schemes where the intermediate network nodes determine the next-hop node of a packet. Thus an intermediate node directly controls the traffic rates on the links leading to the next-hop nodes. The complexity of computation/storage for this algorithm does not depend significantly on the total number of paths used. However, in this case, per-session information needs to be maintained at the intermediate nodes. Note that the flow control algorithms used in the current internet are end-to-end flow control algorithms. However, as pointed out in [19], hop-by-hop congestion control algorithms have certain advantages over end-to-end algorithms, and could be used in LAN based networks.

On a different note, it is worth mentioning here that the second algorithm presented in this paper could be used to solve a generalized multicommodity flow problem in a distributed way. Since the network flow problem is a special case of the multicommodity flow problem, this algorithm also provides a distributed solution to the network flow problem.

REFERENCES

- [1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, 1993.
- [2] S. Athuraliya, S. Low, D. Lapsley, "Random Early Marking", *Proceedings of the First International Workshop on Quality of future Internet Services (QofIS) 2000*, Berlin, Germany, September 2000.
- [3] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*, Athena Scientific, 1998.
- [4] J. Chen, P. Druschel, D. Subramanian, "An Efficient Multi-Path Forwarding Method", *Proceedings of Infocom 1998*, March 1998.
- [5] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.
- [6] J. Chen, P. Druschel, D. Subramanian, "A Simple, Practical, Distributed Multi-path Routing Algorithm", TR98-320, July 1998, Rice University.
- [7] F. P. Kelly, "Charging and Rate Control for Elastic Traffic", *European Transactions on Telecommunications*, vol. 8, no. 1, 1997, pp. 33-37.
- [8] F. Kelly, A. Maulloo, D. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability", *Journal of Operations Research Society*, vol. 49, no. 3, 1998, pp. 237-252.
- [9] K. Kar, S. Sarkar, L. Tassiulas, "A Simple Rate Control Algorithm for Maximizing Total User Utility", To appear in *Proceedings of Infocom 2001*, April 2001.
- [10] K. Kar, S. Sarkar, L. Tassiulas, "Optimization Based Rate Control for Multirate Multicast Sessions", To appear in *Proceedings of Infocom 2001*, Anchorage, USA, April 2001.
- [11] S. Kunniyur, R. Srikant, "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", *Proceedings of Infocom 2000*, March 2000.
- [12] R. La, V. Anantharam, "Charge-Sensitive TCP and Rate Control in the Internet", *Proceedings of Infocom 2000*, March 2000.
- [13] S. Low, "Optimization Flow Control with On-line Measurement or Multiple Paths", *Submitted for publication*, www.ee.mu.oz.au/staff/~slow/research/
- [14] S. Low, D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, December 1999.
- [15] J. Moy, "OSPF Version 2", STD 54, RFC 2328, April 1998.

- [16] B. T. Poljak, "A General Method of Solving Extremum Problems", *Soviet Math Doklady*, vol. 8, no. 3, 1967, pp. 593-597.
- [17] R. T. Rockafellar, *Convex Analysis*, Princeton Univ. Press, 1970.
- [18] N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, 1985.
- [19] F. A. Tobagi and W. K. Nouredine, "Back-Pressure Mechanisms in Switched LANs Carrying TCP and Multimedia Traffic", *IEEE Globecom '99, Symposium on High-Speed Networks*, December 1999.
- [20] S. Vutukury and J.J. Garcia-Luna-Aceves, "MPATH: a loop-free multipath routing algorithm", *Elsevier Journal of Microprocessors and Microsystems* 24 (2000), pp. 319-327.
- [21] W.T. Zaumen, J. J. Garcia-Luna-Aceves, "Loop-free Multipath Routing Using Generalized Diffusing Computations", *Proceedings of Infocom 1998*, March 1998.

