A Two-Time-Scale Design for Edge-Based Detection and Rectification of Uncooperative Flows

X. Fan[*], K. Chandrayana, M. Arcak, S. Kalyanaraman, J. T. Wen
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180

Submitted to *The IEEE/ACM Transactions on Networking*

Abstract

Existing Internet protocols rely on cooperative behavior of end users. We present a control-theoretic algorithm to counteract *uncooperative* users which change their congestion control schemes to gain larger bandwidth. This algorithm *rectifies* uncooperative users; that is, forces them to comply with their fair share, by adjusting the prices fed back to them. It is to be implemented at the edge of the network (e.g. by ISPs), and can be used with any congestion notification policy deployed by the network. Our design achieves a separation of time-scales between the network congestion feedback loop and the price-adjustment loop, thus recovering the fair allocation of bandwidth upon a fast transient phase.

1 Introduction

In a network which does not differentiate among users, the equilibrium rate for any user is primarily decided by the congestion control being used [1]. With new software advancements, however, "*uncooperative*" users can change their congestion control schemes to gain more than their fair share of bandwidth, at the cost of cooperative users. This uncooperative behavior can lead to TCP unfriendliness, congestion collapse [3], [4] and, to a traffic-based denial-of-service to cooperative users [5], [6]. Detecting uncooperative users, and "*rectifying*" their flow rates to comply with cooperative rates, and thus, improving quality of service for individual flows becomes an important emerging problem in network management.

[*] Corresponding author. Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA. Tel.: +1-518-276-8205; fax: +1-518-276-6261.

Among rectification mechanisms proposed in the literature, the majority are "router-based" that is, they modify the router algorithm to detect and limit uncooperative flows, e.g. Active Queue Management (AQM) schemes or scheduling disciplines. In [3] and [4], the authors study AQM schemes, and investigate the effect of uncooperative flows on network throughput and loss rates. Flow Random Early Drop (FRED), a modified RED scheme, is proposed in [7] to detect uncooperative users, and to limit their rates by increasing their packet drop probabilities. In [8], the authors combine the BLUE queue management algorithm with a Bloom filter to detect and rate-limit uncooperative flows. Several other rate-based schemes are surveyed in [9]. Scheduling schemes, such as ack-spacing, have been suggested to manage uncooperative flows in [10].

More recently, *edge-based* price-adjustment mechanisms have been proposed in [11] and [12], which manage uncooperative flows only at edge routers. A significant advantage of this approach is that it does not require core network upgrades and can be implemented without performing per flow management at routers. By estimating each flow's incoming rate and using it to label flow's packet, the Core-Stateless Fair Queueing (CSFQ) algorithm in [11] computes the forwarding probability from link fair rate estimation. However, this design only applies to network in which all nodes implement Fair Queueing. In [12], the authors manage uncooperative flows by mapping their utility function to a specified target network behavior at the edge. This study, however needs to estimate the utility function to achieve this edge-based price adjustment, and is thus restricted to a specific form of TCP.

In this paper, we develop an edge-based price-adjustment algorithm using tools from control theory. Rather than address a specific protocol, we develop our design within the optimization framework of Kelly [1], [2], [13], [14], [15], which is applicable to diverse types of networks, and encompasses numerous protocols such as TCP Reno, TCP Vegas, FAST [16, 17] etc. Our algorithm recovers the cooperative share of bandwidth prescribed in Kelly's framework, with a new feedback loop implemented at the edge router, and, hence, referred to as the "edge supervisor". It detects uncooperative users by comparing their sending rates with "*audit*" rates calculated according to an ideal, cooperative, model, and increases their price feedback. Although in this design edge supervisor does perform per flow management by this price adjustment loop, core routers, which are in general more complex than edge routers, do not perform per flow management, and therefore the implementation complexity is significantly reduced. Our algorithm is independent of congestion notification policy deployed by the network, and thus, can be used with any Active Queue Management scheme, as well as Drop Tail queueing.

We design the price adjustment loop to evolve in a faster time-scale than the existing price feedback loop from the links, because, then, uncooperative flows are rectified during a fast transient phase, after which stability and convergence properties of the desired cooperative network model is recovered. Indeed, using tools from singular perturbations theory [18], [19, Chapter 11], we prove that the fast and slow feedback loops, when combined, ensure convergence of the sending rates to their cooperative values. The type of convergence established is "semi-global" [19], which means that any desired region of attraction can be achieved by increasing the feedback gain of the price-adjustment loop.

The paper is organized as follows: Section 2 overviews Kelly's primal and dual flow control algorithms. Section 3 studies the primal algorithm and presents our price adjustment design for uncooperative users. Section 4 extends this design to the dual algorithm. In Section 5, we implement our price adjustment algorithms in NS-2 and evaluate their performance for various single and multi-bottleneck topologies, for both marking and dropping congestion notification policies, and with and without AQM schemes. In particular, we show that given a standard network behavior like TCP-Friendliness, our algorithm forces uncooperative users to comply with their fair-share of the bandwidth. Conclusions are given in Section 6.

Notation: We denote by $R_+ = (0, \infty)$, and, by R_+^N vectors whose entries are in R_+. Given a function $f(x)$, its positive projection is defined as

$$(f(x))_x^+ := \begin{cases} f(x) & \text{if } x > 0, \text{ or } x = 0 \text{ and } f(x) \geq 0 \\ 0 & \text{if } x = 0 \text{ and } f(x) < 0. \end{cases}$$

If x and $f(x)$ are vectors, then $(f(x))_x^+$ is interpreted in the component-wise sense.

2 Overview of Kelly's primal and dual flow control algorithms

In Kelly's framework [1], network flows are modeled as the interconnection of users and communication links as shown in Figure 1. Packets from each users (with sending rate x_i) are routed through the links with the aggregate link rate

$$y = R_f x \tag{1}$$

where R_f is the forward routing matrix. Each link j has a fixed capacity c_j, and based on its congestion and queue size, a link price, p_j is computed:

$$p_j = h_j(y_j), \quad j = 1, \cdots, L. \tag{2}$$

The link price information is then sent back to each source with the aggregate source price,

$$q = R_b p. \tag{3}$$

where $R_b = R_f^T$, since the links only feed back price information to the users that utilize them.

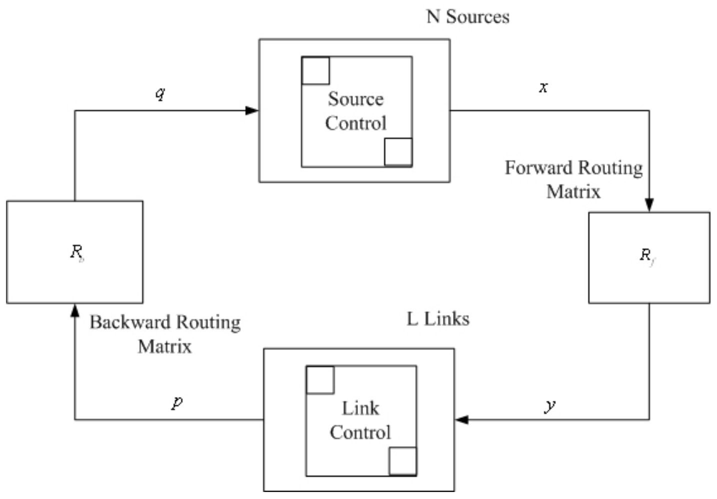


Figure 1: Network flow control model.

Kelly formulated the flow control task as a static optimization, and dynamic stabilization, problem. The static optimization problem computes the desired equilibrium by maximizing the sum of the source utility functions $U_i(x_i)$, while complying with capacity constraints in the links:

$$\max_{x\geq 0}\sum_{i=1}^{N} U_i(x_i) \quad \text{subject to } \underbrace{Rx}_{y} \leq c. \tag{4}$$

The dynamic problem is to design the source rate update law based on the aggregate price, and the link price update law based on the aggregate rate, to guarantee stability of the equilibrium. For this problem, Kelly introduced two dynamic algorithms: The *Primal Algorithm* consists of a first order source update law, and a static penalty function for the link to keep the aggregate rate below its capacity:

$$\dot{x}_i = \kappa_i \left(U_i'(x_i) - q_i\right), \quad p_j = h_j(y_j). \tag{5}$$

The penalty functions $h_l(y_l)$ are designed to enforce the link capacity constraints $y_l \leq c_l$, $l = 1,\cdots,L$, i.e., to keep the aggregate rate y_l below its capacity c_l.

