Fair Congestion Control for Long-lived Background Flows

Costas Courcoubetis
Department of Engineering Systems and Design
Singapore University of Technology and Design
Singapore
costas@sutd.edu.sg

Antonis Dimakis, Michail Kanakakis
Department of Informatics
Athens University of Economics and Business
Athens, Greece
{dimakis, kanakakis}@aueb.gr

***Abstract*— We consider a new economic framework for designing congestion control algorithms for long-lived background flows. The algorithms aim at the fair sharing of the network capacity left over by the much shorter (and usually more interactive and delay sensitive) flows, between all competing background flows even if some use TCP in particular. The algorithm acts as a weighted variant of TCP with adaptive weight adjustment based on an estimate of the average throughput that would be obtained by a TCP flow following the same path, with the resulting throughput behavior ranging from low priority to that of TCP. It turns out that this design approximately maximizes the social welfare for the background traffic for appropriately defined utilities for bandwidth averaged over long time scales and a social cost function reflecting the negative externalities on the average download time of interactive flows, over all possible protocol designs. We illustrate our approach by proposing two concrete congestion control algorithms, each of them guaranteeing the throughput of TCP in the case of resource scarcity or resource abundance respectively. Two implementation alternatives are explored; one based on adjusting the congestion window ramp up speed and another based on alternating transmitting and idle periods. We use both analysis and simulations to validate our approach of implementing new background transfer protocols.**

I. INTRODUCTION

A key element of the success of the internet architecture is the ability to accommodate the needs of very diverse applications. Even though connection rates differ by few orders of magnitude and file transfer sizes vary by more than ten orders of magnitude the transport layer is pre-dominantly based on TCP and its variants. Recently however the widespread use of certain applications such as peer-to-peer file sharing and video streaming has not only challenged TCP but also the end-to-end principle. For example many internet service providers (ISPs) violate this principle by throttling file-sharing traffic using deep packet inspection at their equipment [1]. Commonly the throttling is applied during peak hours and file-sharing users are let free during off peak hours such as the nighttime. From an economic viewpoint heavy file-sharing usage during peak-time causes high social costs because of longer download times experienced by other users using the network at the same time. This cost is especially high to users performing interactive tasks such as web browsing, database transactions and video streaming. We should note here that (the majority of) file-sharing users are not malevolent; it is just that the transport generic protocols they use –most often TCP- is not designed to differentiate between peak and off-peak hours. Many file-sharing users –as they perform a mostly non-interactive task- would be equally happy if their downloads were shifted during off-peak hours. Thus while other more fair bandwidth sharing patterns are possible, the use of TCP by file-sharers makes unfair outcomes persist.

Recognizing this, specialized congestion control algorithms for background, i.e., non-interactive, data transfers have recently emerged, e.g., LEDBAT [2] [3], TCP-LP [4] and TCP Nice [5]. Such protocols –called also Lower than Best Effort (LBE) protocols- are typically designed to behave as low priority traffic, i.e., they are designed to cause low social costs due to download delays. However when they coexist with long-lasting flows which use non-LBE protocols, e.g., TCP, their throughput is severely decreased for a long time period. Such bandwidth starvation effects are a serious consideration in the design of LBE protocols [6]. This unfair bandwidth sharing pattern between non-LBE and LBE protocols, especially if both carry long flows- is manifested in weakened incentives for adoption of such new protocols instead of the incumbent TCP in public environments. Thus LBE protocols as opposed to TCP favor low costs due to delays but at a significant throughput degradation. Note also that in certain topologies with multiple bottlenecks LBE protocols might both obtain low throughput *and* impose higher delay costs. (See Section IV.B.)

Thus the incumbent TCP, and LBE protocols seem to represent two extreme operating points: TCP obtains higher throughputs at the expense of higher social costs while LBE protocols favor the latter but suffer from the point of view of throughput. We believe that the unfairness in the bandwidth sharing patterns is the cause of the engineering and economic issues present in each case. Furthermore, as we show in this paper, these problems can be attacked at the same time by exploring other more fair operating points within the two extremes. In this paper we build on the economic model and the two basic theoretical algorithms in [7], and design congestion control mechanisms implementing the algorithms. Our aim is to explore the issues involved in their design as well as to evaluate the quality of the resulting operating points from a fairness perspective.

We characterize fairness in terms of a social welfare maximization problem involving utilities for the background flow users attached to their long-term average throughputs, and

with a social cost accounting for the delay impact on short interactive flows. A basic feature of this approach –which makes it different from the ones based on Kelly's utility maximization- is that the utilities here model the demand for long-term throughput as a function of the marginal *delay*-based cost. Thus the demand does not depend on the instantaneous marginal resource price –as in Kelly-like models [9][10]-, which in general depends on other factors except delay, such as the link characteristics, the detailed behavior of the competing flows etc. This is important in the congestion controller design stage for setting the response based on fairness objectives which do not depend on such detailed characteristics that cannot be known at that time.

The paper is organized as follows. The basic notation and the social welfare maximization problem introduced in [7] which motivates the work here is presented in Section II. In Section III two congestion control algorithms are described as well as the main functional building blocks. In Section IV we present simulation results of the two algorithms in a single-link and two-link topologies and compare with TCP and LBE protocols. Finally we conclude in Section V.

II. MOTIVATION

In this section we motivate our algorithm by considering a model of a single bottleneck link. We emphasize that the algorithm itself does not depend on any assumption of a single bottleneck; it is in this case though that the algorithm possesses an optimality property.

