Charge-Sensitive TCP and Rate Control in the Internet

Richard J. La, and Venkat Anantharam
Department of Electrical Engineering & Computer Sciences
University of California at Berkeley
hyongla@eecs.berkeley.edu, ananth@eecs.berkeley.edu

Abstract—**We investigate the fundamental problem of achieving the system optimal rates in a distributed environment, which maximize the total user utility, using only the information available at the end hosts. This is done by decomposing the system problem into two subproblems – network and user problems – and introducing an incentive-compatible pricing scheme, while maintaining proportional fairness. We demonstrate that when users update their parameters by solving their own optimization problem, at an equilibrium the system optimum is achieved. Furthermore, this algorithm does not require any explicit feedback from the network and can be deployed over the Internet with modifications only on the end hosts. In the second part of paper we model as a noncooperative game the case where the choice of each user's action has nonnegligible effect on the price per unit flow at the resources and investigate the Nash equilibria of the game. We show, in the simple case of a single bottleneck, that there exists a unique Nash equilibrium of the game. Further, as the number of users increases, the unique Nash equilibrium approaches the system optimum.**

I. Introduction

As the Internet explodes in size and in the number of users, one of the challenging questions network designers face is how to provide a fair and efficient allocation of the available bandwidth. To this end researchers have proposed many different rate allocation mechanisms. In the current Internet, many connections use Transmission Control Protocol (TCP), which is a window-based congestion control mechanism, so as to control the transmission rate. TCP however does not necessarily lead to an efficient or fair allocation of bandwidth among the connections [2], [4]. The fact that the Internet is in a public domain and potentially noncooperative environment has stimulated much work on pricing mechanisms. Researchers have proposed different schemes based on time and volume measurements [8] or on a per-packet charge [13]. Furthermore in [3] it is shown that flat rate charging leads to inefficiency and a large number of low-usage users end up subsidizing a small number of high-usage users. This argues that usage-based pricing is desirable.

Recently Kelly [6] has suggested that the problem of achieving the system optimum can be decomposed into two subproblems – network and user problems – and a system optimum is achieved when users' choices of charges and the network's choice of allocated rates are in equilibrium. Kelly *et al.*[7] have proposed two classes of algorithm which can be used to implement proportionally fair pricing and which solve a *relaxation* of the system optimum problem.

The algorithm in [7] requires explicit feedback from the switches inside network. Due to the size of the Internet, an algorithm that requires an extensive modification inside the network may not be suitable for deployment. Further, many connections in the Internet use TCP to control their transmission rates. In this paper we investigate the fundamental problem of achieving the system optimum that maximizes the aggregate utilities of the users, using only the information available at the end hosts. We introduce an algorithm that may be deployed over the Internet without significant modification within the network. This algorithm can be thought of as a natural extension of an algorithm introduced in [11]. The algorithm effectively decomposes the system problem into user and network problems. While the users solve the user optimization problems on a larger time scale, the underlying window-based transmission rate control mechanism solves the network problem on a smaller time scale. The unique fixed point of the mapping defined from the algorithm is proven to result in the system optimal rates.

The rest of the paper is organized as follows. We first motivate the problem and describe the model used in the analysis and the methodology adopted by the algorithm, followed by some related work. Section IV explains the algorithm, which is followed by a numerical example. Section VI models the problem as a noncooperative game and discusses the Nash equilibrium points of the game.

II. Motivation, Background & Model

In this section we first motivate our problem by describing some of the currently most popular congestion control mechanism used in the Internet. We discuss some of the problems with the current congestion control mechanisms. We then formulate the problem of maximizing the total users' utility and describe the model we will use to present our congestion control proposal.

A. Motivation

From its advent the Internet has been decentralized, relying on disciplined behavior from the users. Without a centralized authority, the network users have a great deal of freedom as to how they share the available bandwidth in the network. The increasing complexity and size of the Internet renders centralized rate allocation control impractical.

In view of these constraints, researchers have proposed many rate allocation algorithms to be implemented at the end hosts in a decentralized manner. These algorithms can be roughly categorized into two classes: rate-based algorithms and window-based algorithms. A rate-based algorithm directly controls the transmission rate of the connection, based on either feedback from the network [7] or on measurements taken at the end host. A window-based algorithm adjusts the congestion window size, which is the maximum number of outstanding packets, in order to control the transmission rate and backlog inside the network associated to the connection.

The most widely deployed flow/congestion control mechanism in current Internet is the Transmission Control Protocol (TCP), which is a window-based congestion control mechanism. TCP however does not necessarily lead to a fair or efficient rate allocation of the available bandwidth. It is well known that TCP[1] as currently implemented suffers from a high packet loss rate and a delay bias. The high packet loss rate is a consequence of periodic oscillation of the window sizes, while the delay bias is the result of a discrepancy in window update rate among different connections. In order to address these issues Brakmo *et al.*[1] have introduced another version of TCP, named TCP Vegas, with a fundamentally different bandwidth estimation scheme. They have demonstrated that TCP Vegas results in a lower packet loss, which in turn leads to higher efficiency, and have suggested that TCP Vegas does not suffer from a delay bias, which leads to a more fair rate allocation[2] of the bandwidth. This was later proved by Mo *et al.*[10] in the case that there is a single bottleneck in the network.

Mo and Walrand [11] in another paper, have proposed and studied another fair window-based end-to-end congestion control mechanism, which is similar to TCP Vegas but has a more sophisticated window updating rule. They have shown that proportional fairness[3] can be achieved by their $(p, 1)$-proportionally fair algorithm and that max-min fairness can be achieved as a limit of (p, α)-proportionally fair algorithms as α goes to ∞ [11].

Clearly fairness is a desirable property for a rate allocation mechanism. There is, however, another aspect of a rate allocation problem that has not been addressed by the previous mechanisms. Due to the various requirements of different applications it is likely that the users will have different utility functions. This suggests that although fairness is a desirable property, fairness alone may not be a good objective. We suggest that a good rate allocation mechanism should not only be fair, but should also allocate the available bandwidth in such a way that the overall utility of the users is maximized. In this paper we describe a pricing mechanism that achieves these goals without requiring knowledge of the users' utility functions and without requiring any explicit feedback from the network.

B. Model and Background

We consider a network with a set $\mathcal{J}$ of resources or links and a set $\mathcal{I}$ of users. Let C_j denote the finite capacity of link $j \in \mathcal{J}$. Each user has a fixed route $\mathcal{J}_i$, which is a non-empty subset of $\mathcal{J}$. We define a zero-one matrix A, where $A_{i,j} = 1$ if link j is in user i's route $\mathcal{J}_i$ and $A_{i,j} = 0$ otherwise. When the throughput of user i is x_i, user i receives utility $U_i(x_i)$. We assume that the utility $U_i(x_i)$ is an increasing, strictly concave and continuously differentiable function of x_i over the range $x_i \geq 0$[4]. Furthermore, we assume that the utilities are additive so that the aggregate utility of rate allocation $x = (x_i, i \in \mathcal{I})$ is $\sum_{i \in \mathcal{I}} U_i(x_i)$. Let $U = (U_i(\cdot), i \in \mathcal{I})$ and $C = (C_j, j \in \mathcal{J})$. In this paper we study the feasibility of achieving the maximum total utility of the users in a distributed environment, using only the information available at the end hosts. Under our model this problem can be formulated as follows.