The *Dual Algorithm* consists of a static source update and a first order dynamic price update:

$$x_i = U_i'^{-1}(q_i), \dot{p}_j \quad = \gamma_j (y_j - c_j)^+_{p_j} \tag{6}$$

From (6), the unique equilibrium for the dual control law is obtained from the equations

$$q_i^* = U_i'(x_i^*), \quad i = 1,\cdots,N \tag{7}$$

$$p_l^* \begin{cases} = 0 & \text{if } y_l^* \leq c_l \\ \geq 0 & \text{if } y_l^* = c_l \end{cases} \qquad l = 1, \cdots, L, \tag{8}$$

which as shown in [1], correspond to the solution of the optimization problem (4), in which p_l's play the role of Lagrange multipliers for the capacity constraints. For the primal control law (5), the equilibrium obtained from

$$q_i^* = U_i'\left(x_i^*\right), \quad i = 1, \cdots, N \tag{9}$$

$$p_l^* = h_l\left(y_l^*\right) \quad l = 1, \cdots, L, \tag{10}$$

approximates the optimality condition (7)-(8) with the help of the penalty functions $h_l\left(y_l\right)$. The stability of these two algorithms and their extensions has been established in [2], [13], [20], [21], [22], [14], [15], [17], [24].

3 Uncooperative users in Kelly's primal algorithm

We now assume that some users, which we call "uncooperative", use more aggressive utility functions to increase their share of bandwidth; that is, instead of $U_i\left(x_i\right)$ in (5), they implement $\tilde{U}_i\left(x_i\right)$:

$$\dot{x}_i = \kappa_i\left(\tilde{U}_i'\left(x_i\right) - \tilde{q}_i\right). \tag{11}$$

To rectify these uncooperative users, we propose that the supervisor at the edge of the network (e.g., internet service providers) adjusts the price feedback from its nominal value q_i to $\tilde{q}_i$. An ideal design of $\tilde{q}_i$ would be

$$\tilde{q}_i = q_i + \tilde{U}_i'\left(x_i\right) - U_i'\left(x_i\right), \tag{12}$$

which replaces $\tilde{U}_i'\left(x_i\right)$ in (11) with the cooperative $U_i'\left(x_i\right)$. However, this design is not implementable because $\tilde{U}_i\left(x_i\right)$ is not known to the supervisor. Instead, in our design, we obtain an estimate of $\tilde{U}_i\left(x\right)$ with the help of the cooperative reference model:

$$\dot{\hat{x}}_i = \kappa_i\left(U_i'\left(x_i\right) - q_i\right), \quad \hat{x}_i\left(0\right) = x_i\left(0\right). \tag{13}$$

The $\hat{x}_i$ thus calculated differs from x_i by $e_i := \hat{x}_i - x_i$, which, from (11)-(13), is governed by

$$\dot{e}_i = \kappa_i\left(\tilde{q}_i - q_i - \tilde{U}_i'\left(x_i\right) + U_i'\left(x_i\right)\right). \tag{14}$$

This means that, if we design the price adjustment to be

$$\tilde{q}_i = q_i - \rho_i e_i, \tag{15}$$

with a sufficiently high gain $\rho_i > 0$, then the variable e_i evolves in a faster time scale than x_i, and reaches the quasi-steady state $\rho_i e_i \approx -\tilde{U}_i'(x_i) + U_i'(x_i)$. Thus, after a fast transient, our design (13), (15) approximates the non-implementable scheme (12). For cooperative users, where $\tilde{U}_i(x_i) = U_i(x_i)$, (13) and (15) yield $\tilde{q}_i = q_i$, which means that no price adjustment is applied. Note that U_i in (13) is not necessarily the same for each user. This means that the supervisor can intentionally set up different utility functions, and use this flexibility to only adjust high bandwidth flows while leaving low bandwidth flows without rectification.

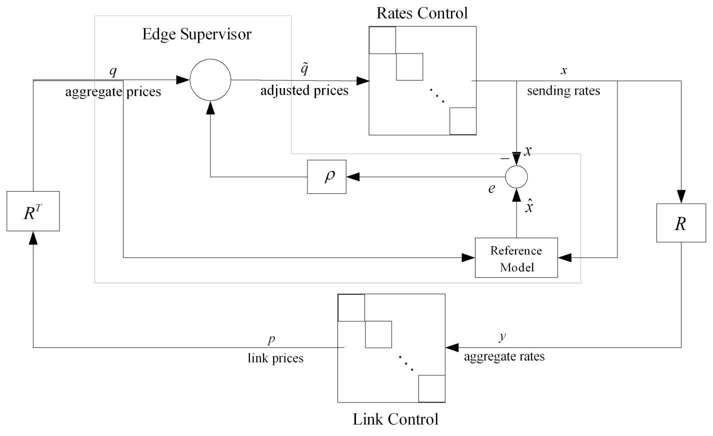


Figure 2: Price adjustment for uncooperative users in Kelly's primal algorithm.

The algorithm (13), (15) is depicted with a block diagram in Figure 2. In implementation, the edge router performs the price adjustment (15) every round trip time. The phrase "fast time scale" is used to indicate that when ρ is large in (15), the supervisor subsystem approaches its quasi steady-state faster than the dynamics of the network. In Theorem 1 below, we use tools from singular-perturbations theory [18], [19] to prove that (13), (15) achieves asymptotic stability of the cooperative value x^* in (9)-(10):

Theorem 1 *Consider the network (1)-(3), where some users implement the uncooperative algorithm (11), rather than (5). Suppose* $U_i(x_i) : \mathrm{R}_+ \to \mathrm{R}$ *are increasing and sufficiently smooth functions,* $U_i''(x_i) < 0 \;\; \forall x_i \in \mathrm{R}_+$*, and* $U_i(x_i) \to -\infty$ *and* $\tilde{U}_i(x_i) \to -\infty$ *as* $x_i \to 0$ *for* $i = 1, \cdots, N$*. Then, the price adjustment algorithm (13), (15) ensures that, for any compact set* $\Omega \subset \mathrm{R}_+^N$ *of initial conditions* $x(0)$*, there exists* $\rho_i^* > 0$ *such that, if* $\rho_i > \rho_i^*$*, then* $x(t)$ *and* $\hat{x}(t)$ *remain bounded, and* $x(t)$ *converges to the cooperative value* x^* *in (9)-(10).*

The assumptions of Theorem 1 on the utility functions $U_i(x_i)$ are standard in the literature [1], [14], [25]. In particular, the assumption $U_i(x_i) \to -\infty$ as $x_i \to 0$ ensures that R_+^N is positively-invariant, i.e., if x is initially in R_+^N, it will remain in R_+^N for all $t \geq 0$. It is satisfied by commonly used utility functions such as $U_i(x_i) = -\frac{a_i}{x_i}$ (variant of TCP Reno) and $U_i(x_i) = a_i \log x_i$ (TCP Vegas) [14]. For others, such as $U_i(x_i) = \frac{\sqrt{2}}{\tau_i} \tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right)$ (TCP Reno), we can modify Theorem 1 and prove stability by using positive projection functions as in [15]. It is reasonable to make the same assumptions for $\tilde{U}_i'(\cdot)$ as for $U_i'(\cdot)$, because cheating users would typically change the parameters of the nominal utility functions, such as a_i in TCP Vegas above. However, this assumption excludes some traditional unresponsive flows referred to as UDP or CBR, in which, users send data at a constant rate without acknowledging any feedback from the network. □

Proof: To represent the algorithm (11), (13) and (15) in the standard singularly perturbed form [18], [19], we let

$$\omega_i := \rho_i e_i \tag{16}$$

$$\varepsilon_i = \frac{1}{\rho_i} \tag{17}$$

and obtain:

$$\dot{x}_i = \kappa_i \left(\tilde{U}_i'(x_i) - q_i + \omega_i\right). \tag{18}$$

$$\varepsilon_i \dot{\omega}_i = -\kappa_i \left(\omega_i + \tilde{U}_i'(x_i) - U_i'(x_i)\right). \tag{19}$$

