Emulating Low-priority Transport at the Application Layer: A Background Transfer Service

Peter Key
Microsoft Research
Roger Needham Building
7 J J Thomson Avenure
Cambridge, CB3 0FB, U.K.
peterkey@microsoft.com

Laurent Massoulié
Microsoft Research
Roger Needham Building
7 J J Thomson Avenure
Cambridge, CB3 0FB, U.K.
lmassoul@microsoft.com

Bing Wang
Computer Science
Department
University of Massachusetts
Amherst, MA 01003-4610
bing@cs.umass.edu

ABSTRACT

Low priority data transfer across the wide area is useful in several contexts, for example for the dissemination of large files such as OS updates, content distribution or prefetching. Although the design of such a service is reasonably easy when the underlying network supports service differentiation, it becomes more challenging without such network support. We describe an application level approach to designing a low priority service — one that is 'lower than best-effort' in the context of the current Internet. We require neither network support nor changes to TCP. Instead, we use a receive window control to limit the transfer rate of the application, and the optimal rate is determined by detecting a change-point. We motivate this joint control-estimation problem by considering a fluid-based optimisation framework, and describe practical solutions, based on stochastic approximation and binary search techniques. Simulation results demonstrate the effectiveness of the approach.

Keywords

Background transfer, Low priority, Stochastic Approximation, Binary Search, Application Reaction

General Terms

Performance, Algorithms

Categories and Subject Descriptors

[**C.4 Performance of Systems; C.2.2 Network Protocols**]

1. INTRODUCTION

We define background transfers of data or information to be transfers across a network whose aim is to transfer the data as quickly as possible, but without impairing the performance of foreground transfers. In other words, background transfers are lower-priority than foreground flows, but seek to fully utilize spare capacity. In the context of the current Internet, we are looking for a 'lower than best effort' quality of service, i.e. lower than the current base-line quality offered. Examples of background transfer include large file backup, transferring updates to the current operating system (e.g., Microsoft's Background Intelligent Transfer Service, (BITS)), contents prefetching [1] and distribution, storage management and caching in peer-to-peer systems [2].

One way to achieve low-priority background transfer is to use priority queues at routers. However, this is not currently practical due to lack of both consensus and router support. Two alternative approaches, TCP Nice [3] and TCP-LP [4] require no support from the routers but instead modify the congestion detection and avoidance in TCP to achieve a low-priority transfer. They react both earlier and more aggressively than standard loss-based TCP (Reno). End-to-end delays are used to trigger a congestion signal before loss occurs, a TCP Vegas style approach, while more aggressive back-off mechanisms than those of standard TCP are used to give a lower share of network resources. Although only sender-side modification is required, this modification of TCP stack may still be difficult to deploy widely.

In this paper, we design an application-level approach for background transfers that uses TCP as the underlying transport protocol. Two components are needed: to infer the available capacity and to adjust the sending rate of the background transfer accordingly. There is a large literature on available capacity inference, e.g. [6, 7, 8]. However, most current work relies on probing the end-to-end path, which requires coordination between sender and receiver. Furthermore, UDP probes can be blocked by firewalls and also impose extra load on the network. Using TCP directly to infer the available capacity, on the other hand, tends to cause overestimation [8].

In our application-level approach, we tightly couple the available capacity inference and rate adjustment: the rate of the background transfer is controlled by adjusting the receiver-advertised window size, which enforces a limitation on the rate used by the application. In turn, the rate obtained for a given receiver window is used to infer whether that rate is above or below the available capacity, which in turn triggers an adaptation of the receiver window. This is based on the fact that the actual TCP sending window is the minimum of the receiver window and the congestion window. By controlling the receiver window size, we also avoid overestimating the available capacity as observed in [8].

Receiver based capacity sharing is studied in [9, 10] to achieve service differentiation in the access links. In [11], receive window limitation is used as a rate control to mitigate the network effects of prefetching. In [12], the receiver advertised window is adjusted at a web cache to achieve proportional fairness among flows. In [13],

SIGMETRICS/Performance'04, June 12–16, 2004, New York, NY, USA.

the authors use the receiver advertised window to limit the rate of the TCP video traffic on a VPN link between a video server and a proxy. Receiver window adjustment is also leveraged to increase TCP throughput [14, 15]. However, all the above are in contexts different from background transfer. Furthermore, none of the above studies couple the receiver window adjustment together with the available capacity estimation in order to fully utilize the available capacity.

The outline of this paper is as follows. We identify the relationship between receiver window and achieved rate in Section 2, using fluid-flow models and an optimisation framework to show how the goodput of the background flow can be used to detect the change point at which the background flow starts to interfere with the foreground flow. In other words, we can identify a critical value of the receiver window which we wish to target. We also give some empirical validation of this approach. In section 3 we discuss control objectives, and show how target relationships between foreground and background flows can be expressed in a utility maximisation framework, and discuss fairness among background flows. In Section 4, we propose two practical application-level approaches for background transfer: one is based on binary search and the other is based on stochastic approximation [16]. We evaluate the performance of the two approaches using *ns* simulation in Section 5, where we also show the performance of TCP-LP [4].

2. RELATIONSHIP BETWEEN RECEIVER WINDOW AND ACHIEVED RATE

In this section, we investigate the relationship between the achieved rate and the receiver window advertised by the receiver. The findings of this section motivate the objectives discussed in the next section and underpin the method to be described in section 4. We investigate this relationship under two different assumptions: first, in Section 2.1, we assume buffer space at all links is large enough to prevent any packet losses. Secondly, in Section 2.2, we then consider the case where buffering is minimal and flows may experience loss.

We consider a general network comprising a collection $\mathcal{L}$ of links, each link ℓ having an associated nominal capacity C_ℓ. There is a set $\mathcal{R}$ of flows, each flow $r \in \mathcal{R}$ identifies a subset of $\mathcal{L}$, the links it uses. This set of flows is divided into two subsets, namely the set $\mathcal{F}$ of foreground flows and the set $\mathcal{B}$ of background flows. Each foreground flow implements TCP's congestion control. Finally, when there is only one background flow, we denote it by b.

We also make use of the following notations for each flow r, whether foreground or background. We let w_r denote the receiver window, and $cwnd_r$ the (time-average of the) TCP congestion window. It is assumed to be a function of the corresponding loss rate p_r only, i.e. $cwnd_r = \varphi(p_r)$ for a given function φ. The round-trip propagation delay is denoted by τ_r, and the queueing delay by d_r, hence the total round-trip delay reads $rtt_r = \tau_r + d_r$. As the effective window is the minimum of the congestion window and the receiver window, these parameters combine to provide the throughput formula:

$$x_r = \frac{\min(w_r, cwnd_r)}{\tau_r + d_r}. \tag{1}$$

Finally, the goodput is the throughput, reduced by the losses, hence the goodput formula

$$y_r = (1 - p_r)\frac{\min(w_r, cwnd_r)}{\tau_r + d_r}. \tag{2}$$

In the remainder of this section, we focus on the case of a single background flow b, and investigate the relationship between its goodput y_b and its receiver window, w_b. For the sake of tractability, we treat the case where all flows are constrained by delays and there are no losses (Section 2.1), and the case where there are no queueing delays and the flows are constrained by losses (Section 2.2). The behaviour as predicted by the analysis in these sections is confirmed by simulations in Section 2.3.

