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, ananth}@eecs.berkeley.edu

Abstract

In a communication network a good rate allocation algorithm should reflect the utilities of the users while being fair. We investigate this fundamental problem of achieving the system optimal rates in the sense of maximizing aggregate utility, in a distributed manner, using only the information available at the end hosts of the network. This is done by decomposing the overall system problem into subproblems for the network and for the individual users, by introducing a pricing scheme. The users are to solve the problem of maximizing individual net utility, which is the utility less the amount they pay. We provide algorithms for the network to adjust its prices and the users to adjust their window sizes such that at an equilibrium the system optimum is achieved. Further, the equilibrium prices are such that the system optimum achieves weighted proportional fairness. It is notable that the update algorithms of the users do not require any explicit feedback from the network, rendering them easily deployable over the Internet. Our scheme is incentive compatible in that there is no benefit to the users to lie about their utilities.

1 Introduction

As the Internet explodes in size and in the number of users, one of 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, most connections use variants of the Transmission Control Protocol (TCP), which is a window-based congestion control mechanism. It is widely recognized, however, that TCP does not generally lead to a fair or efficient allocation of bandwidth among the connections [3, 18, 5]. The fact that the Internet is now in the public domain, and thus in a potentially noncooperative environment, has stimulated much work on pricing mechanisms to ensure that users do not misbehave and to provide quality of service in accordance with users' willingness to pay. Researchers in this area have proposed different schemes based on time and volume measurements [11] or on per-packet charges [21]. Furthermore, research has shown that flat rate charging leads to inefficiency, where a large number of low-usage users end up subsidizing a small number of high-usage users [4]. This argues that usage-based pricing is desirable. A good discussion on the market structure is given in [16]. They present several different possible market structures for network pricing, such as a competitive market and monopoly, and provide insight as to how pricing schemes should be designed based on the market structure.

Recently Kelly [9] has suggested that the problem of rate allocation should be posed as one of achieving maximum aggregate utility for the users. These users are assumed to be of elastic traffic, and can adjust their rates based on the feedback from the network as TCP does. Since the solution needs to be decentralized and should be incentive compatible in that users should have no incentive to lie about their true utilities, Kelly proposes using pricing to decompose the overall system problem into subproblems for the network and for the individual users, with the goal of designing algorithms such that a system optimum is achieved when users' choices of rates they are willing to pay for and the network's choice of allocated rate are in equilibrium. Kelly *et al.*[10] have proposed two classes of algorithms which can be used to implement proportionally fair pricing and solve a *relaxation* of system optimum problem.

The algorithm in [10] 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 utility of the users, using only the information available

at the end hosts. We assume that the users are of elastic traffic and can adjust their rates based on their estimates of network congestion level. One example of such traffic is FTP users that can adapt to the network congestion. Since the total transfer time or per-transfer delay is determined by the rate the user receives, the user's utility function can be defined as a function of the rate. Another example may be web browsing, where users download a set of objects within a page. For these web browsing users they may have certain utility based on how quickly they can transfer a page and the objects in the page. It is well known in economics that a utility function can be used to capture a user's preferences. We introduce two algorithms that can be deployed over the Internet without significant modification within the network. These algorithms can be thought of as a natural extension of an algorithm introduced in [19]. These algorithms effectively decompose the system problem into a network problem and a problem for each user, by introducing a pricing mechanism. In the first algorithm, 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 second algorithm does not require users to solve the user optimization problem. Instead, it incorporates the user optimization problem into the window size updating rules. The unique fixed point of the mapping defined from these algorithms is proven to result in the system optimal rates. Further, the price a user pays equals the delay cost it imposes on the other users[1]. We show that our algorithms do not require any explicit feedback from the network, and can be easily deployed in the current Internet. We also demonstrate that these algorithms are incentive compatible: it is always in users' own interest to tell the truth about their utility functions. We present the simulation results obtained using the well known ns-2 simulator and demonstrate the convergence of our algorithms to the 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, followed by some related work. Section 4 explains the pricing scheme, which is followed by the algorithms in sections 5 and 6. We present two numerical examples in section 7. We briefly discuss how the backlog in the network can be controlled using our pricing scheme in section 8 and then finish by drawing some conclusions and indicating some directions for future research in section 9.

[1]This can be also interpreted as an externality cost imposed on the network.

2 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 our problem of maximizing the total utility of the users and describe the model we use to present our congestion control proposal.

2.1 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 [10, 19, 14, 8]. 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 [10] 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 the 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[2] as currently implemented suffers from a high packet loss rate and a delay bias [13, 18, 2]. The high packet loss rate is a consequence of periodic oscillation of the window sizes and aggressive slow start, while the delay bias is the result of a discrepancy in window update rates among different connections. In order to address these issues Brakmo *et al.*[2] 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 rate, 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[3] of the bandwidth. This was later proved by Mo *et al.*[18] in the case that there is a single bottleneck in the network.

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

[3]Fairness refers to max-min fairness that is discussed in section 2.2.

Mo and Walrand [19] 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[4] 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 ∞ [19].

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. As mentioned in section 1 users' preferences can be captured by appropriate 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.

2.2 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 $\mathcal{J}$.[5] 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 $U_i(x_i)$. For example, suppose that a user is transferring a file. The per-transfer delay is inversely proportional to the rate it receives. Hence, the utility of the user may be measured as a function of its rate. We assume that the utility $U_i(x_i)$ is an increasing, strictly concave and continuously differentiable function of x_i over the range $x_i \geq 0$.[6] Furthermore, we assume that the utilities are additive so that the aggregate utility of rate allocation $x = (x_i, i \in \mathcal{I})$ is $\sum_{i \in \mathcal{I}} U_i(x_i)$. This is a reasonable assumption since these utilities are those of independent network users. 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.

[4]Proportional fairness is introduced by Kelly in [9], and is discussed in the next subsection.

[5]We assume that there is a routing mechanism and do not discuss the issue of routing in this paper.

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

SYSTEM(U,A,C):

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

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 users' rates due to the large scale of the network. Hence, Kelly in [9] has proposed to consider the following two simpler problems.

Suppose that each user i is given the price per unit flow λ_i. Given λ_i, user i selects an amount to pay per unit time, p_i, and receives a flow $x_i = \frac{p_i}{\lambda_i}$.[7] Then, the user's optimization problem becomes the following [9].

$USER_i(U_i; \lambda_i)$:

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

The network, on the other hand, given the amounts the users are willing to pay, $p = (p_i, i \in \mathcal{I})$, 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 [9].

NETWORK(A,C;p) :

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

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 [9] that one

[7] This is equivalent to selecting its rate x_i and agreeing to pay $p_i = x_i \cdot \lambda_i$.

