

# Optimization Based Rate Control for Multipath Sessions

K. Kar<sup>a</sup>, S. Sarkar<sup>b</sup> and L. Tassiulas<sup>a</sup>

<sup>a</sup>ECE Department, University of Maryland,  
College Park, MD 20742, USA

<sup>b</sup>EE Department, University of Pennsylvania,  
Philadelphia, PA 19104, USA

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.

## 1. 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 [4]. 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 [10],[5],[9],[8],[7]).

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 [2],[3],[15],[16],[11]). 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 [5], the approaches proposed in [5] have certain limitations. From a practical perspective, an important limitation of the algorithms is their high communication overhead. In these algorithms, the “congestion feedback” communicated from the network to the end-hosts are real numbers, and this poses a difficulty in conveying the feedback using a small number of bits without sacrificing precision.

Moreover, in the algorithms proposed in [5], 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 [5]. 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 (Section 2). 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 [2],[3]).

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 (Section 3). 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 [11],[15],[16]).

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 [12] and N.Z. Shor [13].

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.

## 2. 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.

### 2.1. 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, differentiable, 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 [2],[3]). 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:

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

$$y_p \geq 0 \quad \forall p \in P \tag{2}$$

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 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  $j \in J$ .*

## 2.2. 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 projection on  $[0, \infty)$  (therefore, for any scalar  $\tilde{y}$ ,  $[\tilde{y}]_+ = \max(0, \tilde{y})$ ). 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  $\kappa$  is a positive constant, and  $\lambda_n > 0$  is the step-size at the  $n$ th iterative step.

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  $\kappa$  and the step-sizes.

## 2.3. 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).

Define the overall user utility function  $U : \mathfrak{R}_+^{P_1} \rightarrow \mathfrak{R}$  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

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

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 [6]. 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.

Note that 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. 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  $\lambda_n = \lambda \quad \forall n$ . 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 (refer to [6] to see how  $r(\lambda)$  can be calculated in terms of  $\lambda$ ). Note that the above theorem also implies that given any neighborhood around the optimum, we can choose the step-size  $\lambda$  to be sufficiently small so that our algorithm (with constant step-sizes) converges to that neighborhood.

## 2.4. 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.

**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

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

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  $(\sum_{l \in \tilde{L}_p} e_l)$ , the current number of congested links in  $p$ .

Periodically :

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

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

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

## 2.5. 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 (or even half a byte) to the  $C$  field should be enough (note that

one byte would allow a maximum of 255 links on a session's path, while half a byte would allow a maximum of 15 links). 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.

### 3. Multipath Flow Control without Explicit Knowledge of Paths

In this section, we present a rate control algorithm which does not require the user to keep track of the different paths it uses. The algorithm works on a hop-by-hop basis, and is applicable in scenarios where each node in the network directly controls the traffic rates on the links leading to the next-hop nodes. In this algorithm, the storage/processing complexity at the network nodes/end-host 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. In certain practical scenarios, however, this state overhead at the network nodes could be reduced considerably, as discussed in Section 3.5.

#### 3.1. 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 [15],[16]). 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  $I_l \subseteq J$  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  $U_j : \mathbb{R}_+ \rightarrow \mathbb{R}$ , which is assumed to be concave, differentiable, 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.

### 3.2. 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 2. We will describe a distributed implementation of this algorithm in Section 3.4.

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  $\varepsilon_l$  is the “link congestion indicator” of link  $l$  ( $\varepsilon_l$  is similar to the variable  $e_l$  defined in the Section 2.2). 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  $\nu_{kj}$  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  $\pi_l$  and  $\theta_l$  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  $x_{lj}$  is

$$x_{lj}^{(n+1)} = \begin{cases} [x_{lj}^{(n)} + \lambda_n (U'_j(\sum_{l \in O_{sjj}} x_{lj}^{(n)}) - \kappa(\varepsilon_l^{(n)} + \nu_{\theta_{lj}}^{(n)}))]_+ & \text{if } l \in O_{sjj} \\ [x_{lj}^{(n)} + \lambda_n (-\kappa(\varepsilon_l^{(n)} - \nu_{\pi_{lj}}^{(n)}))]_+ & \text{if } l \in I_{dj} \\ [x_{lj}^{(n)} + \lambda_n (-\kappa(\varepsilon_l^{(n)} + \nu_{\theta_{lj}}^{(n)} - \nu_{\pi_{lj}}^{(n)}))]_+ & \text{otherwise} \end{cases} \quad (11)$$

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

### 3.3. Convergence Analysis

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  $\mathbf{P}_2$ . Next we state some convergence results under various conditions of the step-sizes, similar to those stated for the algorithm in Section 2. 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  $\kappa > \kappa_1$ ,*

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

The value of  $\kappa_1$  depends on the parameters of the problem  $\mathbf{P}_2$ , and can be calculated given a particular instance of the problem (see [6]). In certain important cases,  $\kappa > A$  is sufficient to guarantee the convergence results as stated in the above theorems [6].