APPENDIX I: SUBGRADIENTS AND THEIR PROPERTIES

Definition 1: [18] (*Subgradient and Subdifferential*) Consider a convex and continuous function f defined on a convex set $F \subseteq \mathfrak{R}^k$. Then a vector $w_0 \in \mathfrak{R}^k$ is called a *subgradient* of f at a point $x_0 \in F$ if it satisfies

$$f(y) - f(y_0) \geq (w_0, y - y_0) \quad \forall y \in F$$

The *subdifferential* of f at $y_0 \in F$, denoted by $\partial f(y_0)$, is the set of all subgradients of f at y_0 , i.e.,

$$\partial f(y_0) = \{w_0 \in \mathfrak{R}^k : f(y) - f(y_0) \geq (w_0, y - y_0) \quad \forall y \in F\}$$

In general, subgradient at a point may be non-unique. However, if $\nabla f(y_0)$ exists, then $\partial f(x_0) = \{\nabla f(x_0)\}$.

Next we state some properties of subgradients (see Theorems 1.12 & 1.13 of [18]), which will be useful in our analysis.

Lemma 5: Let I be a finite index set. Let $f_i, i \in I$, be convex, continuous functions defined on a convex set F . Let $y_0 \in F$, and $w_{i0} \in \partial f_i(y_0), i \in I$.

(a) Let $f(y) = \sum_{i \in I} a_i f_i(y)$, where $a_i \geq 0, i \in I$. Then $\sum_{i \in I} a_i w_{i0} \in \partial f(y_0)$.

(b) Let $f(y) = \max_{i \in I} f_i(y)$. Define $\tilde{I}(y) = \{i \in I : f_i(y) = f(y)\}$. Then $w_{i0} \in \partial f(y_0)$, for all $i \in \tilde{I}(y_0)$.

In terms of subgradients, the optimality condition is as follows (Theorem 1.11 of [18]):

Lemma 6: Let f be a convex, continuous function defined on a convex set F . Then an interior point y_0 of F is the minimum point of f in F if and only if $0 \in \partial f(y_0)$.

APPENDIX II: PROOF OF THEOREM 1

We will first state a lemma that would be used in the proof of Theorem 1. For each $l \in L$, define $g_l : \mathfrak{R}_+^{|R|} \rightarrow \mathfrak{R}$ as $g_l(y) = \sum_{p \in \tilde{P}_l} y_p - c_l$. Thus the capacity constraint for link l can be simply written as $g_l(y) \leq 0$.

Now consider the following problem

$$\tilde{\mathbf{P}} : \quad \text{maximize} \quad \sum_{j \in J} U_j \left(\sum_{p \in P_j} y_p \right) - \kappa \sum_{l \in L} \max\{0, g_l(y)\}, \quad \text{subject to} \quad y_p \geq 0 \quad \forall p \in P$$

where κ is a non-negative constant.

Now define a function $\tilde{U} : \mathfrak{R}_+^{|R|} \rightarrow \mathfrak{R}$ as $\tilde{U}(y) = \sum_{j \in J} U_j(\sum_{p \in P_j} y_p) - \kappa \sum_{l \in L} \max\{0, g_l(y)\}$. Thus $\tilde{\mathbf{P}}$ is the problem of maximizing $\tilde{U}(y)$ subject to $y \geq 0$. Let \tilde{Y}^* denote the set of optimal solutions of $\tilde{\mathbf{P}}$, and \tilde{U}^* be the corresponding optimal value.

Lemma 7: If $\kappa > A$, then \tilde{Y}^* coincides with Y^* .

Proof: Define $Y_L = \{y : g_l(y) \leq 0 \forall l \in L\}$. Thus the set of link constraints can be simply written as $y \in Y_L$. Consider a point $\tilde{y} \notin Y_L$. Therefore, there exists a $\tilde{l} \in L$, such that $g_{\tilde{l}}(\tilde{y}) > 0$. Choose any $\tilde{p} \in \tilde{P}_{\tilde{l}}$. Then from the properties in Lemma 5, $\frac{\partial \tilde{U}(y)}{\partial y} \Big|_{\tilde{y}_{\tilde{p}}} \leq A - \kappa < 0$. Therefore, $\partial \tilde{U}(\tilde{y})$, the set of subgradients of \tilde{U} at \tilde{y} , can not include the zero vector. Therefore, from Lemma 6, \tilde{y} can not be an optimal solution of $\tilde{\mathbf{P}}$. Therefore all optimal solutions of $\tilde{\mathbf{P}}$ must belong to Y_L . However, for any $y \in Y_L$, the values of the objective function of \mathbf{P} and $\tilde{\mathbf{P}}$ are equal. Therefore any optimal solution of \mathbf{P} is an optimal solution of $\tilde{\mathbf{P}}$, and vice versa. Therefore, for $\kappa > A$, $\tilde{Y}^* = Y^*$. \square