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 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 a smaller rate. In this sense, max-min fairness gives an absolute priority to the users with smaller rates. However, often in order to achieve a max-min fair allocation the optimality of the network needs to be sacrificed as discussed in [17]. In order to handle this tradeoff between fairness and optimality Kelly [9] has proposed another definition of fairness. A vector of rates $x' = (x_i', i \in \mathcal{I})$ is said to be *weighted proportionally fair*[8] 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:

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

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

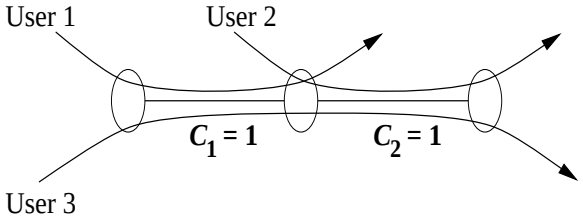


Figure 1: Max-min and proportional fairness.

Let us demonstrate the differences between max-min and proportional fairness by an example. Clearly max-min and proportional fairness are the same in a single link case. Thus, we consider a multiple link case. Consider the example in Figure 1. There are three users that share the network with two links. One can easily see that the unique max-min fair allocation in this example is $(x_1, x_2, x_3) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and every user receives the same rate. By solving *NETWORK(A, C; $\underline{1}$)*, where $\underline{1}$ is a vector, whose elements are all 1, one can show that the proportionally fair allocation

[8]This is called *rates per unit charge are proportionally fair* in [9]

is $(x_1, x_2, x_3) = (\frac{2}{3}, \frac{2}{3}, \frac{1}{3})$. Note that the sum of rates received from all links is the same for all users under the proportionally fair allocation.

3 Related Work

In this section we briefly describe some of the past work we draw upon and discuss how it is related to the problem we address in this paper.

3.1 Rate-based Algorithms

Based on the idea of proportional fairness Kelly *et al.*[10] 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 p_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 $\mu_j(t) = b_j(\sum_{i:j\in\mathcal{J}_i} x_i(t))$, where $b_j(\cdot)$ is an increasing function of the total rate going through it. Consider the system of differential equations

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

where

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

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 p_i, under some conditions on $b_j(\cdot), j \in \mathcal{J}$, the above system of differential equations converges to a point that maximizes the following expression

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

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

$$p_i(t) = x_i(t) \cdot U_i'(x_i(t))$$

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

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

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.

3.2 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 [2] and the algorithm proposed by Mo and Walrand [19] use the estimated queue size to adjust the congestion window size[9].

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 [5, 6, 18]. In order to deal with these issues Mo and Walrand [19] 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 a 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:

$$A^T x - C \leq 0 \tag{6}$$

$$Q(A^T x - C) = 0 \tag{7}$$

$$X(AC'q + d) = w \tag{8}$$

$$x \geq 0, q \geq 0, \tag{9}$$

where

$$C = (C_1, ..., C_J)^T, q = (q_1, ..., q_J)^T,$$

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

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

w_i is the congestion window size of user i, $\bar{d}_i$ is the propagation delay of route $\mathcal{J}_i$ not including the queueing delay, 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 the fact that the window size of connection i should equal the sum of the amount of fluid in transmit 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 is assumed to be sufficiently large, 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 [19] 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 [19] each connection has a target queue size $p_i > 0$ and attempts to keep p_i packets at the switch buffers. Let $d_i(t), w_i(t)$, and $x_i(t)$[10] 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.

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

Suppose that users update their window sizes 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.

4 Pricing Scheme for Charge-Sensitive TCP

The algorithms suggested in [10, 7] 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 long term 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.

4.1 Pricing Scheme

Section 3.2 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. 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[11] 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. In other words, the queueing delay caused by a user can be interpreted as the delay cost it imposes on other users [16].

Suppose that the network attempts to recover the increase in 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 per unit time[12] 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 per unit time 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

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

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

[12] Throughout the rest of the paper we refer to the price per unit flow per unit time and price per unit time as the price per unit flow and price, respectively, when there is no confusion.

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 the $(p,1)$-proportionally fair algorithm or 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 [2, 18]. 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 the $(p,1)$-proportionally fair algorithm, where the resulting rates are weighted proportionally fair with the 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 the $(p,1)$-proportionally fair algorithm for a fixed p, the price of a user equals its target queue size, and the user can infer its own price, h_i, from its target queue size p_i. Hence, a user can use its target queue size to indicate or implicitly communicate its willingness to pay.[13] The fact that users can estimate their own prices eliminates the need for any explicit feedback from the network. One thing to note is that the rate of a user depends not only on its own target queue size, but also on those of other users. Hence, the prices per unit flow at the resources depend on the congestion level in the network.

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 the fact that the price user i pays per

[13] In the following sections we use willingness to pay and target queue size interchangeably.

unit time equals its target queue size p_i and the total system cost per unit time is given by

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

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

In the following two sections, based on this pricing mechanism, we propose two algorithms that can be deployed over the current Internet without an extensive modification inside the network. All that is required is a modification 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 $SYSTEM(U, A, C)$.

5 Algorithm I

In the previous section 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 section 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)*.

The assumptions we make on the utility functions throughout the rest of paper are given in Appendix A. These assumptions are satisfied by the most commonly used utility functions such as $U(x) = a \cdot \log(x + b)$ and $U(x) = c \cdot x^d$, where $a > 0$, $0 \leq b \leq 1$, $c > 0$, and $0 < d < 1$ are arbitrary constants.

Suppose that ϵ is some small positive constant and users update their willingness to pay or target queue size $p_i, i \in \mathcal{I}$, 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^- and $\hat{p}_i(\lambda) = \arg\max_{p_i} U_i(\frac{p_i}{\lambda}) - p_i$. We assume that the target queue size updates take place on 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^-, $\lambda_i(t)$, user i computes its optimal target queue size 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 to some small positive constant ϵ that is arbitrarily small, so that each user can estimate its well defined price per unit flow along the route, which is strictly positive. We now consider the limit as $\epsilon \downarrow 0$, where (12) can be written as

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

where $U_i'(\cdot) = \frac{\partial U_i(\cdot)}{\partial x_i}$ and $K_i = \lim_{\lambda \downarrow 0} \hat{p}_i(\lambda)$. From Assumption 1 in Appendix A K_i is well defined and greater than 0. We assume that $K_i < \infty$ for all $i \in \mathcal{I}$. This is a reasonable assumption because at *very* high rates users' marginal utility is likely to be very small and users are not likely to pay a price larger than a certain limit, for instance, given by a budget constraint.[14] 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 the $USER_i(U_i; \lambda_i)$ 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 target queue size to such p_i.

Equation (13) implicitly assumes that a user does not anticipate the effect of its own action on prices per unit flow [10] 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 per unit flow. This is discussed in [12].

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

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

[14] These very high rates are assumed to be much larger than the capacity of the routes. Since the rates of the users are constrained by the capacity of their respective routes, these very high rates not of real importance.

where right-hand side is given by (13), $\mathcal{P} = \Re_+^{\mathrm{I}}$, and $\lambda(p)$ is the price per unit flow vector at the unique stable point of the (p,1)-proportionally fair algorithm with target queue size vector p. A fixed point p^* of the mapping $\mathcal{T}$ is a vector in $\mathcal{P}$ such that $\mathcal{T}(p^*) = p^*$.[15] Note from (13) that at any such fixed point p we have $\lambda_i(p) > 0$ for all $i \in \mathcal{I}$. This implies that if $p_i = 0$ for user $i \in \mathcal{I}$, then $x_i(p) = 0$.