An inspection of (18) and (19) shows that the equilibrium for x_i is same as the cooperative x_i^* in (9)-(10), and the equilibrium for ω_i is

$$\omega_i^* = -\tilde{U}_i'(x_i^*) + U_i'(x_i^*). \tag{20}$$

To shift this equilibrium to 0, we define

$$\varpi_i := \omega_i + \tilde{U}_i'(x_i) - U_i'(x_i) \tag{21}$$

and rewrite (18)- (19) as

$$\begin{aligned} \dot{x} &= K\left(U'(x) - R^T h(Rx) + \varpi\right) \\ \varepsilon\dot{\varpi} &= -K\left(\varpi - \varepsilon \frac{\partial\left(\tilde{U}'(x) - U'(x)\right)}{\partial x}\left(U'(x) - R^T h(Rx) + \varpi\right)\right) \end{aligned} \tag{22}$$

where we use the vector notation $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T$, $\varpi = \begin{bmatrix} \varpi_1 & \varpi_2 & \cdots & \varpi_N \end{bmatrix}^T$. $K = \text{diag}\{\kappa_i\}$ and $\varepsilon = \text{diag}\{\varepsilon_i\}$ are diagonal matrixes of the source controller gains $\kappa_i > 0$ and $\varepsilon_i > 0$, $i = 1, \cdots, N$, and $U'(x) \in \mathrm{R}^N$ is a vector whose ith component is the derivative $U_i'(x_i)$ of the utility function $U_i(x_i)$. Likewise, $h(y) \in \mathrm{R}^L$ and $\tilde{U}'(x) \in \mathrm{R}^N$ consist of the penalty functions $h_l(y_l)$ and uncooperative utility functions $\tilde{U}_i'(x_i)$.

To prove asymptotic stability of $(x, \varpi) = (x^*, 0)$ we use the Lyapunov function

$$\begin{aligned} V = &\sum_{i=1}^{N} \left(-\left(U_i(x_i) - U_i(x_i^*)\right) + q_i^*(x_i - x_i^*)\right) \\ &+ \sum_{l=1}^{L} \left(\int_{y_l^*}^{y_l} \left(h_l(\sigma) - h_l(y_l^*)\right) d\sigma \right) + \frac{1}{2}\varpi^T K^{-1} \varpi \end{aligned} \tag{23}$$

in which, the first and the second terms, are identical to the Lyapunov function used in [1, 15] for the proof of the stability of Kelly's Primal algorithm, while the third term is a quadratic Lyapunov function for the dynamics of ϖ subsystem. This Lyapunov function is positive definite and radially unbounded in R_+^N, and yields the derivative

$$\dot{V} \leq -f_1(x)^T K f_1(x) - \varpi^T \varepsilon^{-1} \varpi + \varpi^T \frac{\partial\left(\tilde{U}'(x) - U'(x)\right)}{\partial x} \varpi + \varpi^T f_2(x), \tag{24}$$

where

$$f_1(x) := U'(x) - R^T h(Rx), \tag{25}$$

$$f_2(x) := \frac{\partial\left(\tilde{U}'(x) - U'(x)\right)}{\partial x} \left(\left(U'(x) - R^T h(Rx)\right) + K\left(-U'(x) + R^T h(Rx)\right)\right). \tag{26}$$

We show in Lemma 1 below that, on any compact set of (x, ϖ) that includes $(x^*, 0)$, we can choose ε small enough to ensure $\dot{V}$ is negative definite. The conclusion of Theorem 1 follows from this lemma because, from $\hat{x}(0) = x(0)$, we have $\omega(0) = 0$ and, thus $\varpi(0) = -\tilde{U}'(x(0)) + U'(x(0))$, which means that for any set Ω as in the statement of the theorem, we can find a corresponding region of attraction in (x, ϖ) coordinates, which does not depend on ε. Since V is also independent of ε, we can select a level set of V that encompasses this region of attraction, and design ε from Lemma 1 to render $\dot{V}$ negative definite in this level set. □

Lemma 1: Let the assumptions of Theorem 1 hold, and let $f_1(x)$ and $f_2(x)$ be defined as in (25)-(26). Then, for any compact set Λ of (x, ϖ) that includes $(x^*, 0)$, there exists $\varepsilon^* > 0$ such that if $\varepsilon_i \in (0, \varepsilon^*]$ for all $i = 1, \cdots, N$, then $\dot{V}(x)$ given in (24) is negative definite on Λ.

Proof: We first claim that there exists a constant $\delta > 0$ such that, for any compact set Λ of (x, ϖ) that includes $(x^*, 0)$,

$$f_1(x)^T K f_1(x) \geq \delta \|x - x^*\|^2 . \tag{27}$$

To prove this we show that the Hessian of $f_1(x)^T K f_1(x)$ is positive definite at x^*. To this end, we note that

$$f_1^T(x) K f_1(x) = \sum_{i=1}^{N} \kappa_i f_1^i(x)^2$$

and use the chain rule for the second derivative of the function $\kappa_i \left(f_1^i(\cdot)\right)^2$:

$$\left.\frac{\partial^2 \left(\kappa_i \left(f_1^i(x)\right)^2\right)}{\partial x^2}\right|_{x=x^*} = 2\kappa_i f_1^i(x^*) \left.\frac{\partial^2 \left(f_1^i(x)\right)}{\partial x^2}\right|_{x=x^*} + 2\kappa_i \left.\left(\frac{\partial \left(f_1^i(x)\right)}{\partial x}\right)^T \left(\frac{\partial \left(f_1^i(x)\right)}{\partial x}\right)\right|_{x=x^*} .$$

Because $f_1^i(x^*) = 0$ and because

$$\frac{\partial f_1}{\partial x}(x^*) = U''(x^*) - R^T \left.\frac{\partial h}{\partial (Rx)}\right|_{x=x^*} R \leq U''(x^*) < 0$$

we conclude

$$\begin{aligned}\left.\frac{\partial^2 \left(f_1(x)^T K f_1(x)\right)}{\partial x^2}\right|_{x=x^*} &= 2\sum_{i=1}^{N} \kappa_i \left.\left(\frac{\partial \left(f_1^i(x)\right)}{\partial x}\right)^T \left(\frac{\partial \left(f_1^i(x)\right)}{\partial x}\right)\right|_{x=x^*} \\ &= 2\left(\left.\frac{\partial f_1}{\partial x}\right|_{x=x^*}\right)^T K \left(\left.\frac{\partial f_1}{\partial x}\right|_{x=x^*}\right) > 0\end{aligned}$$

which proves (27).

Next, we apply Young's Inequality [26] to the term $\varpi^T f_2(x)$ in the right hand side of (24):

$$\varpi^T f_2(x) \leq \frac{1}{\lambda} \|\varpi\|^2 + \frac{\lambda}{4} \|f_2(x)\|^2 , \quad \lambda > 0,$$

and get

$$\begin{aligned}\dot{V} \leq &- \varpi^T \left(\varepsilon^{-1} - \frac{\partial \left(\tilde{U}'(x) - U'(x)\right)}{\partial x} - \frac{1}{\lambda} I_{N\times N}\right) \varpi \\ &- \delta \left(\|x - x^*\|^2 - \frac{\lambda}{4\delta} \|f_2(x)\|^2\right) .\end{aligned}$$

Because $f_2(x)$ is zero at zero and continuously differentiable, we can select λ and ε^* such that, for all $x \in \Lambda$,

$$\frac{\lambda}{4}\left\|f_2(x)\right\|^2 \leq \frac{1}{2}\left\|x-x^*\right\|^2$$

$$\frac{1}{2\varepsilon^*}I_{N\times N}-\frac{\partial\left(\tilde{U}'(x)-U'(x)\right)}{\partial x}-\frac{1}{\lambda}I_{N\times N}\geq 0$$

and obtain

$$\dot{V}\leq -\frac{1}{2\varepsilon^*}\varpi^T\varpi-\frac{\delta}{2}\left\|x-x^*\right\|^2,$$

for any $\varepsilon_i\in(0,\varepsilon^*]$, which concludes the proof. □

In Theorem 1, we require that the edge supervisor set $\hat{x}(0)$ equal to $x(0)$. However, it is not difficult to show that the proof holds true for small errors between $\hat{x}(0)$ and $x(0)$. Time delays in the network are not considered in Theorem 1. We wish to emphasize, however, that the high-gain component of our feedback design is limited to the local price adjustment loop, and not the price feedback loop, which is subject to the network delays. Thus, our correction algorithm can be combined with congestion control algorithms that are robust to time delays, such those in [23, 24, 25].