Consider a link of capacity C shared by $k+l$ long-lived background flows, where k of these use TCP and are out of our control while the rest will use our congestion control algorithm. If the link also carries a stream of dynamically arriving and departing short-lived flows with average throughput $C\rho$, then there is $C(1-\rho)$ bandwidth up for grabs for the background flows. We seek to optimize the bandwidth sharing policies used by the l non-TCP flows, called FAIRBAT (fair background traffic) flows hereafter for using our proposed algorithm, such that the leftover capacity is allocated fairly between all background flows including TCP ones. Furthermore, the fair allocation should take into account the impact on the download delays of the interactive flows. This leads to the social welfare maximization problem

$$\max ku_0(y_0) + \sum_{i=1}^{l} u_i(y_i) - \int_0^{\sum_{i=1}^{l} y_i} H\big(\delta(z)\big)\,dz \qquad (1)$$

such that $ky_0 + \sum_{i=1}^{l} y_i = C(1-\rho)$ over $y_0, y_1, \dots, y_l \geq 0$, where:

- y_0, y_i is the average throughput achieved by the TCP and i-th FB flow respectively.
- u_0, u_i is the utility function of TCP and the i-th FB flow respectively. The utilities intend to model users' long-term average throughput response to marginal delay cost externalities[1].
- H is any nonnegative, non-increasing function, and
- $\delta(z)$ is a normalized measure of the delay impact on the interactive flows when the FB flows obtain z average throughput. (See below.)

The cost term reflects the externality imposed on the download delay of interactive users and caused by background downloads. It is important to note that the delay depends also[2] on the instantaneous bandwidth allocations which result from the particular congestion control mechanism used by each FB flow. Let $D_m(z)$ be the average download delay caused by a particular mechanism m when the FB flows obtain an aggregate throughput equal to z measured on a sufficiently long timescale such that the effect of short flows' dynamics is averaged out. Let also $D_0(z)$ be the average download delay if no background flows existed in the link, i.e., $k = l = 0$. Then the *delay impact* $\delta(z)$ is defined as

$$\delta(z) = \frac{D_m(z) - D_0(z)}{\bar{s}}$$

where $\bar{s}$ is the average flow size. Since $\delta(z)$ depends on the particular congestion control m used by each FB flow, the optimization in *(1)* is performed over all such possible mechanisms m in addition to the average throughputs $y_0, \dots y_l$. It is important to note that we do not change TCP; we are optimizing the way FB flows behave with TCP's behavior taken as given. Since all TCP flows behave in the same way, we are justified in assuming that these flows possess the same utility u_0.

Theorem 4 in [7] shows that if the search for a congestion control mechanism is restricted to weighted versions of TCP, i.e., flows which obtain a fixed proportion of TCP's throughput at any point in time –not just in the average sense- then the social welfare is nearly maximized for an appropriate choice of the proportion w_i (also called weight) by each FB flow i. The relative difference between the social welfare obtained in this way and the optimal one vanishes as k increases. More importantly, the gradient projection of the social welfare along the y_i direction, $-u_0'(y_0) + u_i'(y_i) - H\left(\delta\left(\sum y_j\right)\right)$, can be computed on the basis of local information by the i-th FB flow because for weighed TCP one has $\delta\left(\sum y_j\right) = 1/y_0$ (see [7]) and $y_0 = y_i/w_i$. Hence one is led to consider the gradient projection-based weight adaptation algorithm

$$\begin{aligned} w_i(t+1) - w_i(t) &= -g_w u_0'\left(\frac{y_i(t)}{w_i(t)}\right) + g_w u_i'\big(y_i(t)\big) \\ &\quad - g_w H\left(\frac{w_i(t)}{y_i(t)}\right), \end{aligned} \qquad (2)$$

where $y_i(t)$ is the average throughput resulting from the application of TCP weights $w_1(t), \dots, w(t)$ during the time period of constant length T between iterations t and $t+1$. The step gain g_w controls the speed of convergence and is such that

[1] To avoid confusion we emphasize the distinction between our model and the Kelly-based utility framework [9]; the latter models the instantaneous bandwidth share response on network congestion signals such as packet drops or packet delay.

[2] It also depends on the interactive flow arrival and size statistics.

the adaptation speed is sufficiently[3] slow, i.e., the adaptation timescale T/g_w is long.

Notice that for performing the update *(2)* each FB needs to know u_0. While this is not unreasonable since TCP's behavior is well-known, in this paper we plan using the social welfare maximization *(1)* as a device for generating new congestion control algorithms for particular choices of u_0, u_i and H by the algorithm designer. In what follows we consider the effect of two such choices which give rise to two different algorithms, FAIRBAT-I and FAIRBAT-II.

If $u_i(y) = \log y$ for every $i = 0, \dots, l$ then all background flows would have gotten an equal share of $C(1-\rho)$ if the cost term in *(1)* was neglected. The latter term shifts the balance towards allocations which hurt the interactive flows' delay less. If we further assume a delay cost with $H(\delta) = \gamma\delta^2$ for every $\delta > 0$, where $\gamma > 0$ is a constant, then at the equilibrium of *(2)*,

$$\frac{y_0}{y_i} = 1 + \frac{\gamma}{y_0} \quad (3)$$

holds for all $i = 1, \dots, l$.

The implied equilibrium allocation is rather simple and intuitive: at highly congested links where y_0 is small, the FB flow gets a proportionally small amount relative to y_0. At lightly loaded links where y_0 is large, the FB flow gets nearly as much as TCP since $y_i \approx y_0$ then. Of course such a behavior is sensible not only in single bottleneck links but also in multiple bottlenecks. However in the latter case $\delta(\sum y_j) \neq 1/y_0$ and the update *(2)* does not necessarily (approximately) maximizes social welfare. Still the resulting allocations could be desirable as shown in the example in Section IV.B. The viability and performance of such an algorithm, called FAIRBAT-I, or FB-I, is the object of study in the next sections.