Theorem 1 *Suppose that the utility functions satisfy Assumption 1 in Appendix A. Then, 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 B. ■

Theorem 1 tells us that when the users adopt the algorithm given in this section, 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 flow control algorithm in a distributed environment with no additional feedback from the network.

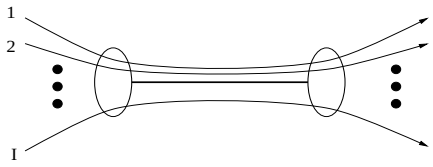


Figure 2: Single bottleneck case.

A natural question that arises now is whether or not the user target queue sizes $p_i, i \in \mathcal{I}$, converge to the unique fixed point of the mapping $\mathcal{T}$. Due to the complex relationship between the target queue size vector and the resulting rate allocation, showing the convergence of users target queue sizes in a general network is a challenging problem. In this section we only show the convergence of user target queue sizes in the case of a single bottleneck link, i.e., every user has only one and the same bottleneck link. In such a single bottleneck case the price per unit flow is

[15] Any fixed point of this mapping is a 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 $\epsilon(n) \downarrow 0$. Here the $\mathcal{T}_{\epsilon(n)}$ describes the systems in which connections continue to probe the network to estimate the congestion level, while $\mathcal{T}$ describes the system in which connections might not be able to probe the network. Thus, with the fixed points of $\mathcal{T}$ being approximated by those of $\mathcal{T}_{\epsilon(n)}$ one can pretend that even for $\mathcal{T}$ the connections are aware of the network conditions, for example, through a modified 'keepalive' mechanism in TCP [22].

same for all users. Hence, each user updates its target queue size based on the current price per unit flow of the system. Suppose that the current target queue size 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 target queue size vector is such that $p^t(0) = \sum_i p_i(0) > 0$. This ensures that $p(n) \neq 0$ for all $n \geq 1$.

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 target queue size based on its rate $x_i(n-1)$ and previous target queue size $p_i(n-1)$ as follows:

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

where $\check{p}_i(n) = \arg\max_{p_i \in \mathcal{P}_i} U_i(\frac{p_i}{\lambda(n-1)}) - p_i$, $\lambda(n-1) = \frac{p^t(n-1)}{C}$, and $p^t(n-1) = \sum_i p_i(n-1)$. Once every user finishes updating its target queue size, 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 target queue sizes $p_i(n)$. This is called the Jacobi update scheme.

The following theorem tells us that the user target queue sizes $p(n)$ converge to p^* under the Jacobi update scheme.

Theorem 2 *Suppose that the utility functions satisfy Assumptions 1 and 2 in Appendix A. In a single bottleneck case the user target queue sizes* $p(n) = (p_1(n), ..., p_I(n))$ *converge to* $p^* = (p^*_1, ..., p^*_I)$ *as* $n \to \infty$ *under the Jacobi update scheme, i.e.,*

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

Proof: The proof is given in Appendix C. ∎

In practice, however, the network users are almost never synchronized. Hence, it is important to show the convergence of user target queue sizes under an asynchronous update scheme with possibly delayed information. In other words, users do not necessarily update their target queue sizes simultaneously and 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 target queue size, and

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

where $0 \leq \tau_i(n) \leq n$. 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* [1].

Theorem 3 *Suppose that the utility functions satisfy Assumptions 1 and 2 in Appendix A. In a single bottleneck case the user target queue sizes* $p(n) = (p_1(n), ..., p_I(n))$ *converge to* $p^* = (p_1^*, ..., p_I^*)$ *as* $n \to \infty$ *under the totally asynchronous update scheme.*

Proof: The proof is given in Appendix D. ∎

A numerical example of the convergence of user target queue sizes to the unique fixed point of the mapping $\mathcal{T}$ with utility functions of $U(x_i) = a_i \cdot \log(x_i + b_i), a_i > 0$ and $0 \leq b_i \leq 1$ is given in section 7.

6 Algorithm II

The algorithm described in section 5 assumes that the users[16] explicitly compute the optimal target queue sizes based on the current prices per unit flow and use the (p, 1)-proportionally fair algorithm to solve the *NETWORK(A, C; p)* problem with the given target queue sizes $p = (p_i, i \in \mathcal{I})$. In other words, the problem is formulated as a discrete model where, given the prices per unit flow from the previous period $n-1$, users solve $USER(U_i; \lambda_i(n-1))$ for period n. Then the (p, 1)-proportionally fair algorithm is used to drive the users to the solution of *NETWORK(A, C; p(n))*. However, there are a few implementation issues that need to be addressed. First, since (p, 1)-proportionally fair algorithm converges asymptotically, the users need to know how long they should wait before updating their target queue sizes again or how often they should update their target queue sizes. Second, even with increasing computational power, it may still require non-negligible amount of CPU time to solve $USER(U_i; \lambda_i)$.

[16] In practice an agent at the end host may carry out the computation on behalf of the user.

In this section we introduce an algorithm that does not require a computation of the optimal target queue sizes, but the users' preferences are implicitly reflected in the window size updating rule. Suppose that the users update their window sizes according to the following system of differential equations:

$$\frac{dw_i(t)}{dt} = -\kappa \cdot M_i(t) \cdot r_i(t) \tag{16}$$

where

$$M_i(t) = \frac{d_i + U_i'(x_i(t)) + x_i(t) \cdot U_i''(x_i(t))}{d_i(t)}, \tag{17}$$

$U_i'(\cdot) = \frac{d}{dx_i}U_i(\cdot), U_i''(\cdot) = \frac{d^2}{dx_i^2}U_i(\cdot)$, $d_i(t) = \frac{w_i(t)}{x_i(t)} = d_i + \sum_{j \in \mathcal{J}_i} \frac{q_j(t)}{C_j}$, $q_j(t)$ is the backlog at resource j at time t, and

$$\begin{aligned} r_i(t) &= \frac{w_i(t) - x_i(t)d_i - x_i(t)U_i'(x_i(t))}{w_i(t)} \\ &= 1 - \frac{x_i(t)d_i}{w_i(t)} - \frac{x_i(t)U_i'(x_i(t))}{w_i(t)}. \end{aligned} \tag{18}$$

Note that user i's utility function appears in both $M_i(t)$ and $r_i(t)$.

The following theorem states that the algorithm given by (16) - (18) converges to the unique stable point, where the resulting rates solve the *SYSTEM(U, A, C)* problem.

Theorem 4 *Suppose that the utility functions satisfy Assumption 3 in Appendix A. Let* $V(w) = \frac{\sum_i (r_i(w))^2}{2}$. *Then* V *is a Lyapunov function for the system of differential equations (16) - (18). The unique value minimizing* V *is a stable point of this system, to which all trajectories converge.*

Proof: The proof is given in Appendix D. ∎

In sections 5 and 6 for the purpose of analysis we have assumed that users are delay insensitive and their utilities depend only on the rates they receive. This is a reasonable assumption since the users are concerned mostly with the overall (per-transfer) delay rather than the delay of individual packets. However, if it turns out that the network users 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 is the optimal rate allocation.