2.1 The delay constrained case

When all links have sufficiently large buffers, there are no losses and flow rates are controlled by their receiver window only, not by their congestion window. The throughputs and goodputs coincide, and the throughput formula (1) simplifies to

$$x_r = \frac{w_r}{\tau_r + d_r}. \tag{3}$$

Under FIFO scheduling at all links, Massoulié and Roberts [17] showed that the achieved rates $x = (x_r)$ solve the optimisation problem

$$\begin{aligned} &\text{Maximise} \quad \sum_{r\in\mathcal{R}} w_r \log x_r - x_r\tau_r \\ &\text{subject to} \quad \sum_{r:\ell\in r} x_r \le C_\ell, \quad \ell \in \mathcal{L} \quad \text{over} \quad (x_r) \in \mathbb{R}_+^{\mathcal{R}}. \end{aligned} \tag{4}$$

The Lagrangean of the latter optimisation problem reads

$$L = \sum_{r\in\mathcal{R}} w_r \log x_r - x_r\tau_r + \sum_{\ell\in\mathcal{L}} \mu_\ell \left(C_\ell - \sum_{r:\ell\in r} x_r \right), \tag{5}$$

where μ_ℓ is the Lagrange multiplier associated with the capacity constraint C_ℓ. This identification made in [17] follows by interpreting the queueing delay at link ℓ as the Lagrange multiplier μ_ℓ, and viewing the queueing delay d_r in the formula (3) as the sum of the multipliers associated with the capacity constraints along the path of flow r.

Let $x_r(w_b)$ denote the corresponding unique optimal solution, where we have emphasised its dependence on the parameter of interest, w_b, that is the receiver window of the unique background flow. The rates $x_r(0)$ correspond to the allocations achieved when the background flow is absent. Hence the spare capacity available to flow b can be written

$$x_b^* = \min_{\ell\in b} \left(C_\ell - \sum_{r:\ell\in r} x_r(0) \right).$$

We assume that $x_b^* < \min_{\ell\in b}(C_\ell)$, so that foreground and background flows may interact. We then have the following result:

THEOREM 1. *Let $w_b^* := x_b^*\tau_b$. In the present setting, foreground flows are affected by the background flow if and only if $w_b > w_b^*$. In addition, the function $x_b(w_b)$ is linear on $[0, w_b^*]$ with slope $1/\tau_b$, such that $x_b(w_b) < w_b/\tau_b$ on $(w_b^*, +\infty)$ and the function $x_b(w_b)/w_b$ is non-increasing on $\mathbb{R}_+$.*

PROOF. By considering the Lagrangean (5), it is easily seen that for $w_b \in [0, w_b^*]$, the problem (4) is solved by taking $x_r(w_b) = x_r(0)$, $r \in \mathcal{F}$, and $x_b(w_b) = w_b/\tau_b$. Thus, when w_b is in the range $[0, w_b^*]$, the background flow does not interfere with any other flows. When $w_b > w_b^*$, necessarily the Lagrange multiplier μ_ℓ associated with the capacity constraint C_ℓ for some $\ell \in b$ needs to be strictly positive, hence $x_b(w_b)$, which equals $w_b/(\tau_b + \sum_{\ell\in b}\mu_\ell)$, is indeed strictly less than w_b/τ_b. The last statement is established by noticing that, if we perform the optimisation in (4) first on the x_r, $r \in \mathcal{F}$, and then on x_b, the solution $x_b(w_b)$ is identified with the solution of the maximisation problem:

$$\text{Maximise} \quad w_b \log x_b - \tau_b x_b + \psi(x_b) \quad \text{over} \quad x_b \ge 0,$$

where the function ψ is defined as

$$\psi(x_b) = \max_{(x_r)\in\mathcal{S}(x_b)} \left\{ \sum_{r\in\mathcal{F}} w_r \log x_r - x_r\tau_r \right\}$$

where $\mathcal{S}(x_b) = \left\{(x_r) \in \mathbb{R}_+^{\mathcal{F}} : \sum_{r:\ell\in r} x_r \le C_\ell - x_b \mathbf{1}_{\ell\in b}\right\}$ is the set of achievable foreground flows given the background flow rate x_b. It is easy to see that ψ is non-increasing and concave. Assuming for simplicity that ψ is differentiable[1], the function $x_b(w_b)$ also reads $f^{-1}(w_b)$, where $f(x) = x(\tau_b - \psi'(x))$. The function $f^{-1}(w_b)/w_b$ is non-increasing if and only if the function $f(x)/x$ is non-decreasing, that is if and only if $-\psi'(x)$ is non-decreasing. The latter property is implied by concavity of ψ. Finally, observing that f and hence f^{-1} are non-decreasing, it can be seen that necessarily, for any $w_b > w_b^*$, one has $x_b(w_b) > x_b^*$. It then follows that for $w_b > w_b^*$, there must exist some foreground flow r such that $x_r(w_b) < x_r(0)$. □

2.2 The loss constrained case

We now assume that buffers are small, so that queueing delays can be neglected, and congestion control relies on losses rather than on buffering. The throughput formula (1) then simplifies to

$$x_r = \frac{\min\left(w_r, \varphi(p_r)\right)}{\tau_r}. \tag{6}$$

Many studies have shown that in such a context, the rates achieved by TCP flows may be interpreted as solving an implicit optimisation problem; see e.g. [19, 20, 21].

We follow this approach, and assume that the rates $(x_r)_{r\in\mathcal{R}}$ are the solution to the problem:

$$\begin{aligned} \text{Maximise} \quad & \sum_{r\in\mathcal{R}} U_r(x_r) - \sum_\ell \Gamma_\ell \left(\sum_{r:\ell\in r} x_r \right) \\ \text{subject to} \quad & 0 \le x_r \le \frac{w_r}{\tau_r} \quad \text{over } (x_r) \in \mathbb{R}_+^{\mathcal{R}}. \end{aligned} \tag{7}$$

The constraint captures the impact of the receiver window size w_r. In the above, the so-called utility functions U_r are assumed to be strictly concave and increasing. It has been suggested (see [22], [19]) that adequate choices for modelling TCP capacity sharing could be $U_r(x) = -1/(\tau_r^2 x)$, or $U_r(x) = \tau_r^{-1}\arctan(\tau_r x)$. The correspondence between the solution to the optimisation problem (7) and the throughput formula (6) can be formally established by identifying the functions $\tau_r^{-1}\varphi(\cdot)$ in (6) with ${U'}_r^{-1}(\cdot)$ (for instance, the latter *arctangent* utility function corresponds to the choice $\varphi(p) = \sqrt{(1-p)/p}$), and by assuming that loss rates p_r are the sums of link loss rates p_ℓ at links along the path of flow r. The formal identification is then valid when the penalty function Γ_ℓ in (7) is related to the loss probability $p_\ell(x)$ at link ℓ when it carries data at rate x via the equation

$$\Gamma_\ell(x) = \int_0^x p_\ell(y)dy.$$

A special choice advocated in [19] consists in setting $p_\ell(y) = (y - C_\ell)^+/y$. The results below rely solely on the assumption that the utility functions U_r are strictly convex increasing, and that the loss rate functions p_ℓ are non-decreasing and continuous. As before, let $(x_r(w_b))_{r\in\mathcal{R}}$ denote the rates achieving the maximum in the optimisation problem (7) when the receiver window of background flow b is w_b. We also make use of the notation

$$\pi_\ell(w_b) := p_\ell\left(\sum_{r\in\mathcal{R}:\ell\in r} x_r(w_b) \right).$$