### 3.4. Distributed Implementation

Next we describe how the algorithm described in Section 3.2 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  $\pi_l$  is responsible for keeping track of  $\varepsilon_l$ , while node  $k$  is responsible for keeping track of  $\nu_{kj}$ . Assume that the rate variable  $x_{lj}$  is updated at node  $\pi_l$ . 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 2, 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  $\lambda$ .

**Source node  $s_j$ 's algorithm:**

On receiving a CP from  $\theta_l$  :

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

Periodically :

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

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

2. For each  $l \in O_{s_j}$ , update  $x_{lj}$  as  $x_{lj} \leftarrow [x_{lj} + \lambda (U'_j(\sum_{l \in O_{s_j}} x_{lj}) - \kappa(\varepsilon_l + \nu_{\theta_{lj}})) ]_+$ .

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 ( $k \notin I_{d_j}$ ) :**

On receiving a CP from  $\theta_l$  :

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

On receiving an RP from  $\pi_l$  :

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  $\varepsilon_l$  as

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

2. For each  $l \in O_{k_j}$ , update  $x_{lj}$  as  $x_{lj} \leftarrow [x_{lj} + \lambda ( -\kappa(\varepsilon_l + \nu_{\theta_{lj}} - \nu_{\pi_{lj}}) ) ]_+$ .

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 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  $\nu_{k_j}$ .

**Intermediate node  $k$ 's algorithm ( $k \in I_{d_j}$ ) :**On receiving an RP from  $\pi_l$  :Read the  $R$  field of RP to know the new value of  $x_{lj}$ .Periodically :1. For each  $l \in O_{kj}$ , update  $\varepsilon_l$  as

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

2. For each  $l \in O_{kj}$ , update  $x_{lj}$  as  $x_{lj} \leftarrow [x_{lj} + \lambda (-\kappa(\varepsilon_l - \nu_{\pi_{lj}}))]_+$ .3. Update  $\nu_{kj}$  as

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

4. Send CPs to previous-hop nodes of session  $j$ , setting the  $C$  field to the updated value of the node congestion indicator  $\nu_{kj}$ .**3.5. Discussion**

Note that since the node congestion indicator variables  $\nu_{kj}$  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.

Note that since a node has to maintain a congestion indicator for each of the sessions going through it, 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. As pointed out in [14], hop-by-hop congestion control algorithms have certain advantages over end-to-end algorithms, and could be used in LAN based networks. In backbones, state aggregation could be used to reduce the overhead of these additional flow states, thus making the algorithm more feasible (while achieving fairness at a coarser scale). In particular, we could achieve fairness at the level of the *ingress-egress* node pairs by maintaining state per ingress-egress pair at the network nodes ("ingress"/"egress" is the node (router) where a session enters into/departs from the network). Moreover, note that all sessions between the same ingress-egress pair will typically have the same set of input and output links at any network node (since the set of paths that these sessions use will typically be the same). In such a case, it is possible to achieve optimal session rates while maintaining state per ingress-egress pair at the network nodes.

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 **P<sub>2</sub>**, set  $K_j = K \ \forall j \in J$ , so that traffic of a session could pass through all possible nodes in the network. Also set  $I_{kj} = I_k \setminus \{d_j\}$ ,  $O_{kj} =$

$O_k \setminus \{s_j\} \quad \forall k \in K_j \quad \forall j \in J$ , 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.

## REFERENCES

1. R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall, 1993.
2. J. Chen, P. Druschel, D. Subramanian, "An Efficient Multi-Path Forwarding Method", *Proceedings of Infocom 1998*, March 1998.
3. J. Chen, P. Druschel, D. Subramanian, "A Simple, Practical, Distributed Multi-path Routing Algorithm", TR98-320, July 1998, Rice University.
4. F. P. Kelly, "Charging and Rate Control for Elastic Traffic", *European Transactions on Telecommunications*, vol. 8, no. 1, 1997, pp. 33-37.
5. 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.
6. K. Kar, S. Sarkar, L. Tassiulas, "Optimization Based Rate Control for Multipath Sessions", Technical Report TR 2001-1, Institute for Systems Research and University of Maryland, 2001.
7. K. Kar, S. Sarkar, L. Tassiulas, "A Simple Rate Control Algorithm for Maximizing Total User Utility", *Proceedings of Infocom 2001*, April 2001.
8. S. Kunniyur, R. Srikant, "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks", *Proceedings of Infocom 2000*, March 2000.
9. R. La, V. Anantharam, "Charge-Sensitive TCP and Rate Control in the Internet", *Proceedings of Infocom 2000*, March 2000.
10. S. Low, D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, December 1999.
11. J. Moy, "OSPF Version 2", STD 54, RFC 2328, April 1998.
12. B. T. Poljak, "A General Method of Solving Extremum Problems", *Soviet Math Doklady*, vol. 8, no. 3, 1967, pp. 593-597.
13. N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, 1985.
14. 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.
15. 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.
16. W.T. Zaumen, J. J. Garcia-Luna-Aceves, "Loop-free Multipath Routing Using Generalized Diffusing Computations", *Proceedings of Infocom 1998*, March 1998.