SYSTEM(U,A,C):

$$\begin{array}{cl} maximize & \sum_{i \in \mathcal{I}} U_i(x_i) \\ subject\ to & A^T x \leq C \\ over & x \geq 0 \end{array} \tag{1}$$

The first constraint in the problem says that the total rate through a resource cannot be larger than the capacity of the resource. Given that the system knows the utility functions U of the users, this optimization problem may be mathematically tractable. However, in practice not only is the system not likely to know U, but also it is impractical for a centralized system to compute and allocate the user rates, due to the large scale of the network. Hence, Kelly in [6] has proposed to consider the following two simpler problems.

Suppose that given the price per unit flow λ_i, user i selects an amount to pay per unit time, p_i, and receives a flow $x_i = \frac{p_i}{\lambda_i}$. Then, the user's optimization problem becomes the following [6].

$USER_i(U_i; \lambda_i)$:

$$\begin{array}{cl} maximize & U_i(\frac{p_i}{\lambda_i}) - p_i \\ over & p_i \geq 0 \end{array} \tag{2}$$

The network, on the other hand, given $p = (p_i, i \in \mathcal{I})$, attempts to maximize the sum of weighted log functions $\sum_{i \in \mathcal{I}} p_i \log(x_i)$. Then the network's optimization problem can be written as follows [6].

NETWORK(A,C;p) :

$$\begin{array}{cl} maximize & \sum_{i \in \mathcal{I}} p_i \log(x_i) \\ subject\ to & A^T x \leq C \\ over & x \geq 0 \end{array} \tag{3}$$

Note that the network does not require the true utility functions $(U_i(\cdot), i \in \mathcal{I})$, and pretends that user i's utility function is $p_i \cdot \log(x_i)$ to carry out the computation. It is shown in [6] that one can always find vectors $\lambda^* = (\lambda_i^*, i \in \mathcal{I})$, $p^* = (p_i^*, i \in \mathcal{I})$, and $x^* = (x_i^*, i \in \mathcal{I})$ such that p_i^* solves $USER_i(U_i; \lambda_i^*)$ for all $i \in \mathcal{I}$, x^* solves $NETWORK(A, C; p^*)$, and $p_i^* = x_i^* \cdot \lambda_i^*$ for all $i \in \mathcal{I}$. Further, the rate allocation x^* is also the unique solution to $SYSTEM(U, A, C)$. This suggests that the problem of solving $SYSTEM(U, A, C)$ can be achieved by an algorithm that solves $NETWORK(A, C; p(t))$ for a given $p(t)$ at time t

[1] The most popular versions of TCP in the Internet are TCP Tahoe and Reno.

[2] Fairness refers to max-min fairness that is discussed later.

[3] Proportional fairness is introduced by Kelly in [6], and is discussed in the next subsection.

[4] Such a user is said to have elastic traffic.

on a smaller time scale, and drives $p(t)$ to p^* on a larger time scale.

Another important aspect of a rate allocation mechanism is fairness. There are many different definitions for fairness, the most commonly accepted one being max-min fairness. A rate allocation is max-min fair if a user's rate cannot be increased without decreasing the rate of another user who is already receiving smaller rate. In this sense, max-min fairness gives an absolute priority to the users with smaller rates. However, oftentimes in order to achieve a max-min fair allocation the efficiency of the network needs to be sacrificed. In order to handle this tradeoff between fairness and efficiency Kelly [6] has proposed another definition of fairness. A vector of rates $x' = (x'_i, i \in \mathcal{I})$ is said to be *weighted proportionally fair*[5] with weight vector p if x' is feasible, i.e., $x' \geq 0$ and $A^T x' \leq C$, and for any other feasible vector x, the following holds:

$$\sum_{i \in \mathcal{I}} p_i \frac{x_i - x'_i}{x'_i} \leq 0$$

Moreover, a rate allocation x solves $NETWORK(A, C; p)$ if and only if it is weighted proportionally fair with weight vector p.

III. Related Work

In this section we briefly describe some of the algorithms that have been proposed in the past and discuss how they are related to the problem we address in this paper.

A. Kelly's Work

Based on the idea of proportional fairness Kelly *et al.*[7] have proposed two classes of *rate-based* algorithms that solve a relaxation of $SYSTEM(U, A, C)$ problem. These algorithms are based on the idea of shadow price charged at a resource, which is a function of the total rate going through the resource. We describe one of the algorithms, which they call the *primal algorithm*, in this subsection.

Suppose that every user adopts a rate-based flow control. Let w_i and $x_i(t)$ denote user i's willingness to pay per unit time and its rate at time t, respectively. Now suppose that at time t each resource $j \in \mathcal{J}$ charges a price per unit flow of $\mu_j(t) = p_j(\sum_{i:j \in \mathcal{J}_i} x_i(t))$, where $p_j(\cdot)$ is an increasing function of the total rate going through it. Consider the system of differential equations

$$\frac{d}{dt} x_i(t) = \kappa(w_i - x_i(t) \sum_{j \in \mathcal{J}_i} \mu_j(t)) \tag{4}$$

where

$$\mu_j(t) = p_j(\sum_{i:j \in \mathcal{J}_i} x_i(t)).$$

These equations can be motivated as follows. Each user first sets a price per unit time it is willing to pay. Then, every user adjusts its rate based on the feedback provided by the resources in the network in such a way that at an equilibrium the price it is willing to pay equals its aggregate cost. The feedback from a resource $J \in \mathcal{J}$ can be thought of as a congestion indicator, requiring a reduction in the flow rates going through the resource.

Kelly *et al.* show that, for fixed w_i, under some conditions on $p_j(\cdot), j \in \mathcal{J}$, the above system of differential equations converge to a point that maximizes the following expression

$$\mathcal{U}(x) = \sum_i w_i \cdot \log x_i - \sum_j \int_0^{\sum_{i:j \in \mathcal{J}_i} x_i} p_j(y) dy.$$

Further, if the users update their willingness to pay $w_i(t)$ according to

$$w_i(t) = x_i(t) \cdot U'_i(x_i(t))$$

while $x_i(t)$ evolves according to (4), then $x(t)$ converges to a unique stable point that maximizes the following revised expression

$$\mathcal{U}(x) = \sum_i U_i(x_i) - \sum_j \int_0^{\sum_{i:j \in \mathcal{J}_i} x_i} p_j(y) dy. \tag{5}$$

Note that the first term in (5) is the objective function in our *SYSTEM(U, A, C)* problem. Thus, the algorithm proposed by Kelly *et al.* solves a relaxation of the $SYSTEM(U, A, C)$ problem.

B. TCP and a Fair End-to-end Congestion Control Algorithm

Unlike a connection with a rate-based congestion control algorithm, a TCP connection adjusts its rate by updating its window size, based on the estimated congestion state of the network. There are several different versions of TCP, which can be categorized into two classes, based on their bandwidth estimation schemes. TCP Tahoe and Reno use packet losses as an indication of congestion in the network, while TCP Vegas [1] and the algorithm proposed by Mo and Walrand [11] use the estimated queue size to adjust the congestion window size[6].

Over the years researchers have observed that the most widely used versions of TCP, which are TCP Tahoe and Reno, exhibit several undesirable characteristics such as a delay bias and a high packet loss rate. In order to deal with these issues Mo and Walrand [11] have investigated the existence of a fair window-based end-to-end congestion control algorithm that updates the congestion window size more intelligently.