The function π_ℓ is naturally interpreted as the loss probability along link ℓ. We shall also denote the goodput obtained by the background flow as

$$y_b(w_b) := x_b(w_b)\left(1 - \sum_{\ell\in b} \pi_\ell(w_b)\right).$$

Again, $x_r(0)$ is the rate obtained by foreground flow r in the absence of background flows. We define the spare capacity available to flow b as

$$x_b^* := \min_{\ell\in b} \sup\left\{ x > 0 : p_\ell(x + \sum_{r\in\mathcal{F}:\ell\in r} x_r(0)) = 0 \right\}.$$

We assume that there exists a link $\ell \in b$ and a flow r', $\ell \in r'$ such that $x_{r'}(0) > 0$ and $p_\ell(x_b^* + \sum_{r\in\mathcal{F}:\ell\in r} x_r(0) + z) > 0$ for all $z > 0$, so that background and foreground flows could interact. In this context, similarly to Theorem 1, we have the following.

THEOREM 2. *Let $w_b^* := x_b^*\tau_b$. In the present setting, foreground flows are affected by the background flow if and only if $w_b > w_b^*$. Moreover, the goodput $y_b(w_b)$ received by the background flow is linear in w_b on the interval $[0, w_b]$, with slope $1/\tau_b$, is such that $y_b(w_b)/w_b < \tau_b^{-1}$ when $w_b > w_b^*$, and the function $y_b(w_b)/w_b$ is non-increasing in w_b.*

PROOF. (sketched). We only provide those steps that differ significantly from those in the proof of Theorem 1. It is readily seen that, for $w_b \in [0, w_b^*]$, the optimisation problem (7) is solved by setting $x_r(w_b) = x_r(0)$, $r \in \mathcal{F}$, and $x_b(w_b) = w_b/\tau_b$. The case where $w_b > w_b^*$ can be further divided into two sub-cases, according to whether the constraint imposed by w_b is binding or not. Denote by $\hat{w}_b$ the critical value for w_b where the receiver window constraint ceases to be binding. In the interval $(w_b^*, \hat{w}_b)$, one has $x_b(w_b) \equiv w_b/\tau_b$. Similarly to the proof of Theorem 1, define

$$\psi(x_b) := \max_{(x_r)\in\mathbb{R}_+^{\mathcal{F}}} \left\{ \sum_{r\in\mathcal{F}} U_r(x_r) - \sum_\ell \Gamma_\ell\left(\sum_{r:\ell\in r} x_r \right) \right\}.$$

This function is non-increasing and concave. The interval $(w_b^*, \hat{w}_b)$ may then be alternatively characterised as the one for which

$$U_b'(w_b/\tau_b) + \psi'(w_b/\tau_b) > 0.$$

In addition, it can be shown that the derivative $-\psi'(w_b/\tau_b)$ coincides with the loss rate $\sum_{\ell\in b}\pi_\ell(w_b)$ over the range $(w_b^*, \hat{w}_b)$. By concavity of ψ, $-\psi'$ is non-decreasing. As a result, the normalised goodput function $y_b(w_b)/w_b$, which reads $\tau_b^{-1}(1 - \sum_{\ell\in b}\pi_\ell(w_b))$, is indeed non-increasing. Finally, in the range $w_b > \hat{w}_b$, one has $y_b(w_b) \equiv y_b(\hat{w}_b)$, and hence the normalised goodput is decreasing there as well. □

As in the context of delay–constrained flows, we observe that goodput normalised by receiver window is constant over a range $[0, w_b^*]$, and decreasing over $(w_b^*, +\infty)$, where the critical window w_b^* is precisely the one we should use to maximise background goodput while not interfering with foreground flows.

[1] This is actually not the case; however a rigorous argument can be constructed along the same lines, based on sub-differentials of convex functions (see Rockafellar [18], Chapter 23) rather than ordinary differentials.

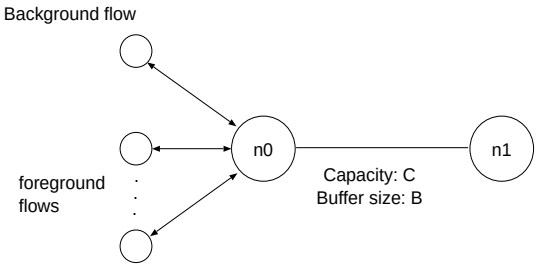

Figure 1: Topology in *ns*: the link between node node $n0$ and $n1$ forms a bottleneck link.

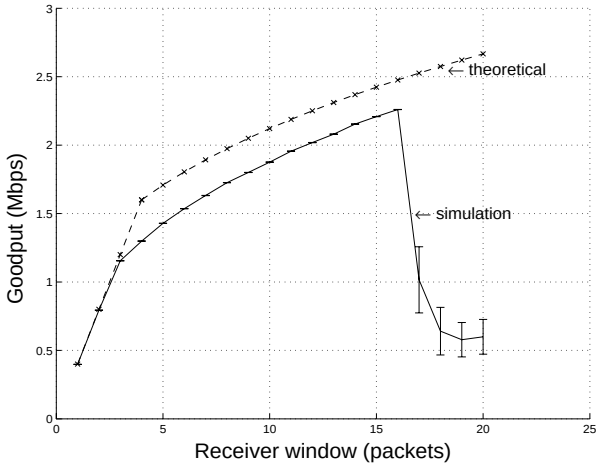

Figure 2: The goodput of the background flow versus the window size with 8 foreground flows, when $C = 2000$ packets per second, $B = 40$ packets.

2.3 Experimental validation

Although we have analysed only two extreme cases, namely large buffers (no loss), and no buffering but losses, we expect the same qualitative behaviour to hold in mixed situations where there is both significant buffering and loss. We now confirm this expectation using *ns* simulations [23].

The topology used in *ns* is shown in Figure 1. The link between nodes $n0$ and $n1$ forms a bottleneck link with capacity C. The buffer size of node $n0$ is B. Here we have a number N of foreground flows, each a long-lived TCP connection, with a round trip propagation delay τ_f and a maximum window size of w_f. The generic network optimisation problem (4), specialised to the present situation, reads

$$\text{Maximise} \quad N(w_f \log x_f - \tau_f x_f) + w_b \log x_b - \tau_b x_b \qquad (8)$$

subject to $Nx_f + x_b \leq C$. We would expect its solution to predict accurately the actual achieved rates in the absence of losses, that is when the buffer size B is large.