Other equilibrium behaviors can be effected for different choices of utility functions and/or $H(\cdot)$, leading to different algorithms. A particular such choice arises when $u_0, \dots, u_l$ are as above and $H(\delta) = \gamma$ for every δ, i.e., a constant. Here, the equilibrium allocation is characterized by the relation

$$\frac{y_0}{y_i} = 1 + \gamma y_0. \quad (4)$$

The rationale behind such allocation is that at lightly congested links where bandwidth is abundant and so y_0 is large, a user of a background flow might be already satisfied with the throughput she gets and not want to compete with TCP for more bandwidth; on the other hand, at highly congested links the user might want to ensure that she achieves at least the throughput that TCP does. Such a behavior makes sense for rational users who would accept to use a transport protocol other than TCP only if they benefit from doing so, i.e., achieve a higher average throughput. An algorithm, called FAIRBAT-II, or FB-II, which implements such a protocol will be dealt with in the next sections.

III. FAIRBAT ALGORITHMS

As stated in the previous section, the algorithm aims at reaching an equilibrium where the average throughput y_i of a FB flow satisfies *(3)* or *(4)* with respect to the average throughput y_0 of a fictitious TCP flow following the same path. Moreover it should achieve this by using weighted TCP at every time; otherwise $\delta(\sum y_j) \neq 1/y_0$ and *(2)* fails to maximize *(1)* in single-link bottlenecks.

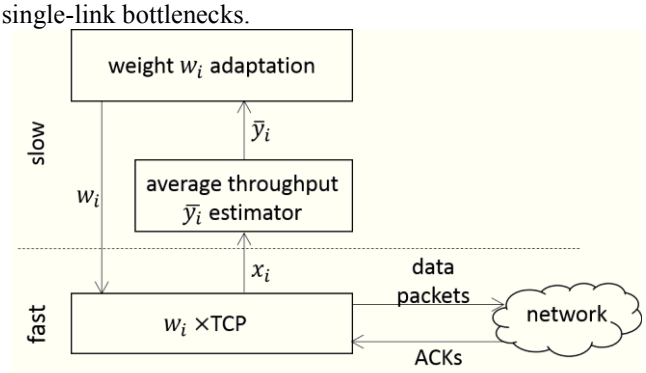

Fig. 1. *The functional blocks of the algorithm interact in a fast control loop (horizontally) between a weighted TCP congestion controller and the network. The TCP weight is adapted in a slower control loop (vertically).*

The algorithm can be thought of as being composed by three separate functional blocks depicted in *Fig. 1*:

1. A *weighted TCP congestion control algorithm* which given any weight α_i it tries to maintain a congestion window size equal to α_i times the congestion window of a fictitious TCP flow following the same path and facing the same loss events.
2. An *estimator of the average throughput* y_i resulting over a long time period during which the weight α_i is kept fixed. This is needed in order to estimate the effect of choosing weight α_i.
3. After the effect of α_i is estimated with sufficient accuracy, the *weight adaptation* step calculates the new weight value by *(2)* to be used during the $t+1$-th iteration.

Thus there are two control loops operating on different timescales. At a fast timescale typically of the order of a few round-trip times, the congestion window controlled by weighted TCP reaches a stochastic equilibrium. The throughput obtained by a FB (or any) flow is constantly modulated by the changing number of short-lived flows sharing part of the bottleneck links on the flow's path. Hence the estimator needs to filter out any throughput fluctuations at least as fast as the timescale of short-lived flow dynamics, and so the estimator's memory should be set on the order of tens of seconds or minutes. Choosing a longer memory will lead to a needlessly slow converging estimate and because of this the weight adaptation will need to be slowed down as well. Thus the weight adaptation control loop ideally should be as slow as the short-lived flow dynamics.

[3] Sufficiently slow for accurate estimates of the average throughput to be available.

In the next paragraphs we describe the implementation of each functional component in *Fig. 1*.

A. Weighted TCP through modifying TCP AIMD: α-TCP

The idea is to modify the TCP Additive Increase Multiplicative Decrease (AIMD) dynamics to obtain a throughput proportional to (unchanged) TCP. More specifically, the congestion window increment on each acknowledgement receipt is α_i/cwnd (where in general $\alpha_i \neq 1$[4]). In [8] it is shown that the throughput is proportional to $\sqrt{\alpha_i}$ and so $\alpha_i = w_i^2$ should be used if one wishes to enforce a weight w_i. Since both algorithms in Section II use $w_i < 1$ we will assume $\alpha_i < 1$. If during the t-th iteration the weight update algorithm uses the weight $w_i(t)$ then the increment $\alpha_i(t) = w_i(t)^2$ is used, and the congestion control algorithm (henceforth called α-TCP) responds to congestion signals according to:

On packet acknowledgement:

$$cwnd = cwnd + \frac{\alpha_i(t)}{cwnd}$$

On packet loss:

$$cwnd = \frac{cwnd}{2}$$

This implementation performs with satisfactory accuracy on Random Early Detection (RED) buffers, for a wide range of values for the system parameters. Under the drop-tail policy this is no longer the case and unpredictable outcomes may occur, where α-TCP flows obtain higher throughput than TCP even though their weight is well below 1. In *Fig. 2*, we show the congestion window evolution of the two protocols competing on a single link. On the left-side of the figure where the congestion window evolutions under RED is shown, both follow a typical saw-tooth pattern and the FB flow correctly obtains a portion of TCP's throughput. On the other hand, under drop-tail the FB flow faces fewer multiplicative decrease events than TCP, meaning that it faces a lower loss rate.