7 Numerical Examples

In this section we give a numerical example of a simple network for each algorithm and demonstrate the convergence of users parameters through simulations. The simulations are run using the ns-2 simulator developed at Lawrence Berkeley Laboratory (LBL) and UC Berkeley.

7.1 Algorithm I

Although the convergence results for the first algorithm 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 Assumptions 1 and 2 in Appendix A, and an appropriate damping constant M.

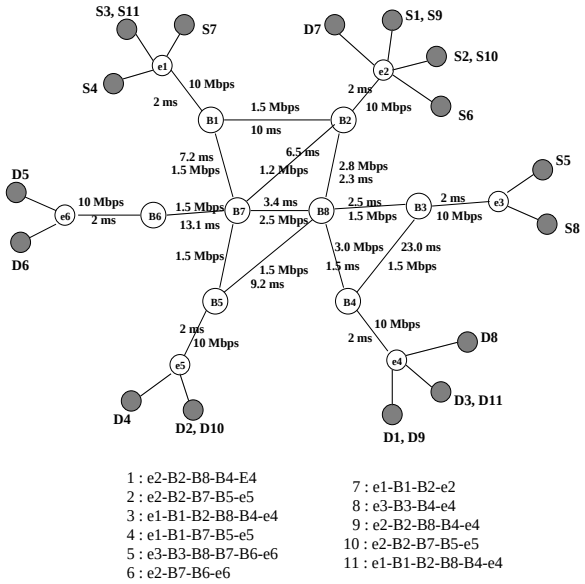


Figure 3: Topology of the simulated network and the users' routes.

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 the delays of the links are indicated by the numbers next to the links. The utility functions of the users are given in Table 1. Each connection updates its target queue size after it receives the acknowledgment for the k_i-th packet after the previous update. Hence, the target queue size updates are not synchronized due to different round trip delays. For the simulation we have set $k_i = 25$, $\kappa = 0.1$, and the damping constant $M = 5$ for all connections. Packet sizes are fixed at 1,000 bytes.

The target queue sizes 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

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

clearly demonstrate the convergence of the users' target queue sizes and, thus, the rates to the system optimum.

7.2 Algorithm II

The topology of the second simulated network is given in Figure 5. There are 18 users that share the network, and the routes of the users are as listed in Figure 5. The capacities and delays of the links are given next to the links in the figure. Table 2 has the utility functions, optimal rates, and equilibrium prices (or target queue sizes in packets) of the users.[17] In this simulation each connection measures its rate, computes $M_i(t)$ and $r_i(t)$, and updates its window size according to (16) - (18) once per round trip time. The parameter κ is set to 0.1 for all connections. Packet sizes are fixed at 1,000 bytes. The simulation is run for 20 seconds.

The evolution of the rates and target queue sizes of users 1, 3, 5, 7, and 11 are plotted in Figure 6. One can see that users' rates converge within seconds to a region around the optimal rates and the target queue sizes converge to the unique equilibrium. Since the users use instantaneous values of the rates and the queueing delays, their rates oscillate slightly around the optimal rates.

The convergence rate of the algorithm obviously depends on the parameters such as κ. Further,

[17] The rates x in $U_i(x)$ are in packets per second, while x_i^*'s are given in Mbps.

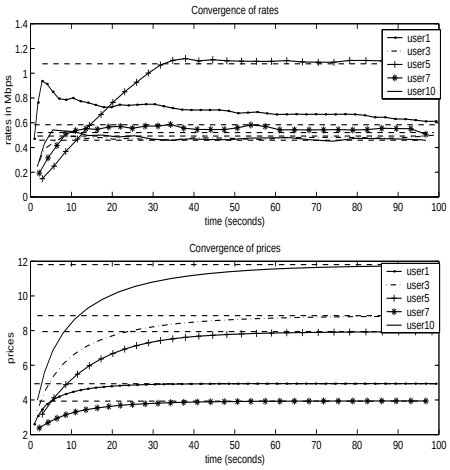


Figure 4: Convergence of user target queue sizes and rates.

simulation results indicate that the convergence rate of the algorithm decreases with the round trip delays of the users as expected. This is because the window sizes of the users are updated only once per round trip time. Suppose that a user enters a network with many users that is already in an equilibrium with the existing users. Then, the simulation results suggest that with a wide range of round trip delays and a reasonable choice for the value of κ the system takes no more than a few seconds to converge to a small neighborhood of the new equilibrium. This fast convergence can be explained from the fact that the arrival of a new user does not perturb the existing users too much, and this allows the new user to quickly estimate the price per unit flow of its route and reach the equilibrium. Due to the size of the Internet and a large number of users, if the arrivals and departures of the users are reasonably frequent, we expect the system to remain close to an equilibrium at any given time.

8 Controlling the Backlog and Queueing Delay in the Network

In the previous sections for simplicity of analysis we have assumed that the buffer sizes at the resources are infinite. In the real network, however, the buffer sizes are finite and there may be packet losses due to temporary buffer overflows. In this section we discuss how our pricing scheme and algorithms can be extended to cope with these packet losses.

In our pricing scheme we have assumed that the price per unit flow charged by a resource j is the queueing delay at the resource. Suppose that the price per unit flow at a resource is proportional to the queueing delay, i.e., $g_j = \gamma \cdot \frac{q_j}{C_j}$, where $\gamma > 0$, but not necessarily equal to the queueing delay. In this case the users face the same user optimization problem, except that now the target queue

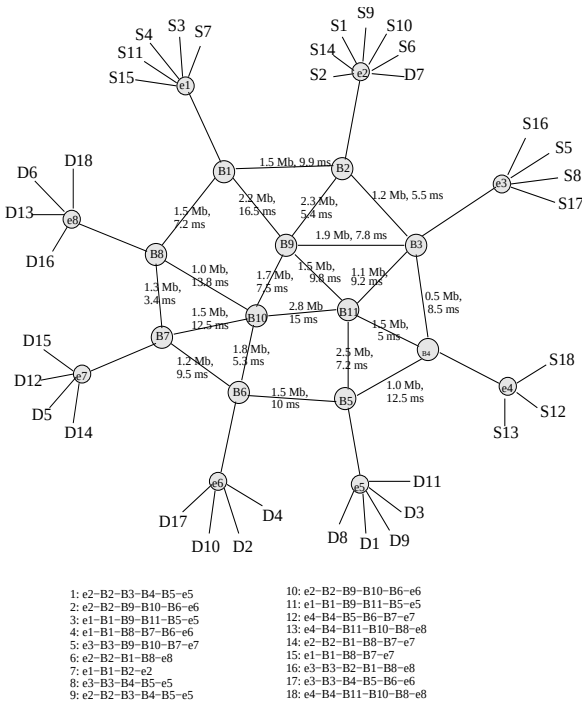


Figure 5: Topology of the simulated network and the users' routes.

size should be set to its willingness to pay divided by γ. More specifically, $U_i'(x_i(t))$ and $U_i''(x_i(t))$ in (17) and (18) need to be replaced by $\gamma^{-1} \cdot U_i'(x_i(t))$ and $\gamma^{-1} \cdot U_i''(x_i(t))$, respectively. One can show that under this more general pricing scheme and modified algorithms, there exists a unique equilibrium of the algorithms, and at the equilibrium the resulting rates are the optimal rates that solve *SYSTEM(U, A, C)*. In other words, the equilibrium prices and resulting rates are invariant under change of γ. One consequence of this is the following. Because the equilibrium prices are the same for all $\gamma > 0$, the target queue sizes of the users are inversely proportional to γ. Therefore, by increasing γ we can arbitrarily reduce the target queue sizes of the users and, hence, the backlog inside the network. Further, as γ goes to ∞, one can see that the queueing delay goes to 0 for all users because the target queue sizes of the users go to 0. What this means in practice is that γ can be chosen to ensure that packet losses are negligible even when the buffer sizes are finite.