We first present the fluid model of the network that describes the relationship between the window sizes, rates, and queue sizes. Throughout the paper we assume that the switches exercise the first-in-first-out (FIFO) service discipline. This model can be represented by the following equations:

$$\begin{aligned} A^T x - C &\leq 0 && (6) \\ Q(A^T x - C) &= 0 && (7) \\ X(AC'q + d) &= w && (8) \\ x \geq 0, q &\geq 0, && (9) \end{aligned}$$

where

$$C = (C_1, ..., C_J)^T, q = (q_1, ..., q_J)^T,$$
$$d = (\bar{d}_1, ..., \bar{d}_I)^T, w = (w_1, ..., w_I)^T,$$
$$X = diag(x), C' = diag(C_1^{-1}, ..., C_J^{-1}), Q = diag(q),$$

[5] This is called *rates per unit charge are proportionally fair* in [6]

[6] We call these loss-based and queue-based TCP, respectively.

w_i is the congestion window size of user i, $\bar{d}_i$ is the fixed propagation delay of route $\mathcal{J}_i$, and q_j denotes the backlog at link j buffer. For simplicity of analysis we assume that the buffer sizes are infinite. The first condition in (6) represents the capacity constraint. The second constraint says that there is backlogged fluid at a resource only if the total rate through it equals the capacity. The third constraint follows from that the window size of connection i should equal the sum of the amount of fluid in transit and the backlogged fluid in the buffers, i.e.,

$$w_i = x_i \cdot \bar{d}_i + q^i,$$

where q^i denotes connection i's total backlog in the buffers.

Let $W = [0, w_{1,max}] \times \cdots \times [0, w_{I,max}]$, where $w_{i,max}$ is connection i's maximum congestion window size announced by the receiver, and $\mathcal{X} = [0, \min_{j\in\mathcal{J}_1} C_j] \times \cdots \times [0, \min_{j\in\mathcal{J}_I} C_j]$. It has been shown in [11] that the rate vector x is a well defined function of the window sizes w, and we can define a function $\mathcal{W}$ that maps a window size vector $w \in W$ to a rate vector $x \in \mathcal{X}$.

Under the (p,1)-proportionally fair algorithm by Mo and Walrand [11] each connection has a target queue size $p_i > 0$ and attempts to keep p_i packets at the switch buffers similarly as in enhanced TCP Vegas. Let $d_i(t), w_i(t)$, and $x_i(t)$[7] denote the round trip delay, the congestion window size, and the rate of connection i at time t, respectively. Suppose that each connection i has a fixed target queue size p_i. Connection i updates its window size $w_i(t)$ according to the following differential equation

$$\frac{d}{dt} w_i(t) = -\kappa \frac{\bar{d}_i}{d_i(t)} \frac{s_i(t)}{w_i(t)},$$

where κ is some positive constant, and $s_i(t) = w_i(t) - x_i(t) \cdot \bar{d}_i - p_i$. Under this algorithm the window sizes converge to a unique point w^* such that for all $i \in \mathcal{I}$

$$w_i^* - x_i(w^*) \cdot \bar{d}_i = p_i,$$

where $x_i(w^*)$ is connection i's throughput when the window sizes are w^*. They show that at the unique stable point w^* of the algorithm the resulting allocation $x(w^*)$ is weighted proportionally fair.

Suppose that users update their windows size according to the following system of differential equations

$$\frac{d}{dt} w_i(t) = -\kappa x_i(t) s_i(t) u_i(t),$$

where

$$s_i(t) = w_i(t) - x_i(t) \cdot \bar{d}_i - \frac{p_i}{(x_i(t)+1)^{\alpha-1}}$$

and

$$u_i(t) = \bar{d}_i - (\alpha - 1) \frac{p_i}{(x_i(t)+1)^{\alpha}}.$$

This algorithm is called a (p, α)-proportionally fair algorithm. They prove that the above algorithm converges to a unique stable point of the system for all fixed α and the max-min fair allocation is achieved as a limit as $\alpha \to \infty$. However, since their work is motivated by the fundamental question of the existence of a fair end-to-end congestion control mechanism, they have not considered the problem of maximizing the aggregate utility of the users, while maintaining fairness.

[7] We use $x_i(t)$ to denote $x_i(w(t))$ when there is no confusion.

IV. Charge-Sensitive TCP

The algorithms suggested in [7], [5] are based on the assumptions that the network can provide the necessary feedback to the users, and users adjust their *rates* based on the feedback information. However, in the current Internet, many, if not most, connections use TCP, which is a window-based congestion control mechanism, to control their rates. Thus, users control the rates only through the window sizes.

There are several arguments for a window-based congestion control algorithm in the Internet over a rate-based algorithm. One of the arguments is the stability of the Internet. Suppose that users use a rate-based congestion control algorithm. If they have incorrect estimates of the available bandwidth and release packets into the network at a rate that is much higher than they should, the network could temporarily experience an extremely high packet loss rate due to buffer overflows and may take a long time to recover from it. A window-based congestion control algorithm not only controls the transmission rate, but also limits the maximum number of outstanding packets according to the congestion window size. This helps alleviate the effect of the inaccurate estimation of the available bandwidth by the users. This is an important advantage of a window-based algorithm. In an open system such as the Internet, it is important to control the number of packets the connections can keep within the network for stability and to ensure that no users can arbitrarily penalize other users by increasing their rates during congestion periods due to incorrect estimates.

In this section we propose an algorithm that can be implemented over the current Internet without extensive modifications inside the network. The algorithm requires simple modifications of the already existing TCP at the end hosts. We demonstrate, using a fluid model, that at an equilibrium of the algorithm, the resulting rates are the optimal rates that solve $SYSTEM(U, A, C)$.

A. Pricing Scheme

Section III-B tells us that, given $(p_i, i \in \mathcal{I})$, the users can reach a solution to *NETWORK(A,C;p)* using a window-based congestion control mechanism, namely the $(p, 1)$-proportionally fair algorithm of Mo and Walrand. Throughout the rest of this paper, we refer to this $(p, 1)$-proportionally fair algorithm when we say TCP. The challenge now is to design a mechanism that drives the users to the right p^* where the resulting rate allocation solves $SYSTEM(U, A, C)$. In this subsection we describe a simple pricing mechanism that can achieve this. We show that the price a user pays can be directly computed by the user without any feedback from the network, by using the same mechanism already built into TCP.

When the total rate through a link is strictly smaller than its capacity, there is no congestion, as each user receives its desired rate without contention from other users. However, when the total rate reaches the link capacity any increase in a congestion window size by a user i in an attempt to increase its own rate results in increased backlog at the resource. This leads to higher queueing delay at the resource. If the users are delay sensitive this increase in queueing delay represents an increase in the implicit[8] cost the users pay through larger delay. From the perspective of the network, this increase in queueing delay may be interpreted as an increase in undesirable delay cost for the system.