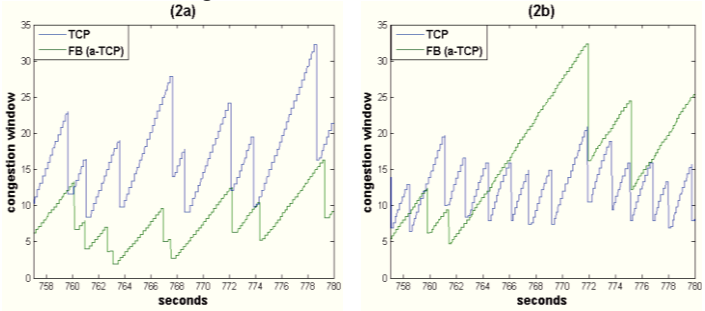

Fig. 2. *The background flows congestion window evolution in time: Fig. 2a depicts the RED queue case, where both FB and TCP flows face equal dropping probability. At the drop-tail queue (Fig. 2b) TCP performs more window decrease events (higher loss rate), resulting to incorrect capacity sharing between flows.*

This paradoxical behavior is well known (e.g., see [11]) and is due to phase-type effects arising by differences in the TCP gain. The effects insists appearing even when we inject (using the "overhead" parameter in ns-2 [12]) small random spaces of the order of tens of milliseconds between transmissions of consecutive packets and is not an artefact of the simulator. Typically the effects tends to disappear in large aggregation points where the synchronization breaks because of the randomness in packet transmissions arising due to the high number of flows. Since we cannot a priori rule out such phase-type effects, we next describe a different implementation which intends to emulate weighted TCP over long timescales and avoids these effects.

B. Weighted TCP through transmitting for only a portion of time: τ-TCP

Contrary to the approach in the previous paragraph where we control the instantaneous flow "aggressiveness", the main idea here is to keep the AIMD dynamics unchanged –so phase-type effects as in α-TCP will not occur- but let FB transmit only for a portion of time. The lifetime of each individual FB flow is divided into periods each consisting of two phases with a constant and a random duration denoted as d_c and d_r, respectively. d_c should be set to be sufficiently larger than a Round Trip Time (RTT) in order for the TCP flows sharing the same links to reach equilibrium throughputs within that time. The purpose of the random second phase is to break any potential synchronization between different FB flows. Let $D = d_c + d_r$ be the total duration of such a period, then each FB flow becomes active and transmits for time $\tau_i D$, while it remains silent without sending any data for $(1-\tau_i)D$, where τ_i is a supplied parameter. Thus, the average throughput obtained over many periods is approximately the τ_i portion of what a constantly transmitting TCP would obtain. Hence if the weight update algorithm wants to apply weight $w_i(t)$ during the t-th iteration then $\tau_i = \tau_i(t)$ should be set equal to $w_i(t) < 1$.

Even though τ-TCP emulates weighted TCP over long timescales it is not clear if the delay impact is close to that of the latter. The delay caused by τ-TCP flows could potentially be much larger than that caused by weighted TCP for the same long-term average throughput. This will entail higher delay costs in *(1)* and the social welfare could decrease. The next theorem shows that the relative delay decrease if weighted TCP is used in place of τ-TCP is no more than approximately 17.3% in the worst case. We note also that the upper bound 17.3% is tight. Recall the definition of $D_m(z)$ in Section II for a congestion control mechanism m used by the FB flows where $m \in \{\alpha\text{-TCP}, \tau\text{-TCP}\}$. Consider the case of a single link of capacity C where l FB flows coexist with k background TCP flows and a Poisson arrival stream of short TCP flows each associated with the download of an exponentially distributed random file size such that the imposed load is $\rho \in (0,1)$. In this system the FB flows could attain any amount of throughput z between 0 and $C(1-\rho)$. The following theorem holds whose proof is relegated to the Appendix:

[4] $\alpha_i = 1$ corresponds to regular TCP.

Theorem 1: For any $k, l \geq 1, C > 0, 0 \leq \rho < 1$ *the following is true:*

$$\sup_{0 \leq z < C(1-\rho)} \frac{D_{\tau\text{-}TCP}(z) - D_{\alpha\text{-}TCP}(z)}{D_{\alpha\text{-}TCP}(z)} \leq \sup_{\lambda > 0} \frac{\lambda - \frac{k f_k(\lambda)}{1 - f_k(\lambda)}}{k + \lambda + 1}$$

where $f_{k+1}(\lambda) = 1 - \frac{k+1}{\lambda} f_k(\lambda)$ *and* $f_1(\lambda) = (\lambda - 1 + e^{-\lambda})/\lambda$.

For $k = 1$ Theorem 1 yields the upper bound

$$\sup_{\lambda > 0} \frac{1 - (\lambda + 1)e^{-\lambda}}{(\lambda + 2)(1 - e^{-\lambda})} \approx 17.3\%$$

Combining this with the $3 - 2\sqrt{2} \approx 17.2\%$ (tight) upper bound of relative delay decrease between weighted TCP and the theoretically optimal algorithm (see [7]), yields (the not tight) 34.5% worst case bound for the difference between τ -TCP and the optimal. Using the variational formula for the upper bound as well as the numerically obtained bounds in [7] one can easily show that 30.5% is a (tight) lower bound for the worst case difference when $k = 1$. For k=2 the bound 12.2% given by Theorem 1 is even lower.

C. Average Throughput Estimator

The goal of the estimator is to track the average throughput of the controlled flow on a timescale where fluctuations due to short flow session dynamics are averaged out. On every iteration just before the weight update, the moving average estimate $\bar{y}_i(t)$ is updated according to

$$\bar{y}_i(t+1) = (1 - g_e)\bar{y}_i(t) + g_e x_i(t), \qquad (5)$$

where $x_i(t)$ is the data rate over the period after the last update only. The gain parameter g_e is appropriately set in order for the estimator memory (which is of the order of) T/g_e to be at least as large as the timescale of short flow dynamics. Choosing a much higher value will lead to excessively slow convergence of the estimates.