We demonstrate this through simulation. Simulations are run with the same network topology and set of users as in section 7.2. We set γ to 3 and 0.3 in the first and second simulation, respectively. The simulations are run for 20 seconds. Figure 7 shows the rates and target queue sizes of the users. One can easily see that users' rates converge to similar values, while the target queue sizes are inversely proportional to γ, i.e., the product of γ and target queue size of each user remains constant. This is consistent with the claim made above.

user	$U_i(x)$	x_i^* (Mbps)	p_i^*
1	$2.4x^{0.7}$	0.121	11.25
2	$0.9x^{0.7}$	0.497	11.34
3	$0.9x^{0.7}$	0.750	15.13
4	$1.2x^{0.7}$	0.180	7.43
5	$0.9x^{0.7}$	0.497	11.34
6	$1.5x^{0.7}$	0.380	15.66
7	$0.6x^{0.7}$	1.500	16.38
8	$2.5x^{0.7}$	0.138	12.85
9	$2.4x^{0.7}$	0.121	11.25
10	$2.4x^{0.7}$	0.706	16.11
11	$0.9x^{0.7}$	0.750	15.16
12	$0.9x^{0.7}$	0.500	15.18
13	$1.2x^{0.7}$	0.500	11.39
14	$0.9x^{0.7}$	0.380	15.66
15	$1.5x^{0.7}$	0.180	7.43
16	$0.6x^{0.7}$	0.380	15.66
17	$2.5x^{0.7}$	0.121	11.25
18	$2.4x^{0.7}$	0.500	11.39

Table 2: Utility functions of the users.

9 Conclusions

In this paper we have investigated the fundamental problem of the existence of an algorithm that achieves the system optimum in a distributed network without any explicit feedback from the network elements to the end hosts. We have described a pricing scheme and two algorithms that solve the system problem at the unique equilibrium point of the algorithms. We have demonstrated that our algorithms are incentive compatible and do not require any feedback from the network, making them easy to deploy. Moreover, they achieve weighted proportional fairness. We have proved the convergence of the first algorithm to the optimal rates in a single bottleneck case under both synchronous and asynchronous update schemes under some mild technical assumptions on the utility functions. The second algorithm is proven to converge to the system optimum in any general

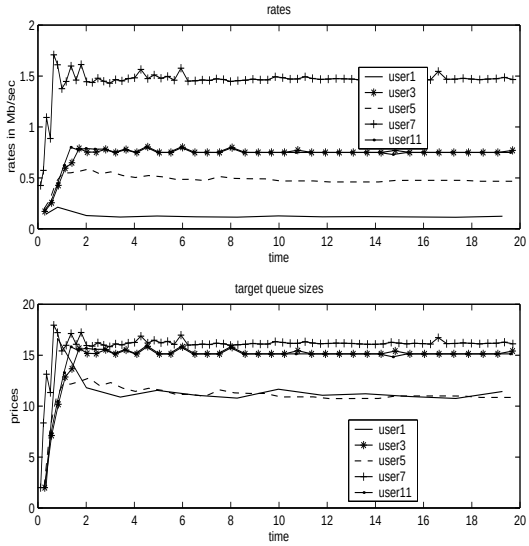


Figure 6: Convergence of user target queue sizes and rates.

network. A few numerical examples are given to demonstrate the convergence of the algorithms in simple networks.

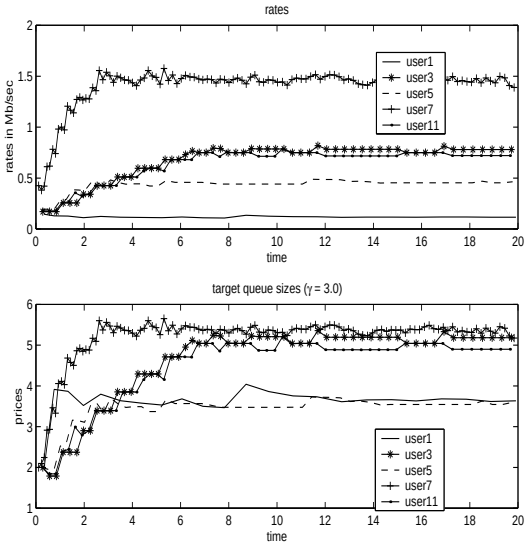
In this paper we have assumed that the routes of the users are fixed. However, if the network charges the users based on the congestion level along the route of the connection, the network should ensure that no users get preferential treatment or penalized through the selection of the routes. For instance, the routing has to be done in such a way that no user is routed through a path that has the highest price per unit flow among the available paths, while other connections are routed through the cheapest path.

We are currently attempting to design a mechanism that uses measured packet loss rates to solve the *SYSTEM(U, A, C)* problem, instead of the queueing delays. We are also working on how one can solve a system problem with a different objective function, such as the total revenue, using a distributed window-based algorithm. Another interesting question that remains is how the algorithm described in this paper can be extended to provide different quality of service (QoS).

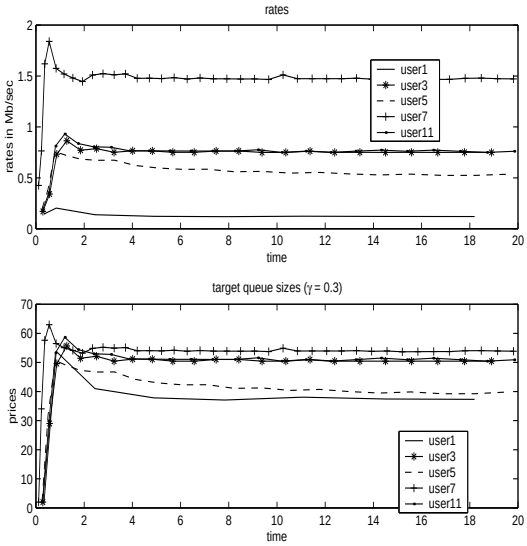
References

[1] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation.* Prentice-Hall, Englwood Cliffs,N.J., 1989.

[2] 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.



(a)



(b)

Figure 7: Convergence of user target queue sizes and rates with various γ values. (a) $\gamma = 3.0$ (b) $\gamma = 0.3$.

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

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

[5] 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.