Suppose that the network attempts to recover the increase in the system cost due to queueing delay through a pricing mechanism as follows. Let $q_j, j \in \mathcal{J}$, denote the backlog at resource j. When the total rate through the resource is strictly smaller than its capacity, we assume that there is no backlog from (7). When resource $j \in \mathcal{J}$ is congested, i.e., the total rate through it equals its capacity, the resource charges a price, where the price per unit flow g_j is the queueing delay at the resource, i.e., $g_j = \frac{q_j}{C_j}$. Hence, the total price per unit flow for a user with route $\mathcal{J}_i \subset \mathcal{J}$ is $\sum_{j \in \mathcal{J}_i} g_j$, and the total price per unit time user i pays is $x_i \cdot \sum_{j \in \mathcal{J}_i} g_j$. Then, the net benefit or the objective function of the user is given by

$$U_i(x_i) - x_i \cdot \sum_{j \in \mathcal{J}_i} g_j = U_i(x_i) - x_i \cdot \sum_{j \in \mathcal{J}_i} \frac{q_j}{C_j} \tag{10}$$

We have, however, claimed that no information is explicitly fed back to the end hosts from the network. Hence, the switches are not allowed to send any information regarding the current price per unit flow back to the end users.

In order to maximize the objective function in (10) given the current prices per unit flow, a user only needs to know the total price per unit flow of its route, but not the price at each resource. We now show that users using TCP can compute their prices per unit flow without any help from the network. Suppose that each connection knows the propagation delay of its route. In practice this is done by keeping track of the minimum round trip time of the packets [1], [10]. Suppose that the target queue sizes of the users are given by $p = (p_i, i \in \mathcal{I})$, and the users window sizes converge to the stable point of TCP, where the resulting rates are weighted proportionally fair with weight vector p. Then, the price per unit time user i pays, h_i, can be computed as follows:

$$h_i = x_i \cdot \sum_{j \in \mathcal{J}_i} \frac{q_j}{C_j} = \sum_{j \in \mathcal{J}_i} q_j \cdot \frac{x_i}{C_j} = \sum_{j \in \mathcal{J}_i} q_j^i = p_i, \tag{11}$$

where q_j^i is connection i's queue size at resource j. Here we have implicitly assumed that the queue size of each connection at a congested resource is proportional to its rate, which is a consequence of the assumption that the switches exercise FIFO service discipline. Therefore, at the stable point of TCP for a fixed p the price of a user equals its target queue size, and the user can infer its own price from its target queue size.

One important aspect of a pricing scheme is fairness. The price a connection pays for using resource $j \in \mathcal{J}$ should be proportional to its rate. In other words, the price per unit flow for each connection at a resource, should be the same. This is obviously the case with the above pricing scheme. Moreover, it is easy to see that when a new user comes into the network, the price the new user pays is exactly the increase in the total system cost, i.e., the increase in the total queueing delay experienced by the packets. This can be seen from that the price user i pays per unit time equals its target queue size p_i and the total system cost per unit time is given by

$$\sum_{j \in \mathcal{J}} C_j \cdot g_j = \sum_{j \in \mathcal{J}} C_j \cdot \frac{q_j}{C_j} = \sum_{j \in \mathcal{J}} q_j = \sum_{i \in \mathcal{I}} p_i.$$

Therefore, the fairness of the pricing scheme is automatically guaranteed.

B. User Adaptation

In the previous subsection we have analyzed the case where users have fixed their target queue sizes $p = (p_i, i \in \mathcal{I})$. However, as more intelligence is embedded in end systems in the future, the users may be able to vary the parameters $p_i, i \in \mathcal{I}$ to maximize their net utility given by (10). In this subsection we propose a user algorithm and show that this algorithm has a unique equilibrium, at which it yields the optimal rates that solve *SYSTEM(U, A, C)*.

We assume that the utility functions satisfy the following:

(A1) $\hat{p}_i(\lambda_i)$ is decreasing in $\lambda_i > 0$, where $\hat{p}_i(\lambda_i) = \arg\max_{p_i} U_i(\frac{p_i}{\lambda_i}) - p_i$.

Suppose that ϵ is some small positive constant and users update their willingness to pay or price $p_i, i \in \mathcal{I}$, at time t according to

$$p_{i,\epsilon}(t) = \begin{cases} \hat{p}_i(\lambda_i(t^-)) & \text{if } \hat{p}_i(\lambda_i(t^-)) \geq \epsilon \\ \epsilon & \text{otherwise} \end{cases} \tag{12}$$

where $\lambda_i(t^-)$ is the price per unit flow along user i's route at time t^-. We assume that the price updates take place in a much larger time scale, while users allow their window sizes to converge to a point close to the unique stable point of TCP for fixed $(p_i(t), i \in \mathcal{I})$. This is a natural assumption since the window sizes typically take only seconds to converge and users are not likely to keep changing their parameters before estimating the current throughput. The intuition behind the updating rule is as follows. At time t based on the price per unit flow at time t^- user i computes its optimal price that maximizes its net utility. If the current price per unit flow is too high, user i prefers to wait till the price per unit flow is lower. In such a case user i needs to probe the network for the available bandwidth and price per unit flow along its route. In order to measure the residual bandwidth not used by the other users, if there is any, and the price per unit flow, user i needs to set its window size large enough so that it utilizes all residual capacity not taken by the other users. Hence, we assume that user i sets the target queue size, which is its willingness to pay in our model, to some small positive constant ϵ that is arbitrarily small, so that each user can estimate its well defined price per unit flow, which is strictly positive, along the route. We now let $\epsilon \downarrow 0$ and consider the limit case as $\epsilon \downarrow 0$. In the limit case as $\epsilon \downarrow 0$ (12) can be written as

$$p_i(t) \quad = \quad \arg\max_{p_i} U_i(\frac{p_i}{\lambda_i(t^-)}) - p_i$$

[8]We say the cost is implicit because the users do not necessarily have to pay in monetary form.

$$= \begin{cases} 0, & \text{if } \lambda_i(t^-) \geq U_i'(0) \\ p_i \text{ such that } U_i'(\frac{p_i}{\lambda_i(t^-)}) = \lambda_i(t^-), & \\ & \text{if } 0 < \lambda_i(t^-) < U_i'(0) \end{cases} \quad (13)$$

where $U_i'(\cdot) = \frac{\partial U_i(\cdot)}{\partial x_i}$. The updating rule in (13) can be motivated as follows. If the price per unit flow λ_i is larger than or equal to $U_i'(0)$, then user i receives a negative net utility from any positive p_i. Thus, user i should wait till the price per unit flow is smaller than $U_i'(0)$. If the price per unit flow λ_i is strictly smaller than $U_i'(0)$, then there exists a unique solution to $USER_i(U_i; \lambda_i)$ problem in (2). This solution is the unique p_i such that $U_i'(\frac{p_i}{\lambda_i}) = \lambda_i$, which maximizes user i's net utility. Hence, user i should set its price to such p_i.

Equation (13) implicitly assumes that a user does not anticipate the effect of its own action on prices [7] and updates its parameter in a myopic manner. This is a reasonable assumption when the size of the network is large and each user occupies at most a small fraction of a resource in the network or when a user cannot correctly compute the price per unit flow as a function of its own p_i. In some simple cases, however, users may be able to correctly estimate the effect of their own actions on the prices. This is discussed in section V.

Let us define a mapping $\mathcal{T} : \mathcal{P} \to \mathcal{P}$ to be

$$\mathcal{T}_i(p) = \arg\max_{\tilde{p}_i} U_i(\frac{\tilde{p}_i}{\lambda_i(p)}) - \tilde{p}_i \quad (14)$$

where $\mathcal{P} = \Re^{\mathrm{I}}_+$, and $\lambda(p)$ is the price per unit flow vector at the unique stable point of TCP with price vector p. A fixed point p^* of the mapping $\mathcal{T}$ is a vector in $\mathcal{P}$ such that $\mathcal{T}(p^*) = p^*$.[9]

Theorem 1: There exists a unique fixed point p^* of the mapping $\mathcal{T}$, and the resulting rate allocation from p^* is the optimal rate allocation x^* that solves $SYSTEM(U, A, C)$.