The above result is fairly intuitive. Comparing problems \mathbf{P} and $\tilde{\mathbf{P}}$, we see that the link constraints in \mathbf{P} have been transferred to the objective function in $\tilde{\mathbf{P}}$. The term $\kappa \max\{0, g_l(y)\}$ can be interpreted as the penalty associated with the violation of the capacity constraint of link l . Thus the above lemma states that when the penalty associated with constraint violations is sufficiently large, the optimal solution set of the unconstrained problem $\tilde{\mathbf{P}}$ becomes the same as that of \mathbf{P} .

Proof of Theorem 1: We will first show that $\lim_{n \rightarrow \infty} \rho(y^{(n)}, \tilde{Y}^*) = 0$.

Choose an arbitrary $\delta > 0$. Let $\delta' = (\delta/2)$. For any $\epsilon' > 0$, define $D_{\epsilon'}$ as $D_{\epsilon'} = \{y : y \geq 0, \tilde{U}(y) \geq \tilde{U}^* - \epsilon'\}$. It follows from Theorem 27.2 of [17] that there exists an $\epsilon = \epsilon(\delta') > 0$ such that

$$D_{\epsilon} \subset \{y : \rho(y, \tilde{Y}^*) \leq \delta'\} \quad (12)$$

Consider an n for which $y^{(n)} \notin D_{\epsilon}$. Therefore, $\tilde{U}(y^{(n)}) < \tilde{U}^* - \epsilon$. Choose any $\tilde{y}^* \in \tilde{Y}^*$. Note that the rate update procedure for the receiver nodes, as stated in (4), can be compactly stated as: $y^{(n+1)} = [y^{(n)} + \lambda_n v^{(n)}]_+$, where $v^{(n)}$ is a subgradient of $\tilde{U}(y^{(n)})$, and $[\cdot]_+$ denotes a projection on the non-negative orthant. Since $v^{(n)} \in \partial \tilde{U}(y^{(n)})$, and using the definition of a subgradient (Definition 1), we obtain

$$(v^{(n)}, y^{(n)} - \tilde{y}^*) \leq \tilde{U}(y^{(n)}) - \tilde{U}(\tilde{y}^*) < -\epsilon \quad (13)$$

From Assumption 1), it is easy to see that $v^{(n)}$ is upper-bounded. Let $\|v^{(n)}\| \leq \tilde{A}$ for all n .

Using these facts, and (13), we obtain,

$$\begin{aligned} \|y^{(n+1)} - \tilde{y}^*\|^2 &= \|[y^{(n)} + \lambda_n v^{(n)}]_+ - \tilde{y}^*\|^2 \\ &\leq \|y^{(n)} + \lambda_n v^{(n)} - \tilde{y}^*\|^2 \end{aligned} \quad (14)$$

$$\begin{aligned} &= \|y^{(n)} - \tilde{y}^*\|^2 + \lambda_n^2 \|v^{(n)}\|^2 + 2\lambda_n (y^{(n)} - \tilde{y}^*, v^{(n)}) \\ &< \|y^{(n)} - \tilde{y}^*\|^2 + \tilde{A}^2 \lambda_n^2 - 2\epsilon \lambda_n \end{aligned} \quad (15)$$

Note that (14) follows from the fact that $\tilde{y}^* \geq 0$ (use projection theorem).

Since $\lambda_n \rightarrow 0$, $\lambda_n \leq (\epsilon/\tilde{A}^2)$ when n is sufficiently large. For all such n , from (15), we get

$$\|y^{(n+1)} - \tilde{y}^*\|^2 < \|y^{(n)} - \tilde{y}^*\|^2 - \epsilon\lambda_n \quad (16)$$