As explained in Section II, under α-TCP no estimator for y_0 is necessary since $y_0 = y_i/w_i$ and this fact allowed a weight update *(2)* requiring only local information. There is a subtle but intrinsic reason why this is not possible under τ-TCP: during inactive phases a competing TCP flow gets a higher bandwidth share than during active phases, so $y_0 w_i > y_i$, i.e., y_i/w_i is underestimating y_0. In large links however where many flows coexist, the phase has a negligible effect on the bandwidth share of a TCP flow, and so y_0 is not (significantly) underestimated.

Moreover in τ-TCP during inactive periods the rate is 0 so the convergence of the $\bar{y}_i$ estimate can be accelerated if inactive phases are taken a priori into account and not use the plain moving average update *(5)*. In particular we use *(5)* only during active phases in order to get an estimate $\bar{y}_0$ of y_0 (which underestimates the true y_0 as explained) and then let $\bar{y}_i = w_i\, \bar{y}_0$. The $\bar{y}_i$ estimate used in the weight update algorithms below is obtained in this manner, when τ-TCP is used.

D. Weight Adaptation

The weight $w_i(t)$ of FB flow i is updated according to *(2)* using the estimated values $\bar{y}_i(t)$ of $y_i(t)$. In particular, the first algorithm in Section II, FB-I, uses the weight update

FAIRBAT-I:

$$w_i(t+1) = w_i(t) - g_w \frac{w_i(t)}{\bar{y}_i(t)} + g_w \frac{1}{\bar{y}_i(t)} - g_w \gamma \left(\frac{w_i(t)}{\bar{y}_i(t)}\right)^2$$

which results, if $u_0(y) = u_i(y) = \log y$, $H(\delta) = \gamma\delta^2$, as assumed in motivating FB-I in Section II. If we use $H(\delta) = \gamma$ instead, we get the update step for our second algorithm:

FAIRBAT-II:

$$w_i(t+1) = w_i(t) - g_w \frac{w_i(t)}{\bar{y}_i(t)} + g_w \frac{1}{\bar{y}_i(t)} - g_w \gamma$$

IV. Simulation Results

In this section, we present the results of the simulation experiments performed in ns-2 [12]. Our aim is to assess the conformance of algorithms FB-I and II to the respective notions of fairness (for both α- and τ-TCP), *(3)* and *(4)*, and to measure their impact on the download delays of short flows. Further we compare with alternative congestion control protocols used by the background flows, such as TCP (Reno), LEDBAT [2] [3], and TCP-LP [4].

In what follows, the short flows follow a Poisson arrival process. Each such flow is associated with the transmission of a finite, exponentially distributed file size with mean 3Mbytes. In Subsection IV.A we consider a single bottleneck uplink with capacity $C = 14$Mbps, and in IV.B a sequence of two links of the same capacity. In the former case, RED is applied at the link buffer so that α- TCP and τ-TCP can both be compared. (Recall that under drop-tail, α-TCP may behave erratically because of phase effects.) In the two-link case, we consider the drop-tail policy so that a comparative analysis between τ-TCP and LEDBAT is possible since the latter was designed to work with the drop-tail policy [2].

A. Single link topology

Here the web flows arrive at rate $1/6$ flows/sec implying a link load $\rho_S = 4/14$ leaving up to $C(1-\rho) = 10$ Mbps available for the background flows. In Fig. 3*Fig. 3* we illustrate the average throughput obtained by FB flows by considering the (average of their) weight at equilibrium as more FB flows share the link. (A unit weight corresponds to the average throughput obtained by competing TCP background flows.) Each curve corresponds to a particular choice of algorithm (FB-I/II, α-TCP or τ-TCP) and to a particular number of coexisting TCP background flows. The curves marked with "theory" depict the y_i/y_0values obtained from *(3)* or *(4)*. Fig. 3a, which concerns the results of FB-I, shows that as the number of background flows and hence the average delay of the short flows increases, each FB flow obtains a smaller share compared to TCP, as required by *(3)*. The converse happens in Fig. 3b as described in *(4)*; the more congested the link becomes, the more similar to TCP the FB flows behave. τ-TCP seems to

consistently underestimate the theoretically optimum FB flow throughput and favor TCP. This may be due to the intrinsic underestimation of TCP throughput discussed in III.C, which agrees also with the fact that the amount of underestimation is less in links with more flows.

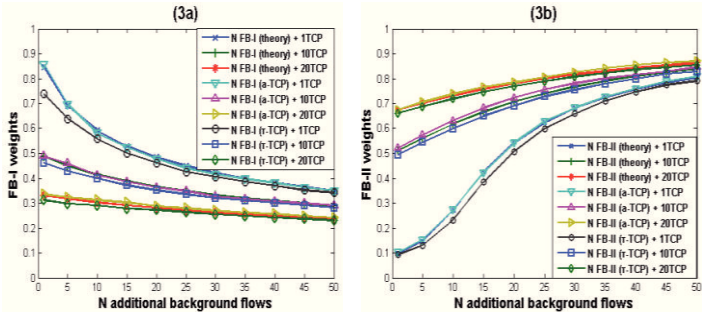

Fig. 3. *FB weight $w_i = y_i/y_0$ at equilibrium: As more background flows enter the network and the importance of delay increases, FB-I trades throughput for lower delay impact on the short flows (Fig. 3a). Contrariwise, in more competitive environments FB-II increases its weight to satisfy the objective for TCP-like throughput under high congestion (Fig. 3b).*

In Fig. 4 we depict the delay of short flows caused by the coexistence with background ones. We do not show separate curves for α-TCP, τ-TCP and "theory" because they are practically identical. For comparison, we plot also the delay caused if the background flows used all TCP. Under FB-I (see Fig. 4a) the delay impact is significantly reduced relative to TCP as the link becomes more congested. This is the case also for FB-II (see Fig. 4b) in light congestion where each additional background flow has a decreasing marginal effect. At higher congestion the marginal increase of delay follows that of TCP.