In implementation, it may also be necessary to know how large the gain ρ_i must be selected. While, in principle, such a value can be obtained from the calculation of ρ_i^* in the proof, this value may be conservative, and depends on the class of utility functions $\tilde{U}_i(\cdot)$ employed by uncooperative users. A more practical value can be obtained by monitoring whether the uncooperative rates persists and by increasing the gain ρ_i accordingly. A further discussion on the choice of this gain is given in Section VI.C.

4 Price adjustment for Kelly's dual algorithm

We next study Kelly's dual algorithm where uncooperative users implement, instead of (6),

$$x_i=\tilde{U}_i'^{-1}(\tilde{q}_i). \tag{28}$$

We assume $\tilde{U}_i'^{-1}(s)\geq U_i'^{-1}(s)$, $\forall s\geq 0$, which means that the uncooperative sending rate is larger than the cooperative rate. To counteract such uncooperative users, the supervisor must replace the nominal price feedback q_i with

$$\tilde{q}_i=\tilde{U}_i'\circ U_i'^{-1}(q_i), \tag{29}$$

which, when substituted in (28), results in the cooperative rate (6). Because a direct solution of (29) would require the knowledge of $\tilde{U}_i'(\cdot)$, which is not available to the supervisor, we propose the dynamic algorithm

$$\tilde{q}_i = q_i + \omega_i, \tag{30}$$

$$\dot{\omega}_i = \rho_i \left(x_i - U_i'^{-1}(q_i) \right), \quad \omega_i(0) = 0, \quad \rho_i > 0, \tag{31}$$

depicted in Figure 3. The equilibrium of (31) is achieved when

$$x_i = U_i'^{-1}(q_i), \tag{32}$$

which indeed coincides with the cooperative rate (6). We achieve asymptotic stability of this equilibrium, again, by designing the adaptation gain ρ_i to be sufficiently high:

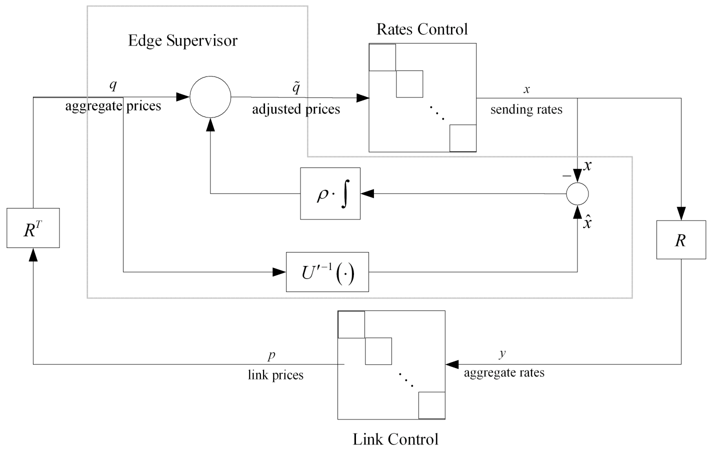


Figure 3: Price adjustment for uncooperative users in Kelly's dual algorithm.

Theorem 2 *Consider the network (1)-(3), (6) and (28), where $U_i(x_i)$ and $\tilde{U}_i(x_i)$ are as in Theorem 1, and $\tilde{U}_i'^{-1}(s) \geq U_i'^{-1}(s)$, $\forall s \in \mathrm{R}_+$. Then, the price adjustment algorithm (28), (31), ensures that, for any compact set $\Omega \subset \mathrm{R}_+^N$ of initial conditions $p(0)$, there exists $\rho_i^* > 0$ such that, if $\rho_i > \rho_i^*$, then $p(t)$, $x(t)$ and $\tilde{q}(t)$ remain bounded, and $x(t)$ and $p(t)$ converge to the cooperative values x^* and p^* in (7)-(8).*

Proof: To represent the algorithm (28), (31) in the standard singularly perturbed form, we let

$$\varepsilon_i = \frac{1}{\rho_i} \tag{33}$$

and obtain:

$$\varepsilon \dot{\omega}_i = \tilde{U}_i'^{-1}(q_i + \omega_i) - U_i'^{-1}(q_i). \tag{34}$$

This means that the equilibrium for ω_i satisfies

$$\tilde{U}_i'^{-1}\left(q_i^* + \omega_i^*\right) = U_i'^{-1}\left(q_i^*\right) \tag{35}$$

which implies, from (28), that

$$x_i^* = \tilde{U}_i'^{-1}\left(q_i^* + \omega_i^*\right) = U_i'^{-1}\left(q_i^*\right). \tag{36}$$

We thus conclude from (8) that the equilibria for x_i, q_i and p_j are the same as the cooperative x_i^*, q_i^* and p_j^*. To shift the equilibrium ω_i^* in (35) to 0, we define

$$\varpi_i = \omega_i - \Phi_i\left(q_i\right) \tag{37}$$

where

$$\Phi_i\left(q_i\right) = \tilde{U}_i' \circ U_i'^{-1}\left(q_i\right) - q_i, \tag{38}$$

and represent the system (11)-(15) and (34) as

$$\begin{aligned}\dot{p} &= \Gamma\left(RU'^{-1}\left(R^T p\right) + \Delta\left(p,\varpi\right) - c\right)_p^+ \\ \varepsilon\dot{\varpi} &= \tilde{U}'^{-1}\left(q + \varpi + \Phi\left(q\right)\right) - U'^{-1}\left(q\right) - \varepsilon\frac{\partial\Phi\left(q\right)}{\partial q}R^T\Gamma\left(RU'^{-1}\left(R^T p\right) + \Delta\left(p,\varpi\right) - c\right)_p^+ .\end{aligned} \tag{39}$$

where $p = \begin{bmatrix} p_1 & p_2 & \cdots & p_L \end{bmatrix}^T$, $\varpi = \begin{bmatrix} \varpi_1 & \varpi_2 & \cdots & \varpi_N \end{bmatrix}^T$. $\Gamma = \mathrm{diag}\left\{\gamma_j\right\}$ is a diagonal matrix of the source controller gains $\gamma_j > 0$, $j = 1, \cdots, L$, $U'^{-1}\left(x\right) \in \mathrm{R}^N$, $\tilde{U'}^{-1}\left(x\right) \in \mathrm{R}^N$ and $\Phi\left(q\right) \in \mathrm{R}^N$ are vector functions as in Theorem 1, and

$$\begin{aligned}\Delta\left(p,\varpi\right) :=& R\tilde{U}'^{-1}\left(R^T p + \varpi + \Phi\left(R^T p\right)\right) - RU'^{-1}\left(R^T p\right) \\ =& R\tilde{U}'^{-1}\left(R^T p + \varpi + \Phi\left(R^T p\right)\right) - R\tilde{U}'^{-1}\left(R^T p + \Phi\left(R^T p\right)\right).\end{aligned} \tag{40}$$

To prove asymptotic stability of $(p,\varpi) = (p^*, 0)$, we use the Lyapunov function

$$V = \sum_{i=1}^{L}\frac{1}{2}\gamma_l^{-1}\left(p_l - p_l^*\right)^2 + \frac{1}{2}\varpi^T\varpi \tag{41}$$

which has the first term same as in [15] for the stability of Kelly's Dual algorithm and the second term as a quadratic Lyapunov function for the ϖ subsystem. This Lyapunov function is positive definite and radially unbounded in R_+^{N+L}, and yields the derivative

$$
\begin{aligned}
\dot{V} \leq\,& (p-p^*)^T R\left(U'^{-1}\left(R^T p\right)+\Delta(p,\varpi)-c\right)_p^+ \\
& -\varpi^T\varepsilon^{-1}\left(U^{-1}(q)-\tilde{U}'^{-1}\left((q+\varpi+\Phi(q))^+\right)\right) \\
& +\varpi^T\frac{\partial\Phi}{\partial q}R^T\Gamma\left(RU'^{-1}\left(R^T p\right)+\Delta(p,\varpi)-c\right)_p^+ \\
\leq\,& (p-p^*)^T R\left(U'^{-1}\left(R^T p\right)-U'^{-1}\left(R^T p^*\right)\right) \\
& -\varpi^T\varepsilon^{-1}\left(U^{-1}(q)-\tilde{U}'^{-1}\left((q+\varpi+\Phi(q))^+\right)\right) \\
& +(p-p^*)^T R\Delta(p,\varpi)+\varpi^T\frac{\partial\Phi}{\partial q}R^T\Gamma\left(RU'^{-1}\left(R^T p\right)+\Delta(p,\varpi)-c\right)_p^+
\end{aligned}
\tag{42}
$$

where the second inequality follows from the arguments in [24, Proof of Theorem 2]. When $Rx^* = c$, we show in Lemma 2 below that, on any compact set of (p, ϖ) that includes $(p^*, 0)$, we can choose ε small enough to ensure $\dot{V}$ is negative definite. Then, the conclusion follows as in the proof of Theorem 1. If $Rx^* = c$ does not hold, we can still establish asymptotic stability following the arguments of [24, Proof of Theorem 5]. $\square$