[6] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[7] 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.

[8] V. Jacobson. Congestion avoidance and control . *Computer Communication Review*, 18(4):314–29, August 1988.

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

[10] 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.

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

[12] R. J. La and V. Ananthram. Charge-Sensitive TCP and Rate Control in the Internet. In *Proc. IEEE INFOCOM '00*, March 2000.

[13] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–50, June 1997.

[14] S. H. Low and D. E. Lapsley. Optimization flow control. *IEEE/ACM Transactions on Networking*, 7(6):861–74, Dec. 1999.

[15] D. Luenberger. *Introduction to Dynamic System.* Wiley, New York, NY, 1979.

[16] J.K. Mackie-Mason and H. R. Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications*, 13(7):1141–9, September 1995.

[17] L. Massoulie and J. Roberts. Bandwidth sharing : objectives and algorithms. In *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

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

[19] J. Mo and J. Walrand. Fair End-to-End Window-Based Congestion Control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[20] W. Rudin. *Principles of Mathematical Analysis, 3rd Edition.* McGraw-Hill, 1976.

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

[22] G. R. Wright and W. R Stevens. *TCP/IP Illustrated, Volume 2.* Addison Wesley, 1995.

A Assumptions on the Utility Functions

Assumption 1 *$\hat{p}_i(\lambda_i)$ is decreasing in $\lambda_i > 0$, where $\hat{p}_i(\lambda_i) = \arg\max_{p_i} U_i(\frac{p_i}{\lambda_i}) - p_i$.*

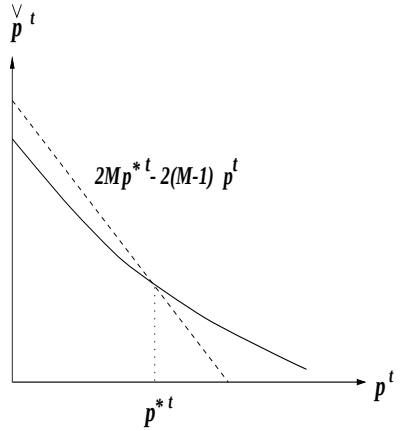


Figure 8: Assumption 2.

Assumption 2 *There exists $M > 0$ such that*

(i) for all $p^t < p^{*t}, \check{p}^t(p^t) - p^t < M(p^{*t} - p^t)$

(ii) for all $p^t > p^{*t}, \check{p}^t(p^t) - p^t > M(p^{*t} - p^t)$

where $p^t = \sum_i p_i, p^{*t} = \sum_i p_i^*, \check{p}^t(p^t) = \sum_i \check{p}_i(p^t)$, and $\check{p}_i(p^t) = \arg\max_{p_i'} U_i(\frac{p_i' \cdot C}{p^t}) - p_i'$.

Assumption 3 *Utility functions of the users satisfy the following:*

$$U_i'(x_i) + x_i \cdot U_i''(x_i) \geq 0, \;\; x_i \in [0, C^i],$$

where $C^i = \min_{j \in \mathcal{J}_i} C_j$.

B Proof of Theorem 1

At an equilibrium p^* of the algorithm, where users' willingness to pay stays 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, i.e., $\lambda^* > 0$.
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 the fact that $p_i^* = x_i^* \cdot \lambda_i^*$ by definition. (2) holds at the unique stable point of TCP with fixed p^* [19]. (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 [9], 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

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

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

$$\begin{aligned} x_i^* > 0 \quad &\Rightarrow \quad U_i'(\frac{p_i^*}{\lambda_i^*}) = U_i'(x_i^*) = \lambda_i^* \\ x_i^* = 0 \quad &\Rightarrow \quad U_i'(0) \leq \lambda_i^*. \end{aligned} \tag{20}$$

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

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

C Proof of Theorem 2

Note that, in a single bottleneck case, updating scheme of the users depends only on the current target queue size and price per unit flow $\lambda(n) = \frac{p_i(n)}{x_i(n)} = \frac{p^t(n)}{C}$, 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 $p(n-1) \neq p^*$,

$$\left|p^t(n) - p^{*t}\right| < \left|p^t(n-1) - p^{*t}\right| \text{ for all } n \geq 1. \tag{21}$$

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)

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

   where the last inequality follows from assumption 1. Now we show that

   $$p^t(n) < p^t(n+1) < p^{*t}$$

   as follows.

   $$\begin{aligned} p^t(n+1) &= \sum_{i\in\mathcal{I}} p_i(n) + \frac{\check{p}_i(n+1) - p_i(n)}{M+1} \\ &= \frac{M}{M+1} p^t(n) + \frac{1}{M+1}\check{p}^t(n+1) \\ &< \frac{M}{M+1} p^t(n) + \frac{1}{M+1}(p^{*t} + M \cdot (p^{*t} - p^t(n))) \qquad (22) \\ &= p^{*t}, \end{aligned}$$

   where the inequality in (22) follows from assumption 2.

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

   Using a similar argument, one can show that

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

The convergence of $p(n)$ follows from the fact that $\check{p}^t(p)$ is a continuous function of p and assumption 2.

D Proof of Theorem 3

We first show that there is a sequence of non-empty sets $\{\mathcal{P}(n)\}$ with

$$\cdots \subset \mathcal{P}(n+1) \subset \mathcal{P}(n) \subset \cdots \mathcal{P}(0) \subseteq \mathcal{P},$$

where $\mathcal{P} = \mathcal{P}_1 \times \cdots \times \mathcal{P}_I$, satisfying the following two conditions:

1. Synchronous convergence condition : we have

   $$\mathcal{T}(p) \in \mathcal{P}(n+1) \text{ for all } n \text{ and } p \in \mathcal{P}(n).$$

   Furthermore, if $\{p^k\}$ is a sequence such that $p^k \in \mathcal{P}(k)$ for every k, then every limit point of $\{p^k\}$ is the unique fixed point of the mapping $\mathcal{T}$.

2. Box condition : for every n, there exist sets $\mathcal{P}_i(n) \subset \mathcal{P}_i$ such that

   $$\mathcal{P}(n) = \mathcal{P}_1(n) \times \cdots \times \mathcal{P}_I(n).$$

From the assumption there exists at least one $i \in \mathcal{I}$ such that $U_i'(0) > \lambda_{max} = \frac{\sum_i P_{max}^i}{C}$. Thus, we know that $p^t(k) > 0$ for all $k \geq 1$.

Let $\mathcal{P}(0) = [\min\{p_i(0), \check{p}_i(\lambda_{max})\},\ p_{max}^i]$. Define for all $n \geq 1$, $\mathcal{P}'(n) = \{\, T'(p) \mid p \in \mathcal{P}(n\text{ - }1) \,\}$, where

$$T_i'(p) = p_i + \frac{\check{p}_i(p^t) - p_i}{M+1}.$$

Take the projection for each $i \in \mathcal{I}$

$$\mathcal{P}_i(n) = \{p_i \mid p_i \text{ is the } i\text{-th element of } p \in \mathcal{P}'(n)\},$$

and define

$$\mathcal{P}(n) = \times_{i \in \mathcal{I}} \mathcal{P}_i(n).$$

Then, $\mathcal{P}'(n) \subset \mathcal{P}'(n\text{ - }1)$ and, hence, $\mathcal{P}(n) \subset \mathcal{P}(n\text{ - }1)$. Further, one can show that $\mathcal{P}(n)$ satisfies the *synchronous convergence condition.* From its construction $\mathcal{P}(n)$ satisfies the *box condition.* The theorem follows from [1].