We now report experiments for $N = 8$ foreground flows, each with characteristics $w_f = 20$ packets and $\tau_f = 100$ ms, a capacity C of 2000 packets per second. τ_b is set to 10 ms. The packet size is fixed to 500 bytes. The maximum aggregate bandwidth usage of the foreground TCP flows is then 1600 packets per second. The foreground TCP flows are started at time 0 and the background flow is started at the 10th second. The window size of the background flow is initially 1 packet, and increases by 1 packet every 40 seconds. For each background flow window size, we obtain the

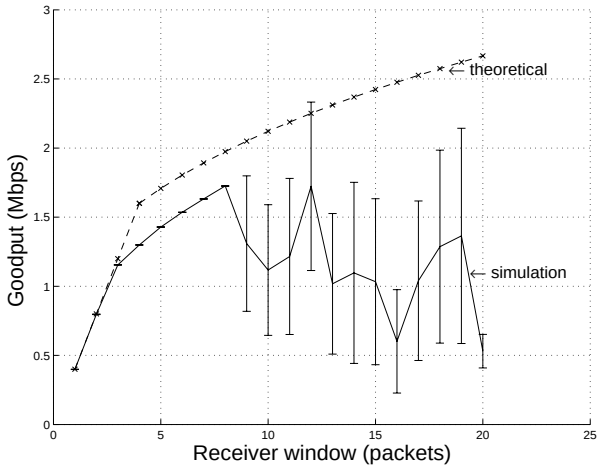

Figure 3: The goodput of the background flow versus the window size with 8 foreground flows, when $C = 2000$ packets per second, $B = 20$ packets.

average background flow goodput every second with a total of 40 samples. We then obtain the mean and the 95% confidence intervals from the 40 samples.

The solid line in Figure 2 shows the background flow goodput as a function of the window size, with 95% confidence intervals, when $B = 40$ packets. The result from solving the simple optimization problem (8) is also shown in the figure, which indicates that the optimal window size for the background flow is 4 packets. From the simulation, the optimal value is lower (3 packets). Packet losses start to occur when $w_b = 16$ packets. Figure 3 shows the background flow goodput versus the window size for a smaller buffer size at node $n0$ ($B = 20$ packets). Packet losses start to occur when $w_b = 8$ packets. Note that at this point, when the background flow is too aggressive, loss occurs and the variance in the background flow goodput increases dramatically; furthermore, the goodput is not necessarily an increasing function of the window size.

3. CONTROL OBJECTIVES

The previous results suggest that, when only one background flow b is present, one could adapt its receiver window w_b, based on its achieved goodput, so as to tune it to the critical value w_b^* at which interference with foreground flows would start occurring.

In the more general situation where there are several background flows $b \in \mathcal{B}$, we still aim at efficiently utilising the spare capacity left by foreground flows $f \in \mathcal{F}$ without interfering significantly with those flows. However, we have the additional objective of ensuring that competing background flows $b \in \mathcal{B}$ share the available bandwidth fairly.

Assuming that for each background flow $b \in \mathcal{B}$ we can identify the corresponding round-trip propagation delay τ_b efficiently, then we can design adaptation rules which tune the receiver window w_b so as to enforce the identity

$$\frac{y_b}{w_b} = \frac{1}{\tau_b} - \epsilon_b(y_b), \qquad (9)$$

where y_b is the corresponding goodput, and $\epsilon_b(\cdot)$ is a function of our choice. Specifically, we shall consider the special case

$$\epsilon_b(y) = \frac{\alpha_b}{1 + \beta_b y}, \qquad (10)$$

where α_b and β_b are two non-negative parameters to be specified. Of particular interest is the case where $\beta_b = 0$ and hence the function $\epsilon_b(\cdot)$ is constant. Note that it is natural to require α_b to scale as $1/\tau_b$, and hence we can write $\alpha_b = \alpha_b^*/\tau_b$ where $= \alpha_b^*$ is some constant, independent of the value of τ_b. For the relationship (9) to be meaningful we require

$$\alpha < \frac{1}{\tau_b} \text{ implying } \alpha_b^* < 1.$$

It is important to understand what is the impact of enforcing relation (9) rather than setting w_b to its ideal value, w_b^*, since now background flows may interfere with foreground flows as a result of this relaxed control objective.

We start our analysis of such interference by considering the same simple situation as in Section 2.3, with one background flow b and N homogeneous TCP foreground flows sharing a single link of capacity of the C, with enough buffering to avoid any data loss. We also assume that $\beta_b = 0$, that is $\epsilon_b(\cdot) \equiv \alpha_b$. Let w_f denote the window size of the foreground TCP flows, and w_b the equilibrium window size of the background flow b. The corresponding equilibrium rates x_f and x_b are again characterised as the unique solution to the optimisation problem (8). It is then straightforward to show

THEOREM 3. *In the present setting, when round-trip times are equal ($\tau_f = \tau_b = \tau$), the relative reduction in the rate of foreground flows in the presence of the background flow caused by using a positive threshold α_b is bounded above by*

$$D = \alpha_b \tau.$$

PROOF. If the capacity is not fully used after introduction of the background flow b, there is no reduction in the foreground flow rates. If it is fully used, when the round trip times are the same, solving the optimisation (8) gives

$$x_b = \frac{w_b}{w_b + N w_f} C = \left(\frac{1}{\tau} - \alpha_b\right) w_b$$
$$x_f = \frac{w_f}{w_b + N w_f} C$$

and hence

$$x_f = \left(\frac{1}{\tau} - \alpha_b\right) w_f.$$

In the absence of the background flow, $x_f = \min(w_f/\tau, C/N)$. With D the reduction in the TCP rate caused by the background flow, if $x_f = w_f/\tau$ then

$$D = \frac{w_f/\tau - \left(\frac{1}{\tau} - \alpha_b\right) w_f}{w_f/\tau} = \alpha_b \tau.$$

On the other hand, if $x_f = C/N$ then $w_f \geq \tau C/N$. Hence $x_f \geq \frac{(1/\tau - \alpha_b)\tau C}{N}$ and

$$D \leq \frac{C/N - (1/\tau - \alpha_b)\tau C/N}{C/N} = \alpha_b \tau.$$

□

Note also that if we put $\alpha_b = \alpha_b^*/\tau_b$, then it follows from equations (9) and (1) that α_b^* is equivalent to the relative queuing delay

$$\alpha_b^* = \frac{d_b}{d_b + \tau_b}.$$

Hence for this case where $\beta_b = 0$, setting α_b^* to some fraction both bounds the relative damage to other stream and bounds the relative queueing delay the background flow sees.

We now proceed to a more systematic study of the equilibrium allocations stemming from relation (9), for general network topologies, both in the delay-constrained and in the loss-constrained cases.

THEOREM 4. *Consider the case of delay-constrained flows, with goodputs as in (3). Assume that foreground flows $f \in \mathcal{F}$ have their receiver window set to a given value w_f, while background flows $b \in \mathcal{B}$ have their receiver window w_b adjusted so as to satisfy (9). Then the corresponding equilibrium achieved rates x_r solve the maximisation problem*

$$\textit{Maximise} \sum_{r \in \mathcal{R}} U_r(x_r) \tag{11}$$

under the link capacity constraints, where the utility functions U_r are given by

$$U_f(x) = w_f \log(x) - \tau_f x, \quad f \in \mathcal{F}, \tag{12}$$

$$U_b(x) = \int_0^x \frac{\tau_b^2 \epsilon_b(y)}{1 - \tau_b \epsilon_b(y)} dy, \quad b \in \mathcal{B}. \tag{13}$$

In particular, for $\epsilon_b(\cdot)$ as in (10), this reads

$$U_b(x) = \frac{\tau_b^2 \alpha_b}{\beta_b} \log\left(1 + \frac{\beta_b}{1 - \tau_b \alpha_b} x\right) \tag{14}$$

where the case $\beta_b = 0$ is understood as the limit as $\beta \to 0$.