Lemma 2: Let the assumptions of Theorem 2 hold, and suppose $Rx^* = c$, that is all links are bottlenecked. Then, for any compact set Λ of (p, ϖ) that includes $(p^*, 0)$, there exists $\varepsilon^* > 0$ such that if $\varepsilon_i \in (0, \varepsilon^*]$ for all $i = 1, \cdots, N$, then $\dot{V}(x)$ given in (42) is negative definite on Λ. $\square$

The proof of Lemma 2 is similar to that of Lemma 1 and, is omitted due to space limitations.

5 Implementation and Simulation Setup

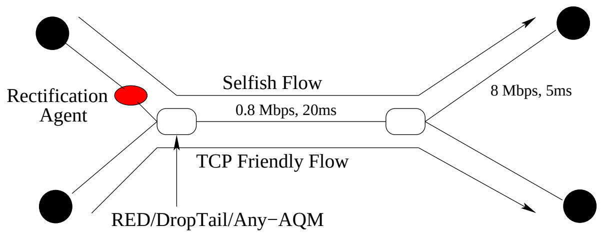
We have implemented the uncooperative framework presented in this paper in the Network Simulator (NS-2). While we have studied both dynamic (Section III) and static (Section IV) users, in simulations we implement the method of Section III because of the prevalence of TCP, which is dynamic and can be modeled as in (11) (see [1]). We added an edge-based supervisor, which adjusts the price feedback to the uncooperative users with the following algorithms

1. Let x_i be the rate of uncooperative user i as in (11).
2. Define an "audit" rate, $\hat{x}_i$, which represents the cooperative rate computed as in (13).
3. Calculate the marking or dropping probability at the edge as $p_i^e = \rho_i(x_i - \hat{x}_i)^+$, where $s^+ = max(s, 0)$, and ρ_i is the gain in (15).
4. For each incoming packet mark/drop a packet with a probability p_i^e.

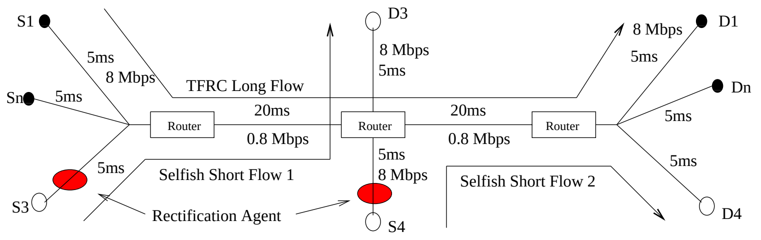
The implementation of this feedback adjustment depends upon the congestion notification policy deployed in the network. In our simulations we present the results with scenarios where marking (ECN) and dropping are used as congestion notification policies. The framework presented in this paper is independent of the buffer management policy deployed in the network; that is, it works with any Active Queue Management scheme as well as with simple Drop Tail queueing.

We note that, unlike the static link assumption in Section III, AQM and Drop-Tail in simulations make use of queue length and, hence, are dynamic algorithms. An extension of the proof of dynamic-source dynamic-link algorithms would be possible, but lengthy. The stability properties observed in simulations in the next section are indeed consistent with those predicted by Theorem 1.

We present simulation results for both single and multi-bottleneck topologies, depicted in Figure 4 a) and b). All the access links are configured to have a capacity equal to four times that of bottleneck links. The bottleneck links capacity and delay is fixed at 0.8Mbps and 20ms respectively unless specifically stated. (However, for some results presented in the paper, we will relax these configuration settings.) For all simulations reported in this paper, the simulation time is 150 seconds, and rate (or throughput) measurements are taken every 0.5 seconds. Each router has a buffer equal to one bandwidth delay product. In setups where the bottleneck routers have Random Early Drop (RED) buffer management policy deployed, the corresponding maximum and minimum threshold are set at $0.8 \times B$ and $0.3 \times B$ where B is the total buffer length; the queue weight was set to 0.002 and the maximum dropping probability to 0.1.



(a) Single-Bottleneck



(b) Multi-Bottleneck

Figure 4: Topologies used in simulations

The multi-bottleneck topology in Figure 4 b) shows one flow between source S1 and destination D1. This flow traverses both the bottleneck links and henceforth in this paper we will call this a *long flow*. The two flows between the source destination pairs [S2-D2] and [S3-D3] go over only one bottleneck link and, therefore, in the rest of paper we refer to them as *short flows*.

We refer to flows, which under same operating conditions, get more rate than TCP as selfish flows. This definition is also often commonly referred to as TCP-Friendliness. Since almost 90% of the traffic carried on the Internet uses TCP, we chose TCP-Friendliness as our definition of conformant flows. In this paper all transport protocols are rate based. Thus, all TCP-Friendly schemes use equation based rate control scheme (TCP Friendly Rate Control - TFRC) presented in [27] and all selfish schemes are variants of TFRC which have conservative decrease algorithms, i.e. upon congestion they decrease more slowly than TCP. We would like to refer the reader to

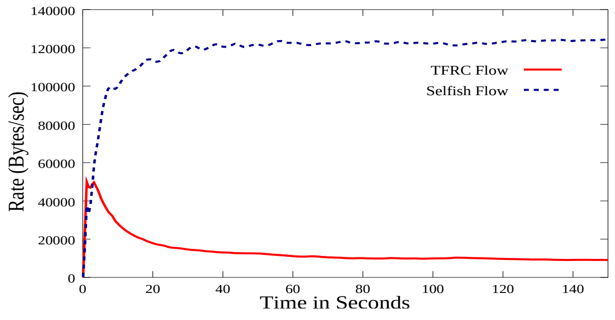
[12] for ways to generate selfish flows. In this paper we have also assumed that all the flows are persistent flows, i.e. they have infinite data to transfer. However, we will also present the results for the scenarios where we have both persistent and short web traffic competing for bandwidth.

6 Simulation Results

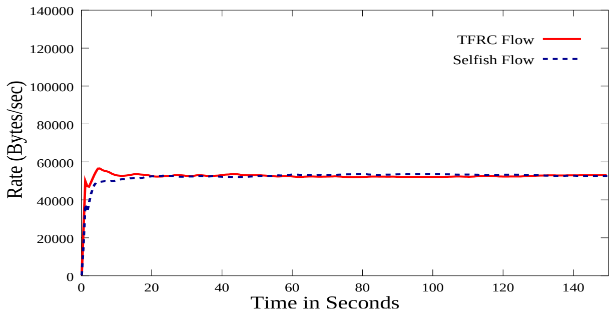
In this section we evaluate our algorithm for both single and multi-bottleneck topologies, with various degree of flow multiplexing. We also test its robustness in the presence of mice like web traffic and reverse path congestion.

6.1 AQM Based Network

In this section we assume that Active Queue Management (AQM) policies are deployed on the bottleneck routers, and all routers have RED queues installed on them. Figure 5 shows the results where two flows compete for bandwidth in a single bottleneck scenario. Among these flows, the first is TCP-Friendly with $U(x) = -1/x$, while the other is uncooperative with the utility function $U(x) = -1/\sqrt{x}$. If both competing flows were TCP-Friendly, they would have shared the bandwidth equitably. However, Figure 5 a) shows that the uncooperative flow grabs a larger share of the bandwidth. With our edge-based algorithm, in Figure 5 b) the two flows share the bottleneck bandwidth equitably. Simulations with other uncooperative utility functions in the single-bottleneck scenario, not presented here, yield similar results.