E Proof of Theorem 4

We first define interior and boundary points. An interior point is defined to be a window size vector, around which we can find an open ball such that all the points in the ball have the same set of bottlenecks. A window size vector, for which we cannot find such an open ball is said to be a boundary point. One can show that the boundaries between different regions of the window size space W with different sets of bottlenecks are smooth and partition the window size space W into finite regions corresponding to the different set of bottleneck links. This can be seen from the fact that the set of window size vectors that result in a rate vector x is given by $\mathcal{W}^{-1}(x) = \{w \mid \mathcal{W}(w) = x\}$, where $\mathcal{W}$ is the function defined in subsection 3.2. This is a positive cone of XA_B with vertex Xd.

We consider the closures of these regions in the partition of W. Consider a point t in an open interval of time, in which the trajectory of interest remains in one of these closures, i.e., an interior point. We show that for small $\epsilon > 0$,

$$V(t+\epsilon) - V(t) = r^T(t) J_r(t)(w(t+\epsilon) - w(t)) + o(\epsilon), \tag{23}$$

where $J_r(t)$ is the partial derivative

$$J_r(t) = \left[\frac{\partial r_i}{\partial w_j}, i, j \in \mathcal{I}\right],$$

which depends on the region in which $w(t+\epsilon)$ is.

We first state a claim that will be used in the proof.

Claim 1 *At an interior point w the partial derivative $J_x = \left[\frac{\partial x_i}{\partial w_j}, i, j \in \mathcal{I}\right]$ is given by*

$$J_x = \bar{D}^{-1}(I - XA_B(A_B^T X \bar{D}^{-1} A_B)^{-1} A_B^T \bar{D}^{-1}) \ , \tag{24}$$

where $\bar{D} = diag(\bar{d}_1, \cdots, \bar{d}_{\mathrm{I}})$ and $\bar{d}_i = d_i + A_B q_B(w)$ for all $i \in \mathcal{I}$.

Proof: We first show that the mapping $\mathcal{W}$ from the window sizes to the rates is differentiable at the interior points. Suppose that w is an interior point and $x = \mathcal{W}(w)$. Let $\mathcal{B}$ be the set of bottlenecks at w. We consider two cases.

Case (1) A_B has full rank, i.e., $rank(A_B) = \mathrm{B} = |\mathcal{B}|$, where A_B is the submatrix of A with columns corresponding to the bottlenecks. Let

$$g_B(q'_B, w) = A_B^T x(q'_B, w) - C_B, \tag{25}$$

where

$$x_i(q'_B, w) = \frac{w_i}{d_i + A_{i.}q'_B} \tag{26}$$

and $A_{i.}$ is the i-th row of A_B. We use q'_B to denote the queueing delays, not queue sizes that are denoted by q_B. We define

$$J_{g_q} = \left[\frac{\partial g_i}{\partial q'_j}, i, j = 1, \cdots, \mathrm{B}\right],$$

where

$$\begin{aligned}
\frac{\partial g_i}{\partial q'_j} &= \frac{\partial}{\partial q'_j}(\sum_m A_{m,i}x_m(q'_B, w) - C_i) \\
&= \frac{\partial}{\partial q'_j}(\sum_m A_{m,i}\frac{w_m}{d_m + \sum_k A_{m,k}q'_k}) \\
&= \sum_m A_{m,i}(\frac{\partial}{\partial q'_j}\frac{w_m}{d_m + \sum_k A_{m,k}q'_k}) \\
&= \sum_m A_{m,i}(-\frac{w_m}{(d_m + \sum_k A_{m,k}q'_k)^2})A_{m,j} \\
&= -\sum_m A_{m,i}A_{m,j}\frac{w_m}{(d_m + \sum_k A_{m,k}q'_k)^2} \\
&= -\sum_m A_{m,i}A_{m,j}\frac{x_m(q'_B, w)}{\bar{d}_m(q'_B)},
\end{aligned}$$

where $\bar{d}_m(q'_B)$ explicitly denotes the dependency of $\bar{d}_m$ on q'_B. This can be written in a matrix form as

$$J_{g_q} = -A_B^T X \bar{D}^{-1} A_B \ . \tag{27}$$

One can easily show that J_{g_q} is nonsingular because $A_B^T X \bar{D}^{-1} A_B$ is positive definite. Therefore, by the implicit function theorem, $q'_B(w)$ is differentiable [20], and so is $x(q'_B, w)$ from (26). Further, from the implicit function theorem,

$$\begin{aligned}
J_q(w) &= \left[\frac{\partial q'_j}{\partial w_i}, j = 1, \cdots, \mathrm{B}, i \in \mathcal{I}\right] \\
&= -(J_{g_q})^{-1} J_{g_w},
\end{aligned}$$

where

$$\begin{aligned}
J_{g_w} &= \left[\frac{\partial g_j}{\partial w_i}, j = 1, \cdots, \mathrm{B}, i \in \mathcal{I}\right] \\
&= A_B^T \bar{D}^{-1} \qquad \text{from (25)}.
\end{aligned}$$

Thus,

$$J_q(w) = (A_B^T X \bar{D}^{-1} A_B)^{-1} A_B^T \bar{D}^{-1}. \tag{28}$$

Case (2) $rank(A_B) = \gamma <$ B. Let $\bar{\mathcal{B}} \subset \mathcal{B}$ such that $rank(A_{\bar{B}}) = \gamma$ and $|\bar{\mathcal{B}}| = \gamma$. Then, applying the implicit function theorem to $g_{\bar{B}}$ as defined in (25) yields the differentiability of $x(q'_{\bar{B}}, w)$. One can show that $x(q'_{\bar{B}}, w)$ is the unique solution of g_B as follows. Since we delete dependent rows from A_B^T, if x is a solution of $A_{\bar{B}}^T x = C_{\bar{B}}$, then it is a also a solution of $A_B^T x = C_B$. Since the solution is unique, $x(q'_{\bar{B}}, w)$ is the solution of g_B. Therefore, the differentiability follows.