Now, for the sake of contradiction, let us assume that there exists a $N'_\epsilon < \infty$ such that $y^{(n)} \notin D_\epsilon$ for all $n \geq N'_\epsilon$. Therefore, there exists $N_\epsilon \geq N'_\epsilon$ be such that (16) holds for all $n \geq N_\epsilon$. Summing up the inequalities obtained from (16) for $n = N_\epsilon$ to $N_\epsilon + m$, we obtain

$$\|y^{(N_\epsilon+m+1)} - \tilde{y}^*\|^2 < \|y^{(N_\epsilon)} - \tilde{y}^*\|^2 - \epsilon \sum_{n=N_\epsilon}^{N_\epsilon+m} \lambda_n \quad (17)$$

which implies that $\|y^{(N_\epsilon+m+1)} - \tilde{y}^*\| \rightarrow -\infty$ as $m \rightarrow \infty$, since $\sum \lambda_n$ diverges. This is impossible, since $\|y^{(N_\epsilon+m+1)} - \tilde{y}^*\| \geq 0$. Hence our assumption was incorrect. Hence, there exists an infinite sequence $n_{1,\epsilon} < n_{2,\epsilon} < n_{3,\epsilon} < \dots$ such that $y^{(n_{i,\epsilon})} \in D_\epsilon$ for all $i = 1, 2, 3, \dots$. This implies that there exists an i_1 such that (16) holds for all $n \geq n_{i_1,\epsilon}$. Also, since $\lambda_n \rightarrow 0$, there exists and i_2 such that $\lambda_n \leq (\delta'/\tilde{A})$ for all $n \geq n_{i_2,\epsilon}$.

Let $i' = \max(i_1, i_2)$. We show that $\rho(y^{(n)}, \tilde{Y}^*) \leq \delta$ for all $n \geq n_{i',\epsilon}$. Pick any $n \geq n_{i',\epsilon}$. There can be three cases:

Case 1: $n = n_{j,\epsilon}$ for some $j \geq i'$: In this case, $y^{(n)} \in D_\epsilon$. From (12), it trivially follows that $\rho(y^{(n)}, \tilde{Y}^*) \leq \delta' < \delta$.

Case 2: $n = n_{j,\epsilon} + 1$ for some $j \geq i'$: In this case, $y^{(n)} = y^{(n_{j,\epsilon}+1)} = [y^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})}]_+$. Thus

$$\begin{aligned} \|y^{(n)} - y^{(n_{j,\epsilon})}\| &= \|[y^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})}]_+ - y^{(n_{j,\epsilon})}\| \\ &\leq \|y^{(n_{j,\epsilon})} + \lambda_{n_{j,\epsilon}} v^{(n_{j,\epsilon})} - y^{(n_{j,\epsilon})}\| \\ &= \lambda_{n_{j,\epsilon}} \|v^{(n_{j,\epsilon})}\| \leq \tilde{A} \lambda_{n_{j,\epsilon}} \leq \delta' \end{aligned} \quad (18)$$

From (18) and the fact that $\rho(y^{(n_{j,\epsilon})}, \tilde{Y}^*) \leq \delta'$ (Case 1), we get

$$\rho(y^{(n)}, \tilde{Y}^*) \leq \rho(y^{(n_{j,\epsilon})}, \tilde{Y}^*) + \|y^{(n)} - y^{(n_{j,\epsilon})}\| \leq \delta' + \delta' = 2\delta' = \delta \quad (19)$$

Case 3: $n_{j,\epsilon} + 1 < n < n_{j+1,\epsilon}$ for some $j \geq i'$: Note that $y^{(n')} \notin D_\epsilon$ for all n' satisfying $n_{j,\epsilon} < n' < n_{j+1,\epsilon}$. From (16), it follows that $\|y^{(n'+1)} - \tilde{y}^*\| < \|y^{(n')} - \tilde{y}^*\|$. Summing up these inequalities obtained for $n' = n_{j,\epsilon} + 1$ to $n - 1$, we obtain $\|y^{(n)} - \tilde{y}^*\| < \|y^{(n_{j,\epsilon}+1)} - \tilde{y}^*\|$. Since this inequality holds for all $\tilde{y}^* \in \tilde{Y}^*$, hence $\rho(y^{(n)}, \tilde{Y}^*) < \rho(y^{(n_{j,\epsilon}+1)}, \tilde{Y}^*)$. Since $\rho(y^{(n_{j,\epsilon}+1)}, \tilde{Y}^*) \leq \delta$ (Case 2), it follows that $\rho(y^{(n)}, \tilde{Y}^*) \leq \delta$.