PROOF. From equations (3) and (9), one finds that

$$d_f = w_f/x_f - \tau_f, \quad f \in \mathcal{F},$$
$$d_b = \tfrac{\tau_b^2 \epsilon(y_b)}{1 - \tau_b \epsilon_b(y_b)}, \quad b \in \mathcal{B}.$$

The queueing delay μ_ℓ at a link ℓ is non-negative, and positive only when the corresponding capacity C_ℓ is saturated. This allows one to identify the queueing delay d_r along the path of a flow r with the sum $\sum_{\ell \in r} \mu_\ell$ of Lagrange multipliers associated with the corresponding capacity constraints, and thus the latter conditions coincide with the stationarity conditions of the Lagrangean of the optimisation problem (11). The expression (14) is readily obtained by replacing $\epsilon_b(\cdot)$ by its expression and performing the integration in (13). □

By solving the optimisation problem (11), one can predict to what extent, for the corresponding fluid model of performance, background flows will interfere with foreground flows at equilibrium. We can also assess whether capacity is going to be shared fairly among background flows. For instance, for strictly decreasing $\epsilon_b(\cdot)$, the associated utility functions U_b are strictly concave, and we expect fair allocations among background flows. On the other hand, when $\epsilon_b(\cdot) \equiv \alpha_b$, U_b is linear, and thus several allocations may achieve the maximum in (11): one can construct examples with two background flows along the same path, and where there is an allocation maximising (11) for which one of the background flows is starved and not the other. Fairness is not a priori granted with constant ϵ_b.

We now turn to the case of loss-constrained flows.

THEOREM 5. *Consider the case of flows constrained only by losses, with throughputs as in (6). Assume again that foreground flows $f \in \mathcal{F}$ have given receiver windows w_f, while background flows $b \in \mathcal{B}$ have their receiver windows w_b adjusted so as to satisfy (9). Assume also that the loss rate along a path is the sum of the loss rates at the links on the path. Assume that, for all $b \in \mathcal{B}$, the function g_b defined by*

$$g_b(p) := \frac{1}{1-p} \epsilon_b^{-1}\left(\frac{p}{\tau_b}\right). \tag{15}$$

is strictly decreasing. Then the equilibrium throughputs x_r, $r \in \mathcal{R}$, solve the optimisation problem

$$\textit{Maximise} \sum_r U_r(x_r) - \sum_\ell \Gamma_\ell \left(\sum_{r:\ell \in r} x_r \right) \tag{16}$$

under the constraints $x_f \leq w_f/\tau_f$, $f \in \mathcal{F}$, where the utility functions U_r are characterised by

$$U_f(x) = \int_0^x \varphi^{-1}(\tau_f z) dz, \quad f \in \mathcal{F}, \tag{17}$$

$$U_b(x) = \int_0^x \min\left(\varphi^{-1}(\tau_b z), g_b^{-1}(z)\right) dz, \quad b \in \mathcal{B}. \tag{18}$$

PROOF. As the function φ is assumed strictly decreasing, the relation (6) yields, for all $f \in \mathcal{F}$,

$$U_f'(x_f) = p_f + \theta_f, \tag{19}$$

where θ_f is non-negative, and positive only when $x_f = w_f/\tau_f$. Indeed, it holds with $\theta_f = 0$ when $x_f < w_f/\tau_f$, in view of the expression of U_f'. When $x_f = w_f/\tau_f$, it holds that $\varphi(p_f) \geq w_f$, and hence $U_f'(x_f) \geq p_f$, which implies (19). The coefficient θ_f may thus be interpreted as the Lagrange multiplier associated with the constraint $x_f \leq w_f/\tau_f$. Moreover, (6) together with (9) entails that, for all $b \in \mathcal{B}$,

$$x_b = \min\left(\frac{1}{1-p_b} \epsilon_b^{-1}(p_b/\tau_b), \frac{\varphi(p_b)}{\tau_b} \right).$$

Recognising the first term in the minimum as $g_b(p_b)$, this also reads

$$U_b'(x_b) = p_b. \tag{20}$$

Finally, the loss rate p_r along the path of a flow r also reads, for all $r \in \mathcal{R}$,

$$p_r = \sum_{\ell \in r} \Gamma_\ell' \left(\sum_{s:\ell \in s} x_s \right). \tag{21}$$

The equations (19), (20) and (21) coincide with the stationarity conditions of the Lagrangean of the optimisation problem (16). Since this is a strictly concave minimisation problem, it admits a unique minimum, which is achieved by the equilibrium throughputs x_r, $r \in \mathcal{R}$. □

For the sake of illustration, when $\epsilon(\cdot)$ is given by (10), tedious but straighforward calculations yield the following expression for $g_b^{-1}(x)$:

$$g_b^{-1}(x) = \frac{1}{2}\left(1 + \frac{1}{x\beta_b} - \sqrt{\left(1 + \frac{1}{x\beta_b}\right)^2 - \frac{4\alpha_b\tau_b}{x\beta_b}} \right). \tag{22}$$

In the special case where the function $\epsilon_b(\cdot)$ is constant, which is obtained by letting β_b tend to zero, this expression simplifies to $g_b^{-1}(x) \equiv \alpha_b\tau_b$. This may then be injected into (17) in order to determine the utility function U_b associated with background flow b, and then assess to what extent it would interfere with foreground flows. We now consider ways of achieving the proposed control objectives.

4. ALGORITHMS

In this section we discuss practical methods for enforcing the control objectives discussed in the previous section, namely to tune the receiver window w_b of background flow b so as to enforce relation (9). We assume that time is divided in control intervals, each of length T. In the nth interval, a receiver window of size $w_b(n)$ is applied to flow b, and an amount $R_b(n)$ of data is received, both $w_b(n)$ and $R_b(n)$ being expressed in the same unit, which is a fixed number of bytes u.

Let $\rho_b(n) := R_b(n)/w_b(n)$. When there is only one background flow b, in view of the results of Section 2, we postulate the following model for the observed quantities $\rho_b(n)$:

$$\rho_b(n) = \phi_b\left(w_b(n)\right) + Z_n, \tag{23}$$

where Z_n represents observation noise, assumed to be centred, and the function ϕ_b is the normalised goodput function multiplied by the control interval length T, which, in view of Theorems 1 and 2, is assumed to be non-increasing, and such that:

$$\phi_b(w) \begin{cases} \equiv \rho_b := T/\tau_b, & \text{if } w \in [0, w_b^*], \\ < \rho_b & \text{if } w > w_b^*. \end{cases}$$

We would expect such a model to be accurate when the control interval length T is long compared to the time needed for TCP to adjust the rates of the competing flows after a change in some receiver window.

In the sequel, we aim at adjusting the receiver window w_b so as to enforce (9), and choose the function $\epsilon(\cdot)$ to be constant. This can also be written as

$$\phi_b(w_b) = \rho_b - \epsilon, \tag{24}$$

where ϵ is now a positive constant. In the sequel we denote by w^* the solution to this equation. Note $w^* \geq w_b^*$ since ϵ is positive.