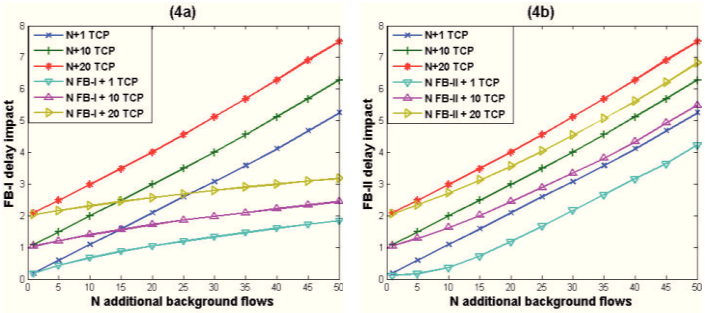

Fig. 4. *The impact on the download delay of web traffic: Under TCP background flows the marginal impact is constant as more background flows are added. The marginal impact of FB flows is varying depending on the level of congestion. FB-I and FB-II behave oppositely.*

Next, we consider how accurately each of the FB-I and FB-II algorithms approximates the maximum social welfare for the respective definition of cost. To do this we check how close the gradient of the objective function in *(1)* is to being zero. For FB-I this corresponds to checking how accurately the equations

$$\frac{1}{y_i} - \frac{1}{y_0} = \gamma\delta^2 \qquad (6)$$

for $i = 1, \dots, l$ are satisfied, where δ is the delay impact. The reason we do not use *(3)* instead is that it is equivalent to *(6)* only when $\delta = 1/y_0$. This fact holds under an ideal weighted TCP algorithm, so since α-TCP and τ-TCP are only approximations we check *(6)* directly. This does not pose any problem for FB-II since the associated objective function is independent of δ, and so we use *(4)*. Note also that y_0, y_i are obtained by directly estimating the average throughput of TCP and FB respectively.

In Fig. 5 we plot the left and right-hand sides of both *(6)* and *(4)* for $k = l = 10$, and $\gamma = \{0.5, 1, 1.5, 2, 2.5\}$. Fig. 5a shows the relevant plots for α-TCP while those for τ-TCP are shown in Fig. 5b. Observe that α-TCP closely satisfies *(6)*, while τ-TCP deviates significantly at higher values of γ. This deviation is not because δ fails to equal $1/y_0$ (since $\gamma/y_0^2 \approx \gamma\delta^2$ as shown in Fig. 5b) but because τ-TCP underestimates y_0 as explained in III.C. Interestingly, significant deviations are observed for FB-I only.

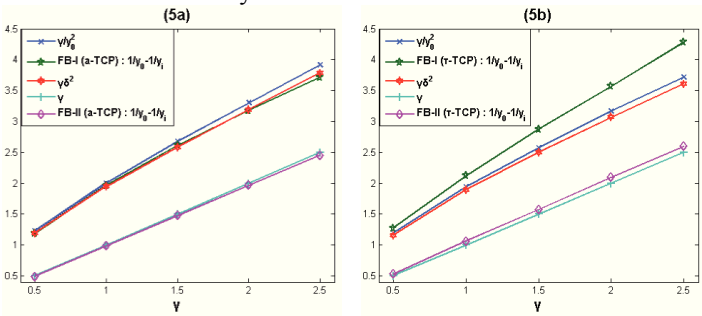

Fig. 5. *Check of equality of left and right-hand sides of the optimality conditions* (6), (4) *for FB-I and FB-II algorithms, respectively. The case of α-TCP is shown on the left figure while τ-TCP is on the right. There is a noticeable deviation from optimality in the latter case for FB-I.*

B. A two-link topology with indirect effects

As demonstrated in the last section, FB-I and FB-II allocate bandwidth between background flows such the social welfare *(1)* is approximately maximized. After all, their design was motivated from that problem. The problem though concerns a single link and FB-I/II cannot maximize social welfare in general multiple link networks as an example in a two-link topology depicted in Fig. 6 suggests.

In this section, we investigate the case where the background flows and the web traffic do not share a common link but the latter are only indirectly affected by the FB flows through the $k = 10$ background TCP that go through both links. In this topology the more bandwidth is consumed by the FB flows in the first link, the better the delay of the short flows at the second link will be. This is because the k TCP background flows will leave more space at the second link, as they are squeezed more in the first. Thus the externalities caused to the short flows by the FB ones are positive. Hence the cost term in *(1)* should reflect these positive externalities. However the formula $\delta = 1/y_0$ which is justified as an approximation of the delay impact in the single link case cannot hold in our case since it models negative externalities only: according to the formula a higher y_0 should improve the delay when in reality it has the opposite effect.

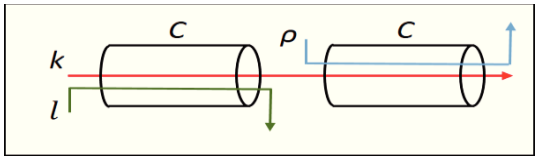

Fig. 6. *A two-link topology with the l FB flows indirectly affecting the stream of short flows (with load ρ) utilizing only the second link. The two links are coupled because of the k TCP background flows which use both links. Here the FB flows cause positive externalities to the short flows.*

So is e.g., FB-I a sensible algorithm for background flows in this scenario? Notice that an optimal weighted TCP algorithm would have to use weights always greater than 1, aiming to achieve the positive externalities. Any congestion control algorithm less aggressive than TCP will not be social welfare maximizing. In particular this is true for LBE algorithms such as LEDBAT, TCP-Nice, TCP-LP. In fact in topologies as in Fig. 6*Fig. 6*, existing LBE protocols will not only achieve lower throughput than FB-I/II but will also cause greater delays to the short flows since the background TCP flows are not going to be sufficiently squeezed.