From cases 1, 2, 3., it follows that $\rho(y^{(n)}, \tilde{Y}^*) \leq \delta$ for all $n \geq n_{i',\epsilon}$. By virtue of the arbitrariness of δ , it follows that $\lim_{n \rightarrow \infty} \rho(y^{(n)}, \tilde{Y}^*) = 0$. Now, from Lemma 7, it follows that if $K > \tilde{K}$, then $\lim_{n \rightarrow \infty} \rho(y^{(n)}, Y^*) = 0$. \square

APPENDIX III: CALCULATION OF $r(\lambda)$

Define a function $\tilde{U}(y)$ as in the proof of Theorem 1, i.e., $\tilde{U}(y) = \sum_{j \in J} U_j(\sum_{p \in P_j} y_p) - \kappa \sum_{l \in L} \max\{0, \sum_{p \in \tilde{P}_l} y_p - c_l\}$. Consider the problem of maximizing \tilde{U}^* subject to $y \geq 0$. Let \tilde{Y}^* be the set of optimal solutions for this problem, and \tilde{U}^* be the corresponding optimal value. Since $\kappa > A$, $\tilde{Y}^* = Y^*$, and $\tilde{U}^* = U^*$ (use Lemma 7 in Appendix II). Let \bar{L} be the maximum number of links on any session's path. Now define the set $\tilde{D}(\lambda)$ as follows

$$\tilde{D}(\lambda) = \{y \geq 0 : \tilde{U}(y) \geq U^* - \frac{1}{2} \kappa^2 \bar{L}^2 |P| \lambda\}$$

Now define $\tilde{r}(\lambda)$ as follows

$$\tilde{r}(\lambda) = \max_{y \in \tilde{D}(\lambda)} \rho(y, Y^*)$$

Then $r(\lambda)$ can be expressed as follows

$$r(\lambda) = \tilde{r}(\lambda) + \kappa \bar{L} \sqrt{|P| \lambda} \quad (20)$$

It is easy to show that $\tilde{r}(\lambda)$, and hence $\tilde{D}(\lambda)$, is bounded. Moreover, from Theorem 27.2 of [17], it follows that $\tilde{r}(\lambda) \rightarrow 0$ as $\lambda \rightarrow 0+$. Therefore, from (20), it follows that $r(\lambda) \rightarrow 0$ as $\lambda \rightarrow 0+$.

APPENDIX IV: A COUNTEREXAMPLE FOR THE DUAL-BASED ALGORITHM

Next we provide a counterexample to show that in the algorithm presented in [13], the path rates may not converge to the optimal rates. Consider the 3-node network shown in Figure 1. There is a single session with source s and destination d , sending traffic on two different paths p_1 and p_2 .

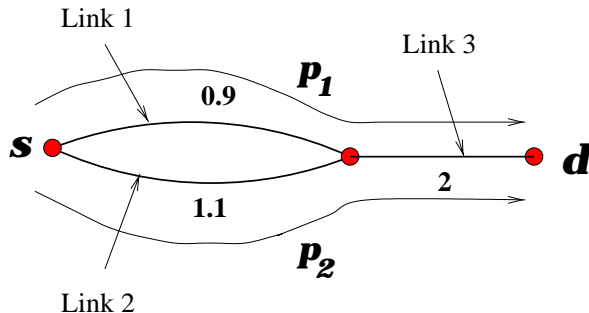


Fig. 1. An example network. (The numbers across the links are the link capacities.)

Assume that the session utility is $\log(1 + x)$, where x is the session rate. Assume that there is a maximum constraint on each path rate, and that is equal to 10 units. It is easy to see that the optimal rates on the two paths are (0.9,1.1). Now assume that the rates and the prices are all zero initially. Moreover, assume that the rates/prices are updated synchronously. Let the step-size for price updates at the n th iteration be equal to $(1/n)$ (such step-sizes ensure the convergence of the dual [10]). Figure 2 shows the path rates in the window 0-250 iterations. Note that after an initial phase, the path rates constantly fluctuate between (0,2) and (2,0). These fluctuations do not die down as the number of iterations increases. Note that the rates fluctuate even as the step-size for price updates become arbitrarily close to zero. Note that the rates (2,0) and (0,2) are infeasible, although these path rates yield the same primal objective function value. Figure 3 shows the link prices in the window 0-250 iterations. In the figure, we see that the link prices converge.