Proof: The proof is given in Appendix A. ∎

Theorem 1 tells us that when the users adopt the algorithm given in the paper, at an equilibrium where no user i changes its parameter p_i, the resulting rate allocation coincides with the system optimal rate allocation. This demonstrates that solving $SYSTEM(U, A, C)$ could be accomplished using a simple window-based TCP algorithm in a distributed environment with no additional feedback from the network.

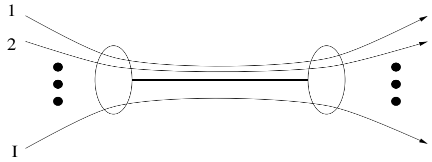

Fig. 1. Single bottleneck case.

A natural question that arises now is whether or not the user prices $p_i, i \in \mathcal{I}$, converge to the unique fixed point of the mapping $\mathcal{T}$. Due to the complex relationship between the price vector and the resulting rate allocation, showing the convergence of users prices in a general network is a challenging problem. In this section we only show, under some conditions, the convergence of user prices in the case of a single bottleneck link, i.e., every user has only one and the same bottleneck link. In a single bottleneck case the price per unit flow is same for all users. Hence, each user updates its price based on the current price per unit flow of the system if it is available. Suppose that the current price is $p \neq 0$ and $p^t = \sum_i p_i$. Let

$$\check{p}_i(p^t) = \arg\max_{p_i \in \mathcal{P}_i} U_i(\frac{p_i}{\lambda}) - p_i,$$

where $\lambda = \frac{p^t}{C}$, C is the capacity of the bottleneck link, and $\mathcal{P}_i = [0, P^i_{max}]$. Here P^i_{max} represents user i's budget constraint, which is assumed to be sufficiently large so that the constraint is not active at the equilibrium. We assume that there exists at least one $i \in \mathcal{I}$ such that $U_i'(0) > \lambda_{max} = \frac{\sum_i P^i_{max}}{C}$ and that the initial price vector is such that $p^t(0) = \sum_i p_i(0) > 0$. This ensures that $p(n) \neq 0$ for all $n \geq 1$.

Suppose that the utility functions satisfy the following additional assumption.

(A2) There exists $M > 0$ such that
(i) for all $p^t < p^{*t}, \check{p}^t(p^t) < 2Mp^{*t} - (2M-1)p^t$
(ii) for all $p^t > p^{*t}, \check{p}^t(p^t) > 2Mp^{*t} - (2M-1)p^t$
where $\check{p}^t(p^t) = \sum_i \check{p}_i(p^t)$ and $p^{*t} = \sum_i p_i^*$.

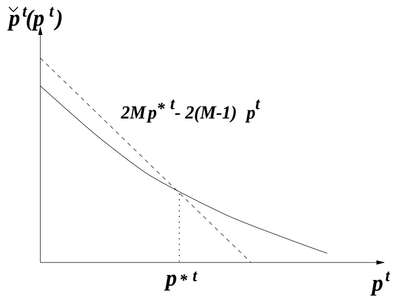

Fig. 2. Assumption (A2).

Consider the following update scheme. We first assume that all users are synchronized and model the user updates with a discrete-time model. At each period $n \geq 1$, every user i updates its price based on its rate $x_i(n-1)$ and price $p_i(n-1)$ as follows:

$$p_i(n) = p_i(n-1) + \frac{\check{p}_i(n) - p_i(n-1)}{M}, \quad (15)$$

where $\check{p}_i(n) = \arg\max_{p_i \in \mathcal{P}_i} U_i(\frac{p_i}{\lambda(n-1)}) - p_i$ and $\lambda(n-1) = \frac{p^t(n-1)}{C}$, where $p^t(n-1) = \sum_i p_i(n-1)$. Once every user finishes updating its price, they wait long enough till the window sizes converge. After window sizes converge, the users repeat the above update procedure, based on the new rate allocation and prices $p_i(n)$. This is called Jacobi update scheme.

The following theorem tells us that the user prices $p(n)$ converge to p^* under assumptions (A1)-(A2).

Theorem 2: In a single bottleneck case, under assumptions (A1) and (A2), the user prices $p(n) = (p_1(n), ..., p_I(n))$ converge to $p^* = (p_1^*, ..., p_I^*)$ as $n \to \infty$ under the Jacobi update

[9]The fixed point of this mapping should be interpreted as the limit of a sequence of fixed points $p(n)$ of the mappings $T_{\epsilon(n)}$, where

$$T_{i,\epsilon(n)}(p) = \begin{cases} \hat{p}_{i,\epsilon(n)}(\lambda_i(p)) & \text{if } \hat{p}_i(\lambda_i(p)) \geq \epsilon(n) \\ \epsilon(n) & \text{otherwise} \end{cases}$$

and $\epsilon(n) \downarrow 0$.

scheme, i.e.,

$$\lim_{n\to\infty} p(n) = p^*.$$

Proof: The proof is given in Appendix B. ■

In practice, however, the network users are almost never synchronized. Hence, it is important to show the convergence of user prices under an asynchronous update scheme with possibly delayed information. In other words, some users may not have an access to the most recent value of the price per unit flow, possibly due to still fluctuating window size and may decide to use an old value instead.

Let T_i be the set of periods at which user i updates its price, and

$$p_i(n+1) = p_i(n) + \frac{\check{p}_i(\lambda(\tau_i(n))) - p_i(n)}{M}, \text{ for all } n \in T^i,$$

where $0 \le \tau_i(n) \le n$. We assume that the sets $T^i, i \in \mathcal{I}$, are infinite and if $\{n_k\}$ is a sequence of elements in T^i that tends to infinity, then

$$\lim_{k\to\infty} \tau_i(n_k) = \infty.$$

The update scheme described here is called a *totally asynchronous update scheme*.

Theorem 3: In a single bottleneck case, under assumptions (A1) and (A2), the user prices $p(n) = (p_1(n), ..., p_I(n))$ converge to $p^* = (p_1^*, ..., p_I^*)$ as $n \to \infty$ under the totally asynchronous update scheme.

The proof of Theorem 3 is omitted in this paper. Refer to [9] for the proof.

Let us give a few examples of user utility functions that satisfy the assumptions (A1)-(A2). Suppose that user i's utility function $U_i(x_i)$ has a form

1. $U_i(x_i) = c \cdot x_i^a$, where $c > 0$ and $0 < a < 1$
2. $U_i(x_i) = c \cdot \log(x_i + b)$, where $c > 0$ and $0 \le b \le 1$

Then, one can easily show that with appropriate budget constraints on the users, both assumptions (A1) and (A2) hold. A numerical example of the convergence of user prices to the unique fixed point of the mapping $\mathcal{T}$ with the second type of utility function is given in the next section.

In this section for the purpose of analysis we have assumed that users are delay insensitive. However, if they are sensitive to delay and the cost due to queueing delay is given by h_i in (11), then the algorithm does not require any pricing mechanism. Recall that the purpose of the pricing mechanism is to impose the system cost due to queueing delay on the users in an incentive-compatible way. In this case when users maximize their utility functions with explicit delay cost, the resulting rate allocation at an equilibrium are the optimal rates.