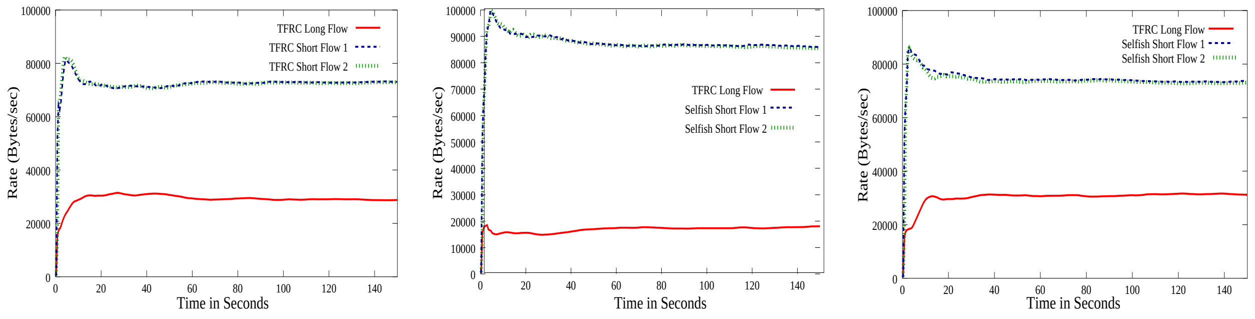
(a) Without Rectification



(b) With Rectification

Figure 5: Single-Bottleneck Topology: Equitable sharing of bandwidth enforced by the edge supervisor.

The multi-bottleneck topology is shown in Figure 4 b) and has two bottleneck links. The TCP-Friendly long-flow, which goes over both bottleneck links, competes for bandwidth against the two uncooperative short-flows, which go over only one bottleneck link. Figure 6 a) shows the result corresponding to the cooperative setup where both the long and short flows use TCP-Friendly rate control scheme. Figure 6 b) shows the result for the setup where we replace the TCP-Friendly short flows with uncooperative rate control schemes with utility function $U(x) = -1/\sqrt{x}$. Once again, we see that, the uncooperative flows get an unfair share of the bandwidth and almost force a traffic volume based denial of service attack. When we employ our edge-based supervisor, with $\rho = 2.5 \times 10^{-5}$, we recover the ideal bandwidth sharing of bottleneck links, as shown in Figure 6 a).



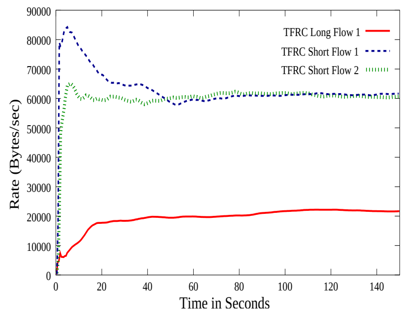
(a) Ideal Sharing (b) Without Rectification (c) With Rectification

Figure 6: Multi-Bottleneck scenario where (a) shows the ideal bandwidth sharing (b) shows the aggravated unfair sharing in the presence of uncooperative flows and (c) shows the rectification of uncooperative flows with our edge supervisor.

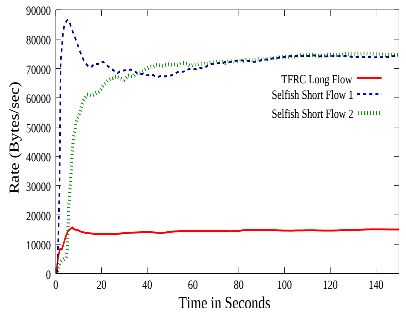
6.1.1 Background Traffic

In this section we will evaluate our framework in the presence of short web flows on a multi-bottleneck topology. In this setup, there is one long TCP-Friendly flow, which competes for bandwidth against one short uncooperative flow, on each bottleneck. HTTP sources are now added to these persistent flows. Each http page sends a single packet request to the destination, which then replies with a file of size which is exponentially distributed with 12 1Kb packets. After a source completes this transfer it waits for a random time, which is exponentially distributed with a mean of 1 second and then repeats the process.

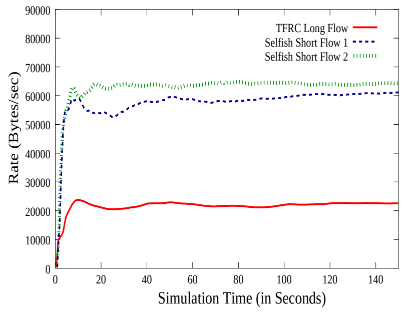
Our results in Figure 7 show that these web traffic or mice flows take some fraction of the bottleneck bandwidth while the persistent TCP and uncooperative flows compete for the remaining bandwidth. Figure 7 a) shows the results where, in the presence of mice traffic, all the persistent flows used a TCP-Friendly rate control scheme. This corresponds to the ideal bandwidth-sharing scenario. The results shown in Figure 7 b) corresponds to the setup where the persistent short flows, i.e. flows which go over only one bottleneck, use a selfish rate control scheme. We see that the selfish flows take an unfair share of the bottleneck bandwidth. Thereupon, we introduce our edge-based supervisor, with $\rho = 10^{-4}$, to rectify the misbehaving persistent short selfish flows, and recover the equitable rates as in Figure 7 c).



(a) Ideal Sharing



(b) Without Rectification

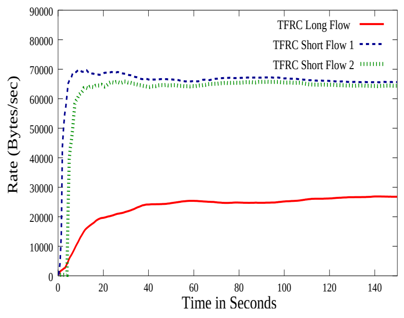


(c) With Rectification

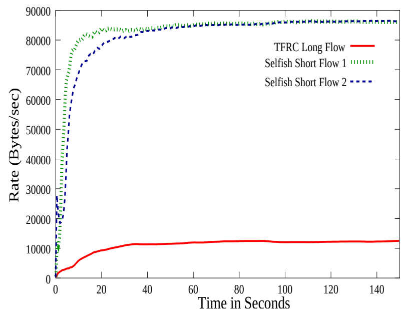
Figure 7: Background Web Traffic: (a) ideal bandwidth sharing scenario (b) aggravated unfair sharing in the presence of uncooperative flows and (c) rectification of uncooperative flows with our edge-based supervisor.

6.1.2 Cross Traffic

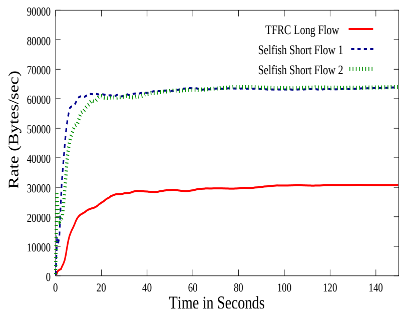
We next evaluate our framework in the presence of reverse path congestion. For this case we used a multi-bottleneck topology with one persistent TCP-Friendly flow traversing two bottleneck links competing against selfish flows going over just one bottleneck. For creating reverse path congestion, we added short TCP-Friendly flows on the reverse paths. Figure 8 a) shows the results corresponding to the ideal case, i.e. all the flows used TCP-Friendly rate control schemes while Figure 8 b) shows the results when the short flows on the forward path used an uncooperative rate control scheme. We then added our edge based rectification agent and the corresponding results are plotted in Figure 8 c) (with $\rho = 4 \times 10^{-5}$). We see that in the presence of edge-based supervisor, the selfish flows are effectively managed.



(a) Ideal Sharing



(b) Without Rectification



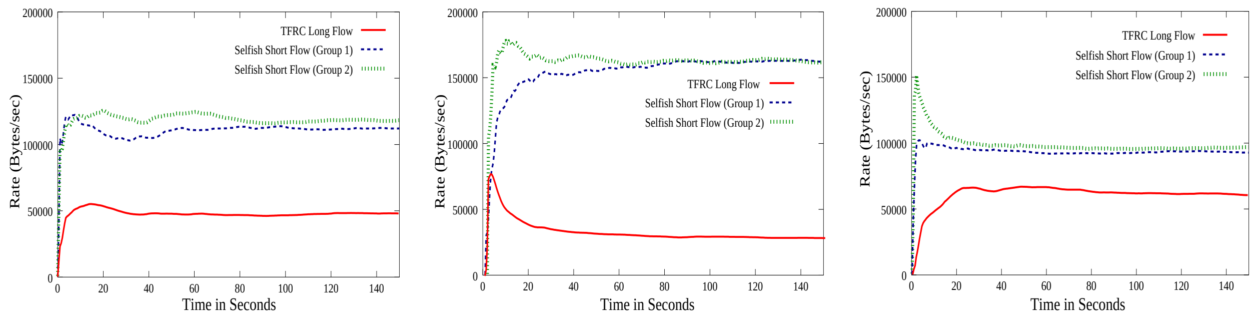
(c) With Rectification

Figure 8: Reverse Path Congestion : (a) ideal bandwidth sharing scenario (b) aggravated unfair sharing in the presence of uncooperative flows and (c) rectification of uncooperative flows with our edge-based supervisor.