To quantify the magnitude of these effects we consider scenarios where the l background flows, which traverse the first link only, all use either LEDBAT, TCP-LP, or TCP and compare with FB-I/II. We us the ns-2 implementation of LEDBAT available in [13], while for TCP-LP we use the one provided in ns-2. (In all cases the default parameters were used.) The number of background TCP flows k=10 is constant and the web flows arrive at a rate of 1/12, flows/sec, resulting to load $\rho = 2/14$. The number l of background flows varies in different experiments from 10 to 30.

In Fig. 7 we present the aggregate throughput obtained by the LBE protocols and FB-I/II normalized by the aggregate throughput obtained by the same l flows when they use TCP. As expected, the l background flows get the highest throughput when they use TCP. The FB-I/II flow throughput is much closer to TCP than to the LBE flows. Nevertheless Fig. 8 shows that the delays caused by FB-I/II (when normalized by the delays caused by TCP) are much lower than those of LBE and close to those under TCP.

It is also interesting to note the effect of the cost coefficient γ. While by *(3)* a lower value of γ yields a more aggressive protocol (see difference between $\gamma = 1$ and $\gamma = 0.5$ in Fig. 7),

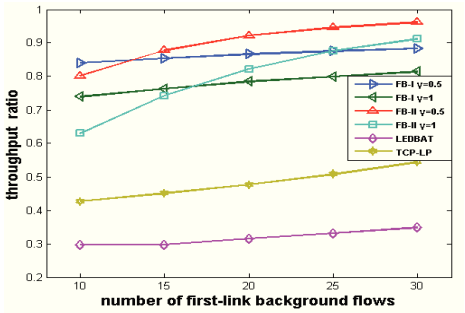

Fig. 7. *The throughput achieved by each of the l background flows in Fig.6 under different protocols. (The throughput is normalized by that when the l flows use TCP.)*

the presence of positive externalities in this topology cause the delay of short flows to decrease, as shown in Fig. 8b.

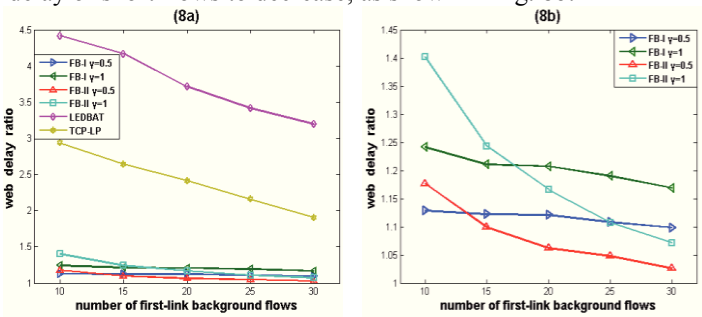

Fig. 8. *The delay caused to the short (web) flows when the l background flows in* Fig. 6 *use different protocols. (The delay is normalized by the one caused when TCP is used.) In 8b the delay caused by FB is shown more closely: more aggressive flows ($\gamma = 0.5$) cause lower delays.*

In other multiple link topologies positive and negative externalities can act at the same time. We expect that the throughput and delay impact performance of FB-I/II will always lie in between that of LBE protocols and TCP, without being clear which of the latter two is the best or worst. As shown in the single link and the two link example of this section, more satisfactory operating points seem to exist between the extremes offered by TCP and LBE protocols and are achievable the FB algorithms.

V. Concluding Remarks

The nearly identical performance of α- and τ-TCP shown in Fig. 3 and 4*Fig. 3* is remarkable and suggests that even though the delay they cause may differ up to 17% for the *same* FB throughput, a small drop in throughput (which implies a small drop in utility) leads to large delay savings. Of course the exact significance of such a change depends on the precise utilities and cost functions but we believe that trading off average throughput for delay is far more effective than employing intricate congestion control mechanisms. Thus the instantaneous bandwidth sharing patterns seem to be unimportant insofar as the long-term throughput remains unchanged. This observation further justifies the approach taken in this paper and [7], i.e., that of considering the problem of allocation of the leftover capacity as logically separate from the problem of instantaneous bandwidth sharing dealt in [9].

Appendix

In this section we obtain the proof of Theorem 1. Let $r_{k,l}$ be the worst case relative delay decrease, i.e., the left-hand side of the inequality in Theorem 1, under k background TCP and l FB flows. We first derive a formula for $r_{k,l}$.

Since the active phases of the τ-TCP flows are not synchronized, the equilibrium number N_l of active flows is binomially distributed with "success" probability τ_l. This probability satisfies $E\left(\frac{N_l}{N_l+k}\right) = \frac{z}{C(1-\rho)}$, where z is the aggregate average throughput obtained by all FB flows, and we call $f = \frac{z}{C(1-\rho)}$ the *share* of leftover capacity obtain by the FB flows. From the analysis in [7] we know that in a link with $k+i$ background TCP (and no FB) flows the resulting average

number of short flows is $(k+i+1)\rho/(1-\rho)$. Since the active phases of the τ-TCP flows are much longer than the timescale of the short flow dynamics, the last formula gives the average number of short flows during periods of time where i τ-TCP flows are active. Thus the overall average number of short flows is $(k+l\tau_l+1)\rho/(1-\rho)$. Using the analysis of weighted TCP from [7] we get the formula

$$r_{k,l} = \sup_{0\le f<1} \frac{l\tau_l - \frac{kf}{1-f}}{k+lt_l+1}.$$

Lemma 1: $r_{k,l}$ is increasing in l and so $r_{k,l} \le r_{k,\infty}$ where $r_{k,\infty} = \lim_l r_{k,l} < \infty$.