V. Numerical Example

In this section we give a numerical example of a simple network and demonstrate the convergence of users' parameters through simulation. Although the convergence results have been proved only for single bottleneck cases, the simulation results indicate that the user rates converge to the system optimal rates even in general networks with the utility functions satisfying the assumption (A1) in the previous section, convexity of $\check{p}_i(\lambda_i)$ in λ_i and an appropriate damping constant M.

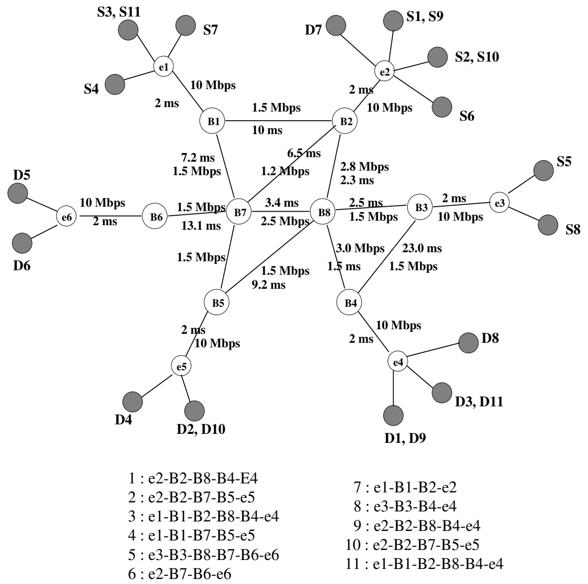

Fig. 3. Topology of the simulated network and the users' routes.

$user$	$U_i(x_i)$	x_i^* (Mbps)	p_i^*
1	$5\log(x+1)$	0.584	4.932
2	$7\log(x+1)$	0.283	6.808
3	$9\log(x+1)$	0.521	8.864
4	$6\log(x+1)$	0.724	5.934
5	$8\log(x+1)$	1.076	7.941
6	$10\log(x+1)$	0.424	9.815
7	$4\log(x+1)$	0.459	3.931
8	$7\log(x+1)$	1.500	6.963
9	$10\log(x+1)$	1.175	9.932
10	$12\log(x+1)$	0.492	11.808
11	$9\log(x+1)$	0.521	8.864

TABLE I

Utility functions of the users and the optimal rates and prices.

The topology of the simulated network is shown in Figure 3. There are 11 users that share the network. The source-destination pairs of the users are given in the figure, and the capacities and delays of the links are indicated by the numbers next to the links. The utility functions of the users are given in Table I. The simulation is run using the Network Simulator (NS) developed at Lawrence Berkeley Laboratory (LBL) and UC Berkeley. Each connection updates its price after it receives the acknowledgement for the k_i-th packet. Hence, the price updates are not synchronized due to different round trip delays. For the simulation we have set $k_i = 25$ and the damping constant $M = 5$ for all connections.

The prices and rates of some of the users are plotted in Figure 4. The dotted lines in the plots represent the optimal rates and unique equilibrium prices, respectively. These plots clearly demonstrate the convergence of the user prices and, thus, the rates to the system optimum.

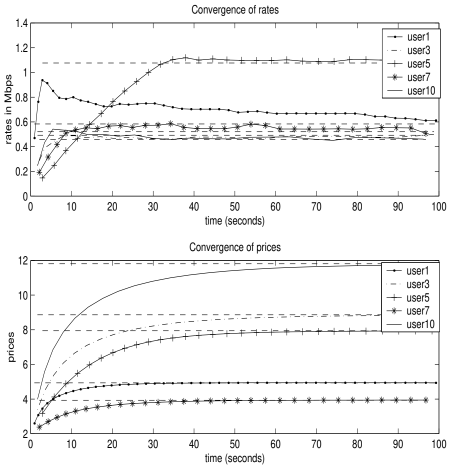

Fig. 4. Convergence of user prices and rates.

VI. Nash Equilibrium

In the previous sections a user is assumed not to anticipate the effect of its own action on the prices. In other words, a user, given an opportunity to update its price, solves (13) to compute its best price as a function of *fixed* $\lambda_i(t)$. However, the price per unit flow of user i is a function of p_i, i.e., $\lambda_i(t) = \lambda_i(p_i, p_{-i}(t))$, where $p_{-i}(t) = (p_1(t), ..., p_{i-1}(t), p_{i+1}(t), ..., p_I(t))$. This assumption may be reasonable in the Internet due to its size and the number of users. In some special cases, however, as a user occupies more and more of a resource in the network, eventually the user will have a nonnegligible impact on prices[10] and an incentive to estimate its impact on the prices [7], i.e., user i may find it beneficial to solve

$$\arg\max_{p_i} U_i\left(\frac{p_i}{\lambda_i(p_i, p_{-i}(t^-))}\right) - p_i \qquad (16)$$

rather than (13). In this section we study these cases where each user has some market power over the prices and is capable of correctly estimating $\lambda_i(p_i, p_{-i}(t^-))$, and investigate how the users may behave. Since it is still questionable that the users can correctly predict the effect of their own parameters on the prices per unit flow in a complicated network, we limit our analysis only to relatively simple networks of a single bottleneck. However, one would expect the same intuition to be extended to more complicated networks. Since both TCP Vegas and the algorithm by Mo and Walrand use the queue size to probe for the available bandwidth, we assume that each user keeps at least P_{min} amount of queue in the network at all times to find available bandwidth. This can be made arbitrarily small as in the previous section. We show that as the number of users increases, the rates at Nash equilibrium points, which will be defined shortly, converge to the system optimal rates.

We consider the simple case of single bottleneck in Figure 1. There are I ≥ 2 users sharing a single link with capacity C. Given the choice of parameters $p = (p_i, i \in \mathcal{I})$, the rate of user i is given by $\frac{p_i \cdot C}{\sum_{k \in \mathcal{I}} p_k}$, where $p_i \in \mathcal{P}_i = [P_{min}, P^i_{max}]$ and P^i_{max} is user i's budget constraint. If an agent on a computer is acting on behalf of the user, the user may specify the maximum amount it is willing to pay per unit time as done in [6]. In this case the price per unit flow depends only on $\sum_j p_j$, and a user i with positive p_i can easily estimate $\sum_j p_j$ and $\sum_{j \neq i} p_j$. Hence, the user can compute its own rate as it changes its parameter, and it is reasonable to assume that users can easily anticipate the effect of its parameter on the price per unit flow.