This example demonstrates, that even if the prices converge to their optimal values, the rates may not, and could achieve values that are infeasible to the original problem. This is due to the lack of strict concavity of the primal objective function. Note that we could make the primal objective function strictly concave by adding to it a strictly concave term $\hat{U}_p(y_p)$ for each $p \in P$. For instance, we could take $\hat{U}_p(y_p) = -\frac{1}{2\kappa} y_p^2$, where κ is some large constant (this kind of approach was successfully applied in [10] to solve a similar problem). In this case, however, with the

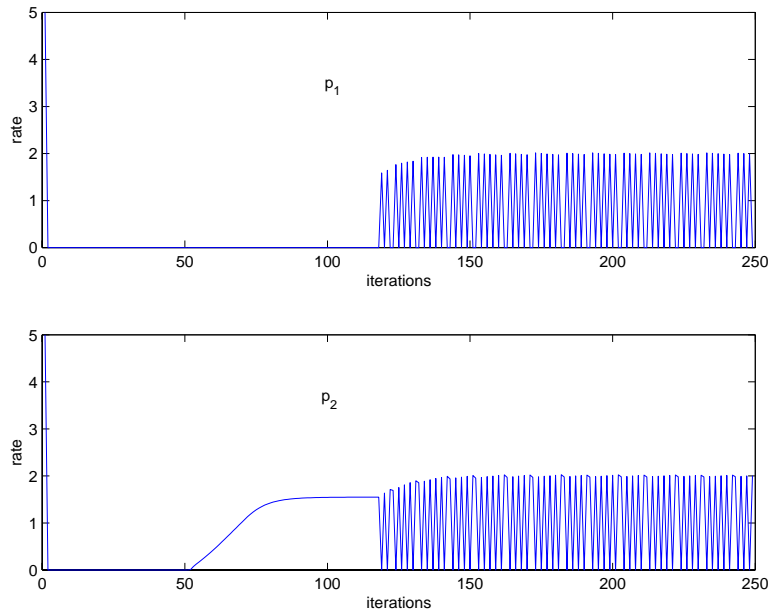


Fig. 2. The rates on paths p_1 and p_2 .

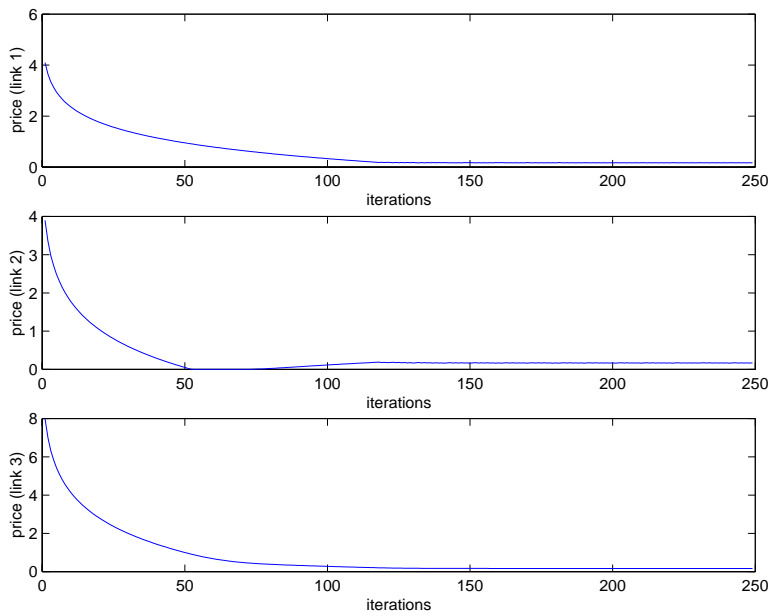


Fig. 3. The prices for links 1,2,3.

addition of these extra functions, the user algorithm would become substantially more complex. More specifically, user j has to maximize the term $U_j(\sum_{p \in P_j} y_p) + \sum_{p \in P_j} \hat{U}_p(y_p) - \sum_{p \in P_j} \nu_p y_p$, where ν_p is the total congestion price for path p . In general, this maximization problem will be too difficult to solve in a single step.