In the two control methods we propose next, we maintain an Exponentially Weighted Moving Average (EWMA) estimate $\hat{\rho}_b(n)$ of ρ_b,

$$\hat{\rho}_b(n) = \begin{cases} (1-\delta)\hat{\rho}_b(n-1) + \delta\rho_b(n) \\ \text{if } \rho_b(n) \geq \hat{\rho}_b(n-1) - \epsilon', \\ \hat{\rho}_b(n-1) \text{ otherwise.} \end{cases} \tag{25}$$

This features two positive parameters $\delta \in (0,1)$ and ϵ', and can be initialised by taking $\hat{\rho}_b(1) = \rho_b(1)$.

We now describe two approaches for enforcing relation (24), one based on binary search and the other based on stochastic approximation. In Sections 4.1 and 4.2, the methods are more specifically motivated in the context of a single background flow. Section 4.3 discusses stability issues specific to the multiple background flows case.

4.1 The Binary Search Method

```
w_b(n) ← w_min

loop {
    if (ρ_b(n) − ρ̂_b(n − 1) < −ε) {
        w_max ← w_b(n)
        return w_max
    }
    w_b(n + 1) ← 2w_b(n)
    n ← n + 1
}
```

Figure 4: Initialisation of w_{max}.

In this approach we maintain a search range $[w_{min}, w_{max}]$ for the desired window w^*. The lower limit w_{min} is initially set to 1, and the upper limit w_{max} may be set to a system dependent limit. In practice, we make a preliminary search for a suitable value

of w_{max} by starting from $w_1 = 1$, and doubling $w_b(n)$ in each subsequent interval, until the condition $\rho_b(n) > \hat{\rho}_b(n-1) - \epsilon$ fails, at which point we set $w_{max} = w_b(n)$. This is shown in Figure 4. A 'time-out' is triggered if the number of units received by the background flow in any control interval is 0, in which case we set $w_{min} = 1$ and start the process to find a new w_{max}.

At the beginning of a control interval, we set

$$w_b(n) = \lfloor (w_{min} + w_{max})/2 \rfloor \tag{26}$$

where $\lfloor \cdot \rfloor$ denotes integer part. At the end of the control interval, if $\rho_b(n) > \hat{\rho}_b(n-1) - \epsilon$, we let $w_{min} = w_b(n)$; otherwise we let $w_{max} = w_b(n)$. Such standard binary search proceeds until the difference $w_{max} - w_{min}$ is 0 or 1. In the absence of noise, and in a static environment we would then be guaranteed to have reached the optimal window w^* satisfying (24), and the search could be stopped there.

In practice we are in a dynamic environment, with noisy observations, and we describe how we proceed when the search interval has length at most 1. If $\rho_b(n) > \hat{\rho}_b(n-1) - \epsilon$, we make the change $w_{max} = w_{max} + 2$; otherwise, we make the changes $w_{max} = w_{max} - 1$ and $w_{min} = 1$.

More generally, at each stage we may divide the search interval for w in the ratio $1 : \sqrt{c}$ rather than in half as in (26), by setting

$$w_b(n) = w_{min} + \max\left(1, \lfloor \frac{w_{max} - w_{min}}{1 + \sqrt{c}} \rfloor\right). \tag{27}$$

In a static environment, Karp et al. [24, Theorems 1 and 2] showed that such an algorithm is near-optimal when at decision time n we incur a 'cost' of $w^* - w_b(n)$ when $w_b(n) \leq w^*$, representing an opportunity cost, and a cost of $c(w_b(n) - w^*)$ when $w_b(n) > w^*$. Here optimality is expressed in terms of either minimum worst-case total cost, or minimum expected total cost when w^* is uniformly distributed on the integers $[1, N]$. For the given algorithm (using (27)), the worst case cost for interval for $w^* \in [1, N]$ is $\sqrt{c}N + O(\log N)$, and the expected average cost (under the uniform distribution for w^* in $[1, N]$) is $(1/2)\sqrt{c}N + O(\log N)$; both are within $O(\log N)$ of the optimal cost. Our binary search procedure is equivalent to putting $c = 1$, when the cost of overestimating or underestimating w^* is the same, and is a useful reference point. Putting $c > 1$ would reflect the greater damage caused by overestimating w^*, since we start taking from other flows. An interesting issue concerns how the presence of the measurement noise terms Z_n in (23) affects such optimality properties.

4.2 The Stochastic Approximation Method

Stochastic approximation is a general technique for finding solutions to an equation $h(x) = 0$, given noisy measurements $h(x)+Z$, using

$$X_n = X_{n-1} + \gamma_n \theta_n,$$

where $\theta_n = h(X_{n-1}) + Z_n$ is the noisy observation of $h(X_{n-1})$. Such algorithms are also known as Robbins-Monro algorithms [25]. Conditions on the gain sequence γ_n, and on the noise variables Z_n, under which X_n converges almost surely to a desired solution are known. In particular, denoting by $\mathcal{F}_n$ the filtration of the random variables involved in the first n stages of the algorithm, under the conditions

$$(i) \sum_{n>0} \gamma_n = +\infty \text{ and } \sum_{n>0} \gamma_n^2 < +\infty, \tag{28}$$

$$(ii) \mathbf{E}(Z_{n+1}|\mathcal{F}_n) = 0 \text{ and } \mathbf{E}(Z_{n+1}^2|\mathcal{F}_n) \leq \sigma^2; n > 0 \tag{29}$$

for some constant σ^2, it is known that the iterations will converge to a solution (with probability 1) to a limit set of the ODE (ordinary differential equation)

$$\dot{x} = h(x). \tag{30}$$

When all solutions to the ODE (30) converge to a unique limiting point, then X_n converges with probability 1 to it.

The control problem at hand can be cast in this framework. Indeed, the equation (24) we are trying to solve is precisely of the form $h(w) = 0$, with $h(w) = \phi_b(w) - \rho_b + \epsilon$, and, ignoring for now the fact that ρ_b is unknown, we do have access to noisy estimates of $h(w_b(n))$, namely $\rho_b(n) - \rho_b + \epsilon$. This thus suggests to use updates of the form:

$$w_b(n) = w_b(n-1) + \gamma_n(\epsilon + \rho_b(n-1) - \hat{\rho}_b(n-2)). \tag{31}$$

In our implementation we also ensure $w_b(n) > 0$ to avoid getting infeasible solutions or getting stuck at $w = 0$.

Provided the estimates $\hat{\rho}_b(n)$ converge to ρ_b, we would expect the corresponding sequence $w_b(n)$ to converge to w^* under mild assumptions on the observation noises Z_n. However we have not pursued this yet, and have instead experimented with the fixed gain version of the update rule (31), where the γ_n are held constant to a common value γ. This is more appropriate in a dynamic environment where the target w^* itself might evolve over time, and is common in practice, where often $\gamma_n \downarrow \gamma_\infty$ for some $\gamma_\infty > 0$ to prevent the algorithm 'freezing' and enable tracking. We are then able to show stochastic convergence in the sense of Lyapunov - informally we remain close to an invariant distribution. The smaller γ, the slower the adaptation, but the smaller the divergence from the associated ODE (informally, the smaller variance).