We proceed to prove the claim with the assumption that A_B has full rank. If not, we can take $A_{\bar{B}}$ as described before and prove the claim. Since A_B has full rank, we know that the queueing delays are well defined from [19]. First, from (8) for all $i \in \mathcal{I}$ we have

$$w_i = x_i(w) \cdot d_i + x_i(w) A_{i\cdot} q'_B(w). \tag{29}$$

If we differentiate (29) with respect to $w_j, j \in \mathcal{I}$,

$$\delta_{i,j} = (A_{i\cdot} q'_B(w) + d_i)(J_x)_{i,j} + x_i(w)(A_{i\cdot}(J_q)_{\cdot j}).$$

Again, note that J_q is well defined from (28). In a matrix form this is given by

$$I = \bar{D} J_x + X A_B J_q,$$

which yields

$$J_x = \bar{D}^{-1}(I - X A_B J_q).$$

Since $J_q = (A_B^T \bar{D}^{-1} X A_B)^{-1} A_B^T \bar{D}^{-1}$ from (28), we get

$$J_x = \bar{D}^{-1}(I - X A_B (A_B^T \bar{D}^{-1} X A_B)^{-1} A_B^T \bar{D}^{-1}).$$

■

We continue the proof of the theorem. If $w(t)$ is an interior point, then from the algorithm and that $x(w)$ is differentiable at interior points from the above claim, we have

$$\begin{aligned}
J_r &= DXW^{-2} - DW^{-1}J_x + XU'W^{-2} - W^{-1}(U' + XU'')J_x \\
&= DXW^{-2} - DW^{-1}\bar{D}^{-1} \\
&\quad + DW^{-1}\bar{D}^{-1}XA_B(A_B^T X \bar{D}^{-1} A_B)^{-1} A_B^T \bar{D}^{-1} \\
&\quad + XU'W^{-2} - W^{-1}(U' + XU'')\bar{D}^{-1} \\
&\quad + W^{-1}(U' + XU'')\bar{D}^{-1}XA_B(A_B^T X \bar{D}^{-1} A_B)^{-1} A_B^T \bar{D}^{-1} \qquad (30) \\
&= -W^{-1}XU''\bar{D}^{-1} + (D + U' + XU'')\bar{D}^{-2} A_B (A_B^T X \bar{D}^{-1} A_B)^{-1} A_B^T \bar{D}^{-1}, \qquad (31)
\end{aligned}$$

where $W = \text{diag}(w(t))$, $X = \text{diag}(x(t))$, $\bar{D} = \text{diag}(d_1(t), \cdots, d_I(t))$, $D = \text{diag}(d_1, \cdots, d_I)$, $U' = \text{diag}(U_1'(x_1(t)), \cdots, U_I'(x_I(t)))$, $U'' = \text{diag}(U_1''(x_1(t)), \cdots, U_I''(x_I(t)))$, and A_B is the submatrix of A with columns that correspond to bottlenecks at $w(t)$[18]. If $w(t)$ is a boundary point, then A_B is the submatrix of A with columns that correspond to bottlenecks at $w(t+\epsilon)$. The second inequality in (30) follows from the fact that $J_x = \bar{D}^{-1}(I - XA_B(A_B^T X\bar{D}^{-1}A_B)^{-1}A_B^T\bar{D}^{-1})$.

Fix $\epsilon > 0$. Define $\delta w(t,\epsilon) = \frac{w(t+\epsilon)-w(t)}{\epsilon}$. Then, since $J_r(\cdot)$ is well defined in $[t, t+\epsilon]$ if the path $w(t')$ follows the straight line between $w(t+\epsilon)$ and $w(t)$ and $V(\cdot)$ is a continuous function of window sizes, one can see that

$$\begin{aligned} V(t+\epsilon) - V(t) &= \int_t^{t+\epsilon} r^T(s) J_r(s) \delta w(t,\epsilon) ds \\ &= \int_t^{t+\epsilon} (r(t) + \Delta r(s))^T [J_r(t) + \Delta J_r(s)] \delta w(t,\epsilon) ds, \end{aligned}$$

where $\Delta r(s) = r(s) - r(t)$, $\Delta J_r(s) = J_r(s) - J_r(t)$, $s \in [t, t+\epsilon]$, and $J_r(\cdot)$ is defined as described above. Then,

$$\begin{aligned} V(t+\epsilon) - V(t) &= \int_t^{t+\epsilon} r^T(t)\ J_r(t)\ \delta w(t,\epsilon)\ ds \\ &\quad + \int_t^{t+\epsilon} r^T(t)\ \Delta J_r(s)\ \delta w(t,\epsilon)\ ds \\ &\quad + \int_t^{t+\epsilon} \Delta r^T(s)\ J_r(t)\ \delta w(t,\epsilon)\ ds \\ &\quad + \int_t^{t+\epsilon} \Delta r^T(s)\ \Delta J_r(s)\ \delta w(t,\epsilon)\ ds \\ &= r^T(t)\ J_r(t)\ (w(t+\epsilon) - w(t)) \\ &\quad + r^T(t) \cdot \int_t^{t+\epsilon} \Delta J_r(s) ds \cdot \delta w(t,\epsilon) \\ &\quad + \int_t^{t+\epsilon} \Delta r^T(s) ds \cdot J_r(t) \cdot \delta w(t,\epsilon) \\ &\quad + \int_t^{t+\epsilon} \Delta r^T(s)\ \Delta J_r(s) ds \cdot \delta w(t,\epsilon) \\ &= r^T(t)\ J_r(t)\ (w(t+\epsilon) - w(t)) + o(\epsilon). \end{aligned} \tag{32}$$

[18]For sufficiently small $\epsilon > 0$, $w(t+\epsilon)$ has the same bottlenecks as $w(t)$.

Hence, from (32) if $r(t) \neq 0$,

$$\begin{aligned}
\lim_{\epsilon \downarrow 0} \frac{V(t+\epsilon) - V(t)}{\epsilon} &= \lim_{\epsilon \downarrow 0} \frac{r^T(t) J_r(t)(w(t+\epsilon) - w(t))}{\epsilon} + \lim_{\epsilon \downarrow 0} \frac{o(\epsilon)}{\epsilon} \\
&= r^T(t) J_r(t) \dot{w}(t) \\
&= -\kappa r^T(t) J_r(t) M(t) r(t) < 0, \qquad (33)
\end{aligned}$$

where $M(t) = \bar{D}^{-1}(D + U' + X \cdot U'')$. The last inequality follows from the fact that $J_r(t)M(t)$ is a positive definite matrix for (30). This extends easily to the closure of any union of open intervals in which the trajectory remains in one of the closures of the partition.

Now consider a point t in the complement Ω of the closures of the union of open intervals in which the trajectory remains in the closure of a single region of the partition of W. Since Ω is open, there is an open interval O containing t and contained in Ω. In particular, for every $t' \in O$, the trajectory is chattering severely across different regions of W near t'. For each such point t' let $R(t')$ be the family of closures of regions to which the trajectory belongs at time t'. Note that $R(t')$ is not empty. For each closure K of a region in the partition of W, K must be missing from the family $R(t')$ for a dense set of times, and this set of times for missing K is open because K is closed. Thus, one can find a non-empty open interval O_1 such that the closure $\bar{O}_1 \subset O$ so that K is missing from $R(t'')$ for all $t'' \in \bar{O}_1$. Repeating this argument successively for all the closures K, one eventually obtain a non-empty open set O_k for which $R(t^*) = \emptyset$ for all $t^* \in O_k$, which is a contradiction. Therefore, $\Omega = \emptyset$, and V is decreasing everywhere as shown in (33). The rest of the proof follows from an argument similar to that of Lyapunov stability theorem [15]