6.1.3 Higher Flow Multiplexing with Background Traffic and Reverse Path Congestion

In the previous sections we have detailed the performance of our edge-based rectification algorithm for various single and multi-bottleneck scenarios. However, to present the efficiency and the robustness of our scheme we limited the number of competing flows to 3. In this section we will increase the number of competing flows.

For the results presented in this section, the TFRC flows competed for bandwidth against selfish flows on a multi-bottleneck topology (shown in Figure 4 b)). However, we increase the capacity of the bottleneck links to 8Mbps and that of access links to 80Mbps. The links delays were not changed and the bottleneck buffer was set to one bandwidth delay product. Finally, all the routers have RED queue management schemes deployed on them. The RED queue configurations are detailed in Section 5. For the edge-based rectification agent we set the ρ as 2.5×10^{-5}.



(a) Ideal Sharing (b) Without Rectification (c) With Rectification

Figure 9: Higher flow multiplexing with background traffic and reverse path congestion in a multi-bottleneck setup.

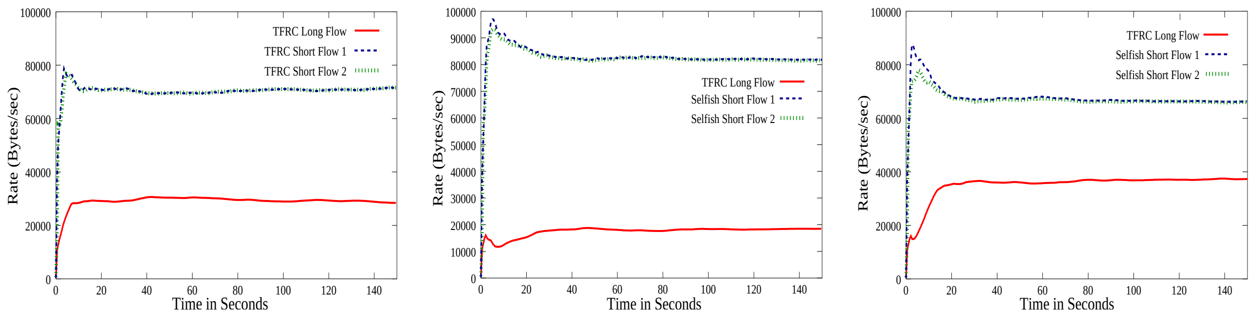
Figure 9 shows the results for the scenario where 5 TFRC flows compete for bandwidth against selfish flows. Further, on each bottleneck there were 5 selfish short flows. To these persistent flows, we also added short web transfers which occupied 10% of the bottleneck bandwidth. (The details about how we generated these flows is provided in Section 6.1.1. On average, at any given time, there were 10 short flows on each bottleneck.) For this simulation we also setup flows on the reverse path. Specifically, on each bottleneck in the reverse path there were 5 flows competing for bandwidth and thus creating congestion on the reverse path. For the results presented in this section we plot the throughput of one flow from each group: TFRC Long flows or flows which go over two bottleneck links, selfish short flows from Group 1 or flows which go over the first bottleneck only; and, finally, the selfish short flows from Group 2 or flows which go over the last bottleneck only.

Figure 9 a) shows the ideal sharing of the bottleneck when the short flows are also TCP-Friendly.

Figure 9 b) shows that in the absence of any policing the uncooperative flows get more share of the bandwidth at the expense of TFRC flows. With our rectification algorithm the fair share of the TFRC flows is restored; see Figure 9 c).

6.1.4 ECN Enabled Network

We conclude this section with the results from a multi-bottleneck test case where the RED scheme is configured to mark the packets (instead of dropping them). Once again, we used a setup where one persistent TCP-Friendly flow competed for bandwidth against one short uncooperative flow. Figure 10 a) shows the ideal bandwidth sharing while Figure 10 b) shows that in the presence of uncooperative flows the resulting bandwidth sharing is unfair. When we introduced the edge-based supervisor, with $\rho = 10^{-4}$, the bandwidth is shared fairly.



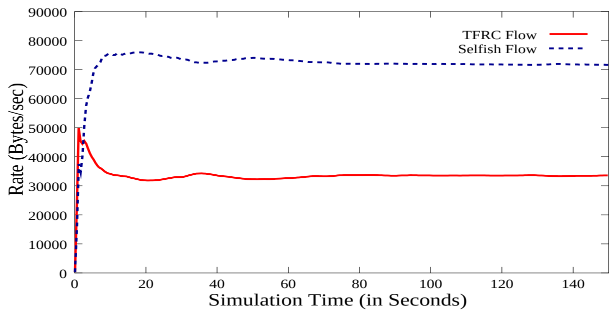
(a) Ideal Sharing (b) Without Rectification (c) With Rectification

Figure 10: ECN Enabled Network: (a) ideal bandwidth sharing scenario (b) aggravated unfair sharing in the presence of uncooperative flows and (c) rectification of uncooperative flows with our edge supervisor.

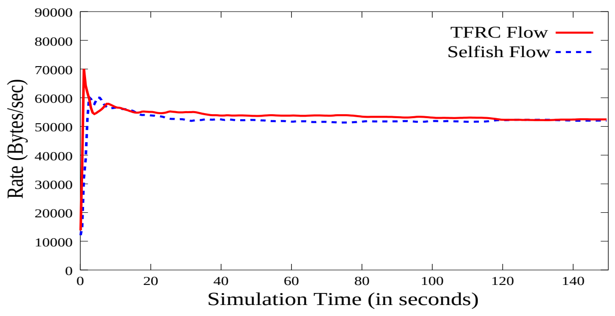
6.2 Drop-tail Queues

In this section, we look at the alternative scenario, where we assume that the network operates with Drop-Tail queues only. We extensively tested with various single and multi-bottleneck topologies and with different kind of uncooperative schemes. Figure 11 a) shows the results with a single bottleneck topology where one uncooperative and TCP-Friendly flows compete for bandwidth. As the figure shows, the uncooperative flows gain an unfair share of the bottleneck bandwidth, which are then corrected with our design. For this simulation the ρ was set to 3×10^{-5}.

Figure 12 shows the results with a multi-bottleneck setup where one long TCP-Friendly flow is sharing bandwidth with short uncooperative flows. Figure 12 a) shows the ideal bandwidth sharing and Figure 12 b) shows the unfair sharing of the bottleneck in the presence of uncooperative flows. Finally, in Figure 12 c) we present results with our rectification agent when $\rho = 1.5 \times 10^{-5}$.

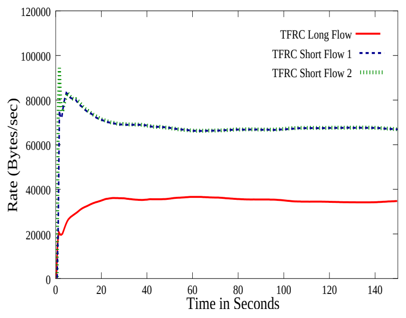


(a) Without Rectification

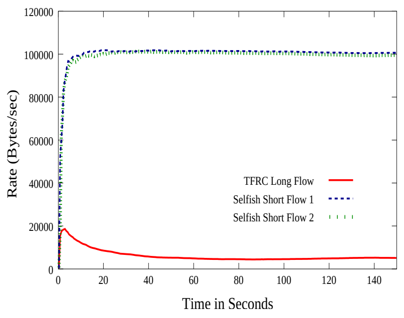


(b) With Rectification

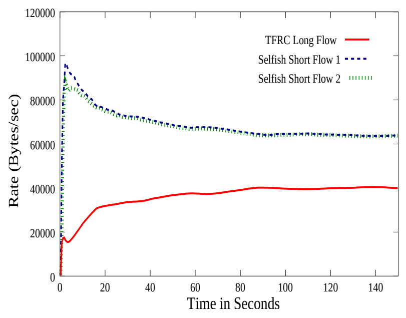
Figure 11: Drop tail queues and single bottleneck topology: rectification of uncooperative flows.