To see this, first consider the noiseless case (for example, the situations of Theorems 1 and 2), and assume that ϕ_b defined in the previous section is a continuous function, and hence

$$\begin{cases} h(w) = \epsilon & 0 \leq w \leq w_b^* \\ 0 \leq h(w) < \epsilon & w_b^* < w \leq w^* \\ \epsilon - \rho < h(w) \leq 0 & w^* < w \end{cases} \tag{32}$$

where $h(w)$ is non-increasing for $w > w^*$ and $h(w) \to \epsilon - \rho$ as $w \to \infty$, the latter following from the assumed finite network capacity, hence there is some C^* such that $\phi_b(w) \leq C^*T/w$. Thus provided $\epsilon < \rho$ there is some w^* such that $h(w^*) = 0$ and the above characterisation of h holds. We then have

LEMMA 1. *In the case of noiseless observations where $\phi_b(w)$ is a continuous function defined as above, the recursion (31) for fixed γ, i.e. $w_b(n) = w_b(n-1) + \gamma h(w_{n-1})$ where $h(w) = \phi_b(w) - \rho_b + \epsilon$ is stable in the sense of Lyapunov, and all trajectories converge to the set of points $Q_\gamma(w^*)$ given by*

$$Q_\gamma = \{w : w^* - \frac{\gamma}{2}\epsilon \leq w \leq w^* + \frac{\gamma}{2}(\rho - \epsilon)'\}.$$

PROOF. Consider the function $V(w) = (w - w^*)^2$ for $w \in \mathbb{R}^+$, which is everywhere non-negative. It follows easily that $\triangle V(w) < 0$ for $w \notin Q_\gamma$, since

$$\begin{aligned} \triangle V(w_b(n)) =& V(w_b(n+1)) - V(w_b(n)) \\ =& 2\gamma(w_n - w^*)h(w_n) + \gamma^2 h(w_n)^2 \end{aligned}$$

and $h(w) \leq \epsilon$ for $w < w^*$ and $h(w) \geq \epsilon - \rho$ for $w > w^*$. □

Now consider the case of additive noise, described earlier. We can then appeal to [26, Proposition 1.2] to show

PROPOSITION 1. *Provided that the conditional second moment of Z_n is bounded by σ^2 – condition (29) (ii), then $V(w)$ is a stochastic Lyapunov function for the stochastic approximation for fixed γ, and the family of probability distributions of $w_b(n)$, $n > 0$, is tight.*

PROOF. With the given continuous function V, is suffices to show that there exists A such that

$$\lim_{w\to\infty} -(w-w^*)h(w) - \frac{1}{A}E[|h(w)+Z|^2] \to \infty$$

as $w \to \infty$, which is true for finite A, since $h(w)$ is bounded above and below and $E[Z^2]$ is bounded by σ^2. □

4.3 Dynamic stability for multiple background flows

When there are multiple competing background flows, window adaptation rules should be designed so that the interacting background flows have a stable dynamic behaviour. Relying on the work of Mo and Walrand [27], we consider differential equations describing the evolution of receiver windows, in the delay constrained case. These should be thought of as modeling the system dynamics under stochastic approximation, when the gain parameters are small, so that the effects of observation noise disappear and the ODE approximation is legitimate.

Specifically, consider the dynamics

$$\dot{w}_b = -\kappa_b s_b y_b u_b, \quad b \in \mathcal{B}, \tag{33}$$

where κ_b is a fixed positive gain parameter, and

$$\begin{array}{ll} s_b & = w_b - \tau_b y_b - v_b(y_b), \\ u_b & = \tau_b + v_b'(y_b), \\ v_b(y_b) & = \frac{\tau_b^2 y_b \epsilon_b(y_b)}{1-\tau_b\epsilon_b(y_b)}. \end{array} \tag{34}$$

We then have the following:

THEOREM 6. *Assume that at any instant t, the goodputs y_r are the solution of (4). Assume also that the functions v_b are such that*

$$0 < u_b < \tau_b, \quad y_b \in \mathbb{R}_+. \tag{35}$$

Then, under the dynamics (33), the goodputs y_r converge to the solution of (11,12,13). Moreover, the dynamics admit the Lyapunov function

$$V\left(\{w_b\}_{b\in\mathcal{B}}\right) := \frac{1}{2}\sum_{b\in\mathcal{B}} \kappa_b s_b^2. \tag{36}$$

PROOF. The proof is a direct adaptation of that of Theorem 6 of Mo and Walrand [27]. □

We note that Condition (35) is not met for functions ϵ_b as in (10). The question of proving stability of dynamics (33), or alternative dynamics with equilibrium points satisfying (9), in either delay or loss constrained situations, is still open.

4.4 Tradeoffs between the two approaches

In binary search, when the search interval is of length at most 1 and the receiver window size is too large, it is decreased by half. This is in the same spirit as the change in congestion window size in TCP, and ensures that binary search adapts quickly to the dynamic available capacity. However, in a stable environment, the receiver window size may still fluctuate, which leads to less efficient capacity utilization. The stochastic approximation algorithm can be tuned so that the receiver window size converges in a stable environment. However, since the increment and decrement in the receiver window size is linear (see (31)), it reacts more slowly to changes in the available capacity.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed schemes using *ns* simulation. In each simulation setting, we add a background transfer controlled by either one of our schemes, or TCP-LP [4]. Our objective is to investigate how the background transfer interferes with foreground traffic, and whether the background transfer utilizes the spare bandwidth efficiently. Each simulation run lasts for 1000 seconds. In each setting, we compare the performances of our schemes with TCP-LP. For our schemes, the data unit $u = 100$ bytes, $\epsilon = 1$ unit and $\epsilon' = 1$ unit. The EWMA parameter δ for ρ_n is set to 0.1, and the control interval T is 0.2 or 0.5 second. For stochastic approximation, the gain parameter $\gamma = 1$. For TCP-LP, we use parameters recommended in [4].

We use settings similar to those in [4]. First, we investigate a *baseline topology* with a single bottleneck. We then explore more complex settings with multiple bottlenecks. In the baseline topology, foreground and background traffic share a single bottleneck link, as shown in Figure 1. The capacity of the bottleneck link is 1.5 Mbps or 10 Mbps. The capacity of all the other links is 100 Mbps. The propagation delay of the bottleneck link is 20 ms. The propagation delay of all the other links is 2 ms. The round-trip propagation delay of a background transfer is therefore 44 ms, close to the median round-trip time of 50 ms between two sites on the same coast in the US [5]. Unless otherwise specified, the buffer size of the bottleneck link, B, is set to 2.5 times the bandwidth-delay product (BDP), corresponding to 0.15 Mb and 1.0 Mb for the link capacity of 1.5 Mbps and 10 Mbps respectively. The description of topology with multiple bottlenecks is deferred to later in this section. In each setting, the foreground traffic can be FTP flows (long-lived TCP), UDP flows or web traffic (short-lived TCP). We now describe the simulation results in detail.