Since the users are likely to be interested only in maximizing their own net benefit, i.e., users are selfish, it is natural to model the problem as a noncooperative game, where the action space for user $i, i \in \mathcal{I}$, is $\mathcal{P}_i$. Let us first define a Nash equilibrium point (NEP) of this game. A price or target queue size vector p^* is an NEP[11] if, for all $i \in \mathcal{I}$, the following holds :

$$p_i^* \in \arg\max_{p_i \in \mathcal{P}_i} \; U_i(x_i(p^*_{-i}, p_i)) - p_i, \qquad (17)$$

where p^*_{-i} denotes $(p^*_1, ..., p^*_{i-1}, p^*_{i+1}, ..., p^*_{\mathrm{I}})$ and $x_i(p^*_{-i}, p_i) = \frac{p_i}{\sum_{j \neq i} p^*_j + p_i} \cdot C$. Equation (17) simply means that no user i can increase its net utility by using another $p_i \in \mathcal{P}_i$. One can easily show that the model given above gives rise to a concave game, and hence the existence of an NEP is guaranteed by Theorem 1 in [12]. Furthermore, the Kuhn-Tucker conditions yield the necessary and sufficient conditions for NEP in this case. In other words, p^* is an NEP if and only if the following conditions hold:

$$\begin{aligned}
U_i'(x_i(p^*)) \cdot \frac{\sum_{k \neq i} p^*_k \cdot C}{(\sum_k p^*_k)^2} &\leq 1 \qquad \text{if } p^*_i = P_{min} \\
U_i'(x_i(p^*)) \cdot \frac{\sum_{k \neq i} p^*_k \cdot C}{(\sum_k p^*_k)^2} &= 1 \qquad \text{if } P_{min} < p^*_i < P^i_{max} \\
U_i'(x_i(p^*)) \cdot \frac{\sum_{k \neq i} p^*_k \cdot C}{(\sum_k p^*_k)^2} &\geq 1 \qquad \text{if } p^*_i = P^i_{max}
\end{aligned}$$

The following theorem states that in the simple case of single bottleneck link there exists a unique NEP of the game.

Theorem 4: In the case of simple network where there is a single bottleneck link with finite capacity and a finite number of users, there exists a unique Nash equilibrium of the game.

Proof: The proof is given in Appendix C. ■

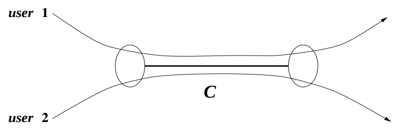

Fig. 5. Inefficiency of Nash equilibrium.

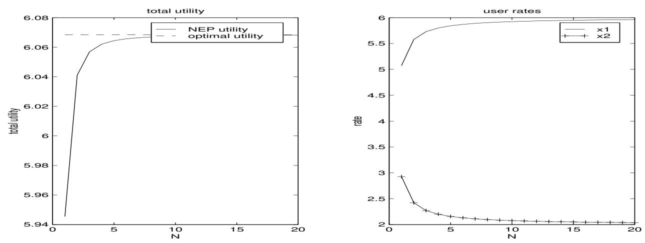

Fig. 6. Total utility and user rates.

[10] In this case users are said to have market power.

[11] In this paper we only consider pure strategy Nash equilibria under the assumption that the action space of each user is limited to pure strategy space.

One of problems with Nash equilibria is that they are not necessarily efficient [2]. Consider the example in Figure 5. There are two users sharing a single bottleneck link. The utility functions of users 1 and 2 are $U_1(x_1) = 3 \cdot \log(x_1)$ and $U_2(x_2) = \log(x_2)$, respectively, and the capacity of the bottleneck link is $C = 8$. Consider the following. Replace each user with $N \geq 1$ identical users and increase the capacity of the link by a factor of N at the same time. In other words, for a given $N \geq 1$, there are N users with utility function $U(x) = 3 \cdot \log(x)$ and N users with utility function $U(x) = \log(x)$, and the capacity of the link is $8N$. Let a user with $U(x) = 3 \cdot \log(x)$ be a type-1 user and a user with $U(x) = \log(x)$ a type-2 user. We have plotted the total utility of the users at the unique NEP, normalized by N as a function of N in the first plot and the rates of type-1 and type-2 users as a function of N in the second plot in Figure 6. The dotted line in the first plot denotes the optimal utility achieved by $x_1 = 6$ and $x_2 = 2$. These plots clearly show the inefficiency of NEP when N is small. As $N \to \infty$, the market power of each user diminishes and the NEP rates asymptotically approach the optimal rates. This is consistent with the results in the previous sections. When the users have no market power, the resulting rates of the users at an equilibrium coincide with the optimal rates. This suggests that in the Internet due to the large number of users, if an NEP exists and the users indeed converge to an NEP, the rate allocation at the NEP is likely to be close to the system optimal rates.

REFERENCES

[1] L.S. Brakmo and L.L. Peterson. Tcp vegas: end to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–80, October 1995.

[2] P. Dubey. Inefficiency of Nash equilibria. *Mathematics of Operations Research*, 11:1–8, 1986.

[3] R. Edell and P. Varaiya. INFOCOM 1999 Keynote Talk. available at http://www.INDEX.berkeley.edu/reports/99-009S.

[4] S. Floyd and V. Jacobson. Connection with multiple Congested Gateways in packet-Switched Networks,Part1: One-way Traffic". *ACM Computer Communication Review*, 21(5):30–47, August 1991.

[5] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. Available at http://www.statslab.cam.ac.uk/˜ frank, June 1998.

[6] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol.8(1):33–7, January 1997.

[7] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, vol.49(3):237–52, March 1998.

[8] F.P. Kelly. On tariffs, policing, and admission control for multiservice networks. *Operations Research Letters*, vol.15(1):1–20, Feb. 1994.

[9] Richard J. La and Venkat Anantharam. Charge-Sensitive TCP and Rate Control in the Internet. Full version in preparation.

[10] J. Mo, R. J. La, V. Ananthram, and J. Walrand. Analysis and Comparison of TCP Reno and Vegas. In *Proc. INFOCOM '99*, March 1999.

[11] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. Technical Report M98/46, UCB/ERL, July 1998. available at http://www.path.berkeley.edu/˜ jhmo.

[12] J.B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 33(3):520–534, July 1965.

[13] H.R. Varian and J.K. Mackie-Mason. Pricing the internet. *Public Access to the Internet*, 1994.

APPENDIX

I. PROOF OF THEOREM 1

At an equilibrium p^* of the algorithm, where user prices stay constant, we have the following:

1. $p_i^* = x_i^* \cdot \lambda_i^*$, where x^* and λ^* are the resulting weighted proportionally fair allocation with weight vector p^* and the price per unit flow vector, which is strictly positive.
2. x^* solves $NETWORK(A, C; p^*)$.
3. p_i^* solves $USER_i(U_i; \lambda_i^*)$ for all $i \in \mathcal{I}$.

(1) follows trivially from that $p_i^* = x_i^* \cdot \lambda_i^*$ by definition. (2) holds at the unique stable point of TCP with fixed p^* [11]. (3) is true by the construction of the algorithm. Furthermore, any p^* with its corresponding λ^* and x^*, that satisfies (1)-(3) is an equilibrium of the algorithm. By Theorem 2 in [6], the existence of such p^* is guaranteed and the resulting rate allocation x^* is the unique solution to $SYSTEM(U, A, C)$.