(a) Ideal Sharing



(b) Without Rectification



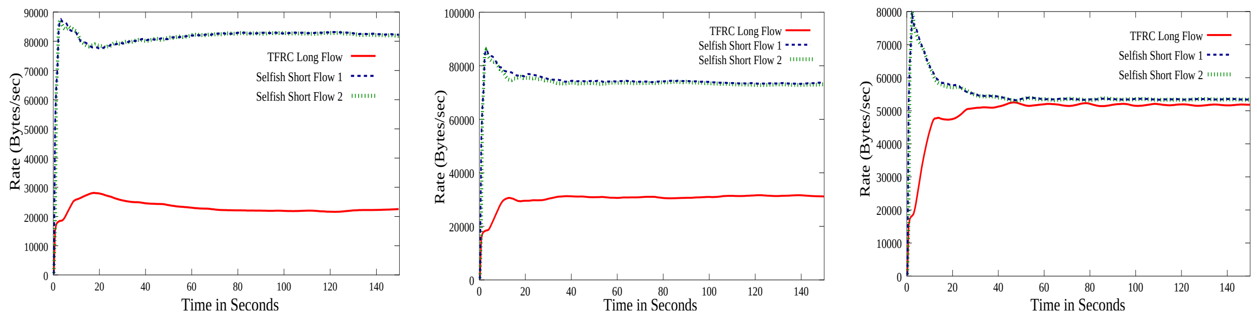
(c) With Rectification

Figure 12: Drop Tail Queues: (a) ideal bandwidth sharing scenario (b) aggravated unfair sharing in the presence of uncooperative flows and (c) rectification of uncooperative flows with our edge supervisor.

6.3 Effect of Gain ρ on Rectification of Selfish Users

The performance of our edge-based rectification algorithm depends on the gain ρ in equation (15). As detailed below, simulation studies indicate that too small or too large values of this ρ may deteriorate the performance. Indeed, Theorem 1 disallows small values of ρ because, otherwise, the desired two-time-scale behavior is not achieved. Although Theorem 1 allows arbitrarily large values for ρ, in practice, such high-gain leads to saturation of dropping or marking schemes, which violate the ”small marking probability” approximation in Equation (3). We see in the following simulations that large ρ might result in “*over-penalization*”, which means that uncooperative flows receive even less than their fair share.

Consider the multi-bottleneck setup shown in Figure 4 b) with one long TFRC flow and one short uncooperative flow on each bottleneck. In Figure 6 we presented simulations with $\rho = 2.5 \times 10^{-5}$. In Figure 13 we compare this result with $\rho = 10^{-5}$ (Figure 13 a)) and with $\rho = 10^{-4}$ (Figure 13 c)).



a) $\rho = 10^{-5}$ b) $\rho = 2.5 \times 10^{-5}$ c) $\rho = 10^{-4}$

Figure 13: Effect of gain ρ on the steady rates of uncooperative and TFRC flows.

We note that a high value of ρ may result in over-penalization, and on the other hand, with a very small value of ρ the selfish users are not sufficiently penalized and they continue to get more share of the bottleneck link(s) at the expense of cooperative users. However, for intermediate values, such as $\rho = 2.5 \times 10^{-5}$ in Figure 13 b), we recover the ideal shares for the uncooperative and the cooperative users.

For all the results reported in this paper we have found that the ideal range of ρ lies between the interval 10^{-4} to 10^{-5}. We also extensively evaluated the edge-based rectification model for different value of selfishness, i.e. users chose different values of $U(x)$, and found observation on ρ consistent with those reported above. A judicious choice of this gain, however, deserves further investigation.

7 Conclusions

We have presented a price adjustment algorithm for both Kelly's primal and dual network flow control models, and tested it on the Network Simulator. This algorithm is to be implemented at the edge of the network and, thus, does not require costly hardware upgrades in the entire network. It is independent of congestion notification policy deployed by the network, and thus, can be used with any Active Queue Management scheme, as well as Drop Tail queueing, which make the algorithm not only of theoretical interest but also practical importance. Although a suitable range for the gain ρ in our algorithm was determined by simulations, a judicious choice of this gain deserves further investigation. An on-line adaptation for ρ may be possible, and is currently being pursued by the authors.

References

[1] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability, *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[2] S. Low and D. Lapsley, Optimization flow control - I: basic algorithm and convergence, *IEEE/ACM Transaction on Networking*, vol. 7, no. 6, pp. 861-874, 1999.

[3] A. Akella, S. Seshan, R. Karp, S. Shenker and C. Papadimitriou. Selfish Behavior and stability of the Internet: A Game Theoretic Analysis of TCP. *Proceedings of ACM Sigcomm*, Aug 2002.

[4] S. Floyd and K. Fall. Promoting the Use of End-to-end Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458-472, 1999.

[5] A. Kuzmanovic and E. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). *Proceedings of ACM SIGCOMM*, Aug 2003.

[6] S. Gorinsky, S. Jain, H. Vin and Y. Zhang. Robustness to Inflated Subscription in Multicast Congestion Control. *Proceedings of ACM SIGCOMM*, Aug 2003.

[7] D. Lin and R. Morris. Dynamics of Random Early Detection, *Proceedings of ACM SIGCOMM*, Augutst, 1997.

[8] W. Feng et. al. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. *Proceedings of INFOCOM*, April 2001.

[9] R. Mahajan and S. Floyd. Controlling High-Bandwidth Flows at the Congested Routers. In ICNP 2001.

[10] Packeteer Inc. http://www.packeteer.com .

[11] I. Stoica, S. Shenker and Hui Zhang.Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks. *SIGCOMM'98*.

[12] K. Chandrayana and S. Kalyanaraman. Uncooperative Congestion Control. *Proceddings of ACM SIGMETRICS 2004*.

[13] S. Kunniyur and R. Srikant. End-to-end congestion control: Utility functions, random losses and ecn marks, *Proceedings of INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000.

[14] S.H. Low, F. Paganini and J.C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, 22(1):28-43, 2002

[15] J. Wen and M. Arcak, A unifying passivity framework for network flow control, *IEEE Transactions on Automatic Control,* vol. 49, no. 2, pp. 162–174, 2004.

[16] L. Brakmo, S. O'Malley, and L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. *Proceedings of the SIGCOMM 1994.*

[17] C. Jin, D. X. Wei and S. Low. FAST TCP: motivation, architecture, algorithms, performance. *Proceedings of IEEE INFOCOM 2004.*

[18] P. Kokotović, H.K. Khalil and J. O'Reilly, Singular Perturbation Methods in Control: Analysis and Design, Academic Press, 1986

[19] H.K. Khalil. Nonlinear Systems. Prentice Hall, Englewood Cliffs, NJ, third edition, 2002.

[20] F. Paganini, J. Doyle, and S. Low, Scalable laws for stable network congestion control, *Proceedings of 2001 Conference on Decision and Control*, Orlando, FL, Dec. 2001, pp. 185-190.

[21] F. Paganini, A global stability result in network flow control, *Systems and Control Letters,* vol. 46, pp. 165-172, 2002.

[22] S. Deb and R. Srikant, Global stability of congestion controllers for the Internet, University of Illinois, Urbana, IL, Internal Report, Feb. 2002.

[23] On the Stability of End-to-end Congestion Control for the Internet. Univ. of Cambridge Tech Report CUED/F-INFENG/TR.398, December 2000.

[24] X. Fan, M. Arcak, J. T. Wen, Robustness of Network Flow Control Against Disturbances and Time-Delay, *Systems & Control Letters*, v 53, n 1, p 13-29

[25] R. Srikant. The Mathematics of Internet Congestion Control. Birkhauser, 2004.

[26] G. H. Hardy, J. E. Littlewood and G. Polya. Inequalities. Cambridge University Press, second edition, 1988.

[27] S. Floyd, M. Handley, J. Padhye and J. Widmer. Equation-Based Congestion Control for Unicast Applications. *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.