Proof: It suffices to show that $l\tau_l$ is increasing in l. Let N be a binomial random variable for $l+1$ trials with success probability p such that $p(l+1) = l\tau_l$, i.e., $EN_l = EN$. Then it is straightforward to check that the ratio of the (binomial) densities of N over that of N_l is unimodal in $i = 0, \dots, l+1$ and that N is neither stochastically greater nor less than N_l. These two conditions and the fact that $EN_l = EN$ imply that N_l is *less than N in the convex order* from Theorem 3.A.53 in [14]. This in turn implies $E\left(\frac{N_l}{N_l+k}\right) > E\left(\frac{N}{N+k}\right)$. Now under $l+1$ FB flows the N_{l+1} active flows obtain the same share f, i.e., $E\left(\frac{N_{l+1}}{N_{l+1}+k}\right) = f = E\left(\frac{N_{l+1}}{N_{l+1}+k}\right)$, and so $E\left(\frac{N_{l+1}}{N_{l+1}+k}\right) > E\left(\frac{N}{N+k}\right)$. Since N_{l+1}, N concern the same number of trials their means must be ordered according to $EN_{l+1} > EN$, i.e., $l\tau_l > (l+1)\tau_{l+1}$. If the limit $\lim_l l\tau_l = \lim_l EN_l$ was infinite then $\lim_l E\left(\frac{N_l}{N_l+k}\right) = 1 > f$ which contradicts our assumption, so $\lim_l l\tau_l < \infty$ and $\lim_l r_{k,l} < \infty$ hold. QED

Since $\lambda = \lim l\tau_l$ exists (see proof of last Lemma), the distributions of N_l converge to a Poisson with rate λ and so $r_{k,\infty}$ can be expressed alternatively as

$$r_{k,\infty} = \sup_{\lambda>0} \frac{\lambda - \frac{kf_k(\lambda)}{1-f_{k,\lambda}(\lambda)}}{k+\lambda+1},$$

where $f_k(\lambda) = E\left(\frac{N}{N+k}\right)$ for a rate λ Poisson random variable.

Lemma 2: $f_{k+1}(\lambda) = 1 - \frac{k+1}{\lambda} f_k(\lambda)$ for every $k \ge 1$, and $f_1(\lambda) = (\lambda - 1 + e^{-\lambda})/\lambda$.

Proof:

$$\begin{aligned} f_{k+1}(\lambda) &= \sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^i}{i!} \cdot \frac{i}{k+1+i} \\ &= \sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^i}{i!} - \sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^i}{i!} \cdot \frac{k+1}{k+1+i} \\ &= 1 - \frac{k+1}{\lambda} \sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^{i+1}}{(i+1)!} \cdot \frac{i+1}{k+i+1} \\ &= 1 - \frac{k+1}{\lambda} f_k(\lambda) \end{aligned}$$

To show the second part notice that

$$\begin{aligned} f_1'(\lambda) &= -\sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^i}{i!} \cdot \frac{i}{i+1} + \frac{1}{\lambda} \sum_{i=0}^{\infty} \frac{e^{-\lambda}\lambda^i}{i!} i\left(1 - \frac{1}{i+1}\right) \\ &= -f_1(\lambda) + 1 - \frac{f_1(\lambda)}{\lambda} \end{aligned}$$

and $\lim_{\lambda\to 0} f_1(\lambda) = 0$. Both the equation and the boundary condition are satisfied for $f_1(\lambda) = (\lambda - 1 + e^{-\lambda})/\lambda$ which hence is the unique solution. QED

REFERENCES

[1] Max Planck Institute (March 18, 2008). "Glasnost: Results from tests for BitTorrent traffic blocking". [Online]. http://broadband.mpi-sws.org/transparency/results/ Retrieved April 3, 2011.

[2] Shalunov, Stanislav, Greg Hazel, Janardhan Iyengar, and Mirja Kuehlewind. "Low extra delay background transport (LEDBAT)." *draft-ietf-ledbat-congestion-04. txt* (2010).

[3] Rossi, Dario, Claudio Testa, Silvio Valenti, and Luca Muscariello. "LEDBAT: the new BitTorrent congestion control protocol." In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pp. 1-6. IEEE, 2010.

[4] Kuzmanovic, Aleksandar, and Edward W. Knightly. "TCP-LP: low-priority service via end-point congestion control." *Networking, IEEE/ACM Transactions on* 14, no. 4 (2006): 739-752.

[5] Venkataramani, Arun, Ravi Kokku, and Mike Dahlin. "TCP Nice: A mechanism for background transfers." *ACM SIGOPS Operating Systems Review* 36, no. SI (2002): 329-343.

[6] Carofiglio, Giovanna, Luca Muscariello, Dario Rossi, and Claudio Testa. "A hands-on assessment of transport protocols with lower than best effort priority." In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pp. 8-15. IEEE, 2010.

[7] Courcoubetis Costas, and Antonis Dimakis. "Fair background data transfers of minimal delay impact." In *INFOCOM, 2012 Proceedings IEEE*, pp. 1053-1061. IEEE, 2012.

[8] Yang, Y.R. and Lam, S.S. "General AIMD congestion control." In *Proceedings of the 2000 International Conference on Network Protocols*, 2000.

[9] Frank Kelly. "Charging and Rate Control for Elastic Network Traffic." European Transactions on Telecommunications, vol. 8, no. 1, pp. 33–37, Jan. 1997.

[10] P. Key, L. Massoulie, and M. Vojnovic, "Farsighted users harness network time-diversity," in *INFOCOM 2005. Proceedings IEEE*, vol. 4, 2005, pp. 2383–2394.

[11] Sally Floyd and Van Jacobson. "Traffic phase effects in packet-switched gateways." ACM SIGCOMM Computer Communication Review, vol. 21, no. 2, pp. 26-42, April 1991.

[12] The Network Simulator ns-2. [Online]: http://www.isi.edu/nsnam/ns/

[13] Dario Rossi:: Software /Ledbat [Online] Available: http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.LEDBAT

[14] Shaked, M. and Shanthikumar, J.G. Stochastc Orders, Springer Series in Statistics, Springer, NY,2007.