The uniqueness of equilibrium point can be shown as follows. First, recall that there exists unique x^* that solves $SYSTEM(U, A, C)$, from the strict concavity of the utility functions. From the Kuhn-Tucker conditions, which are the necessary and sufficient conditions, $\hat{p}_i$ solves $USER_i(U_i; \lambda_i^*)$ if and only if it satisfies

$$\begin{aligned} \hat{p}_i > 0 \quad &\Rightarrow \quad U_i'(\frac{\hat{p}_i}{\lambda_i^*}) = \lambda_i^* \\ \hat{p}_i = 0 \quad &\Rightarrow \quad U_i'(0) \leq \lambda_i^*. \end{aligned} \tag{18}$$

Hence, at any equilibrium p^* one can see that

$$\begin{aligned} x_i^* > 0 \quad &\Rightarrow \quad U_i'(\frac{p_i^*}{\lambda_i^*}) = U_i'(x_i^*) = \lambda_i^* \\ x_i^* = 0 \quad &\Rightarrow \quad U_i'(0) \leq \lambda_i^*. \end{aligned} \tag{19}$$

This immediately tells us that there exists a unique fixed point from

$$p_i^* = x_i^* \cdot \lambda_i^* = \begin{cases} x_i^* \cdot U_i'(x_i^*) & \text{if } x_i^* > 0 \\ 0 & \text{if } x_i^* = 0 \end{cases}$$

II. PROOF OF THEOREM 2

Note that, in a single bottleneck case, updating scheme of the users depends only on the current price per unit flow $\lambda(n) = \frac{p^t(n)}{C}$, where C is the bottleneck link capacity. Hence, in order to show the convergence of $p(n)$, it suffices to show the convergence of $p^t(n) = \sum_i p_i(n)$ to $p^{*t} = \sum_i p_i^*$.

Since we have assumed that $U_i(\cdot), i \in \mathcal{I}$, are continuously differentiable, one can see that $\check{p}^t(p^t)$ is a continuous function of p^t. Thus, it suffices to show that if $p(n-1) \neq p^*$,

$$\left|p^t(n) - p^{*t}\right| < \left|p^t(n-1) - p^{*t}\right| \text{ for all } n \geq 1. \tag{20}$$

This can be easily shown as follows. Let us discuss two separate cases.

1. $\lambda(n) < \lambda^*$, i.e., $p^t(n) < p^{*t}$:
   In this case we have from (15)

   $$p^t(n+1) = p^t(n) + \frac{\check{p}^t(n+1) - p^t(n)}{M} > p^t(n),$$

   where the last inequality follows from assumption (A1). Hence, we only need to show that

   $$2p^{*t} - p^t(n) - p^t(n+1) > 0.$$

From (15) we have

$$\begin{aligned} & 2p^{*t} - p^t(n) - p^t(n+1) \\ = \; & 2p^{*t} - p^t(n) - \frac{\check{p}^t(n+1)}{M} - \frac{M-1}{M}p^t(n) \\ = \; & 2p^{*t} - \frac{2M-1}{M}p^t(n) - \frac{\check{p}^t(n+1)}{M} \end{aligned} \tag{21}$$

Multiply (21) by M.

$$\begin{aligned} \Rightarrow \; & 2Mp^{*t} - (2M-1)p^t(n) - \check{p}^t(n+1) \\ = \; & 2M(p^{*t} - p^t(n)) + p^t(n) - \check{p}^t(n+1) \\ > \; & 0 \text{ from assumption (A2).} \end{aligned}$$

2. $\lambda(n) > \lambda^*$, i.e., $p^t(n) > p^{*t}$:
We know from (15)

$$p^t(n+1) = p^t(n) + p^t(n) + \frac{\check{p}^t(n+1) - p^t(n)}{M} < p^t(n),$$

where the last inequality follows from assumption (A1). Hence, we only need to show that

$$2p^{*t} - p^t(n) - p^t(n+1) < 0.$$

From (15) we have

$$\begin{aligned} & 2p^{*t} - p^t(n) - p^t(n+1) \\ = \; & 2p^{*t} - p^t(n) - \frac{\check{p}^t(n+1)}{M} - \frac{M-1}{M}p^t(n) \\ = \; & 2p^{*t} - \frac{2M-1}{M}p^t(n) - \frac{\check{p}^t(n+1)}{M} \end{aligned}$$

$$\begin{aligned} \Rightarrow \; & 2Mp^{*t} - (2M-1)p^t(n) - \check{p}^t(n+1) \\ = \; & 2M(p^{*t} - p^t(n)) + p^t(n) - \check{p}^t(n+1) \\ < \; & 0 \text{ from assumption (A2).} \end{aligned}$$

The convergence of $p(n)$ follows from that $\check{p}^t(p)$ is a continuous function of p and assumption (A2).

III. Proof of Theorem 4

Suppose that p^1 and p^2 are two NEPs. We consider the following two cases.

1. $x^1 = x^2$, where $x^1 = x(p^1)$ and $x^2 = x(p^2)$.
2. $x^1 \neq x^2$.

We show that in the first case $p^1 = p^2$, and the second case leads to a contradiction.

1. $x^1 = x^2$:
Since $x_i = \frac{p_i}{\sum_j p_j} \cdot C$, the assumption tells us that $p^1 = \alpha \cdot p^2$ for some $\alpha > 0$. We show that $\alpha = 1$. Suppose that $\alpha < 1$. Then, $p_i^2 > P_{min}$ and $p_i^1 < P_{max}^i$ for all $i \in \mathcal{I}$. Hence, from the K-T conditions, we have

$$U_i'(x_i^2) \cdot \frac{\sum_{j \neq i} p_j^2}{(\sum_j p_j^2)^2} \geq 1 \tag{22}$$

and

$$U_i'(x_i^1) \cdot \frac{\sum_{j \neq i} p_j^1}{(\sum_j p_j^1)^2} \leq 1. \tag{23}$$

However, since $\alpha < 1$, for all $i \in \mathcal{I}$

$$\frac{\sum_{j \neq i} p_j^1}{(\sum_j p_j^1)^2} > \frac{\sum_{j \neq i} p_j^2}{(\sum_j p_j^2)^2},$$

which with (22) and (23) leads to a contradiction because we have assumed that $x^1 = x^2$. If we assume that $\alpha > 1$, then by a similar argument we get a contradiction. Thus, $\alpha = 1$ and $p^1 = p^2$.
2. $x^1 \neq x^2$:
We consider two subcases.
   (a) $\sum_j p_j^1 \geq \sum_j p_j^2$:
   Since $x^1 \neq x^2$, there exists a user $i \in \mathcal{I}$ such that $x_i^1 > x_i^2$. From the K-T conditions, we have

   $$U_i'(x_i^1) \cdot \frac{\sum_{j \neq i} p_j^1}{(\sum_j p_j^1)^2} \geq 1 \tag{24}$$

   and

   $$U_i'(x_i^2) \cdot \frac{\sum_{j \neq i} p_j^2}{(\sum_j p_j^2)^2} \leq 1. \tag{25}$$

   Since we have assumed that $x_i^1 > x_i^2$, from the strict concavity of the utility functions, $U_i'(x_i^1) < U_i'(x_i^2)$ and from (24) and (25)

   $$\frac{\sum_{j \neq i} p_j^1}{(\sum_j p_j^1)^2} > \frac{\sum_{j \neq i} p_j^2}{(\sum_j p_j^2)^2},$$

   which tells us

   $$\frac{C - x_i^1}{\sum_j p_j^1} > \frac{C - x_i^2}{\sum_j p_j^2}. \tag{26}$$

   Since we have assumed that $\sum_j p_j^1 \geq \sum_j p_j^2$, (26) implies $C - x_i^1 > C - x_i^2$, which is a contradiction.
   (b) $\sum_j p_j^1 < \sum_j p_j^2$:
   Since $x^1 \neq x^2$, there exists a user $j \in \mathcal{I}$ such that $x_j^1 < x_j^2$. Similarly as in the first subcase, this leads to a contradiction with the assumption.

This proves that if p^1 and p^2 are two NEPs, then $p^1 = p^2$ and there exists a unique NEP of the game.