5.1 Interaction with FTP traffic

We first consider two simultaneous FTP transfers, one being a foreground flow using TCP and the other being a background transfer using binary search, stochastic approximation or TCP-LP. The bandwidth of the bottleneck link is 1.5 Mbps. The packet size in the TCP flow is 1000 bytes. With no constraint from the receiver window and traffic on the reverse path, the TCP flow fully utilizes the bottleneck capacity. After adding a background transfer, the throughput of the TCP flow is only slightly reduced, as shown in the first row of Table 1. This indicates that the perturbation imposed by the background transfer is negligible. The performance when using TCP-LP, binary search and stochastic approximation for the background transfer is very close. For binary search and stochastic approximation, the background transfer uses slightly more bandwidth for shorter control interval (e.g., $T = 0.2$ second) than for longer control interval (e.g., $T = 0.5$ second). We now add 10 FTP flows in the reverse direction, which causes packet losses in the reverse path. The throughput of the TCP flow is then reduced to 48% of the bottleneck capacity. In the presence of background transfer, the throughput of the TCP flow is again only slightly perturbed, as shown in the second row of Table 1.

We now examine one scenario where the throughput of the TCP flow is restricted by the receiver window size. In this case, the TCP flow does not fully utilize the bottleneck capacity and some spare bandwidth is available to the background transfer. The capacity of the bottleneck link is 10 Mbps. In the absence of background transfer, the TCP flow uses 57% of the bottleneck capacity. When the background flow is added, the utilisation of the bandwidth is raised to 97% (binary search) or 100% (stochastic approximation) for control interval $T = 0.5$ second. The corresponding reductions of foreground throughput are only 2% or 6% respectively. This

Scenario	TCP	TCP v.s. TCP-LP	TCP v.s. binary search		TCP v.s. stoch. approx.	
			$T = 0.5$ sec	$T = 0.2$ sec	$T = 0.5$ sec	$T = 0.2$ sec
no reverse traffic	100	97.1 v.s. 2.9	98.7 v.s. 1.3	97.8 v.s. 2.3	98.7 v.s. 1.1	98.0 v.s. 2.1
reverse traffic	48	47.2 v.s. 1.7	46.0 v.s. 1.3	46.0 v.s. 1.4	47.7 v.s. 0.6	46.1 v.s. 2.2

Table 1: Normalized throughput when a FTP flow and a background transfer share a single bottleneck link.

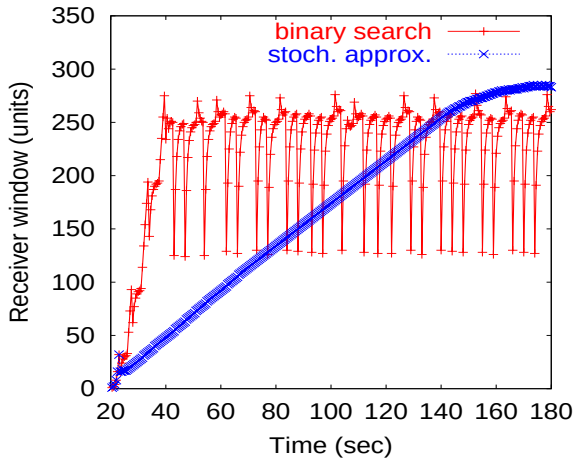

Figure 5: Receiver window of the background transfer evolves with time, $T = 0.5$ second.

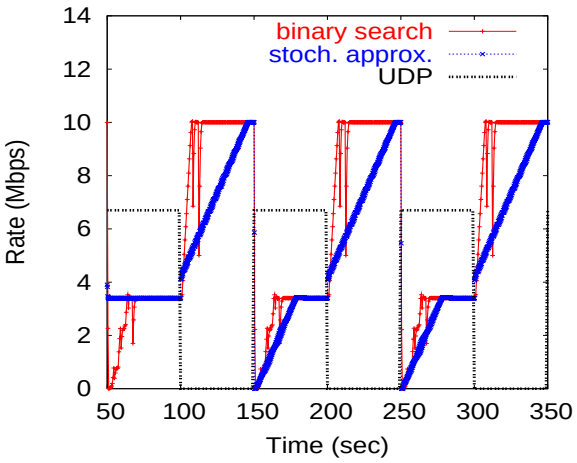

Figure 7: Rates of the background transfer and the UDP flow (on/off duration of 50 seconds) versus time, $T = 0.2$ second.

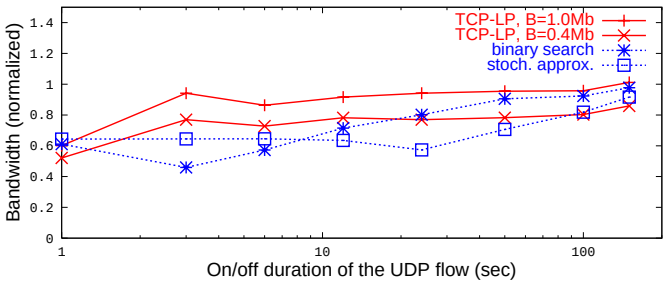

Figure 6: Average bandwidth usage of the background transfer versus the on/off duration of the square wave UDP flow, $T = 0.2$ second.

demonstrates that the background transfer utilizes the spare bandwidth while causing only slight perturbations to the foreground traffic. Figure 5 plots the evolution of the receiver window size of the background transfer with time. We observe that when using binary search, the receiver window reaches 275 units in less than 20 seconds and then fluctuates between 130 and 275 units. When using stochastic approximation, the receiver window exhibits a linear increase with time and then stabilizes at a window size close to 275 units. This reflects the trade-off between the speed of reaction and variability of the two approaches discussed in the previous section.

5.2 Interaction with square-wave UDP traffic

We now investigate the interaction of background transfer with an on/off UDP flow. The capacity of the bottleneck link is 10 Mbps. The buffer size of the bottleneck link, B, is 1.0 Mb or 0.4 Mb. The rate of the UDP flow is 6.7 Mbps and 0 Mbps during the on and off period respectively. The durations of the on and off period are the same, forming a "square wave" of bandwidth usage. Therefore, the average available bandwidth is $(3.3 + 10)/2 \approx 6.7$ Mbps. We vary the on/off duration of the UDP flow from 1 second to 150 seconds. For each setting, we obtain the average bandwidth usage of the background transfer and normalize it by the average available bandwidth. Figure 6 shows the normalized bandwidth usage of the background transfer versus the duration of the on/off period of the square-wave UDP flow. In general, the background transfer utilizes more bandwidth as the on/off duration of the UDP flow increases. When using binary search or stochastic approximation, the bandwidth usage of the background transfer is insensitive to the buffer size. When using TCP-LP, the bandwidth usage for larger bottleneck buffer size (e.g., $B = 1$ Mb) is higher than for smaller buffer size (e.g., $B = 0.4$ Mb). For short on/off duration, our schemes utilize less available bandwidth than TCP-LP, due to the fact that our schemes respond at a larger time scale than the round trip time in TCP-LP. However, as the on/off duration increases, the bandwidth usage under our schemes approaches that under TCP-LP. For short on/off durations, the bandwidth usage with stochastic approximation is higher than that with binary search. This is because the receiver window size fluctuation in binary search affects its bandwidth usage. For long on/off durations, the bandwidth usage by binary search is higher because of its fast responsiveness.

Figure 7 shows the rates of the UDP flow and background transfer as a function of time. The background transfer is controlled by binary search or stochastic approximation using $T = 0.2$ second. The UDP on/off duration is 50 seconds. When using binary search, we observe that the background flow reacts well to changes in the available bandwidth and is indeed able to use around 90% of what is available. When using stochastic approximation, the reaction of the background flow is relatively slower, yet still keeping a good track of the dynamics of the available bandwidth (using 70% of the available bandwidth).

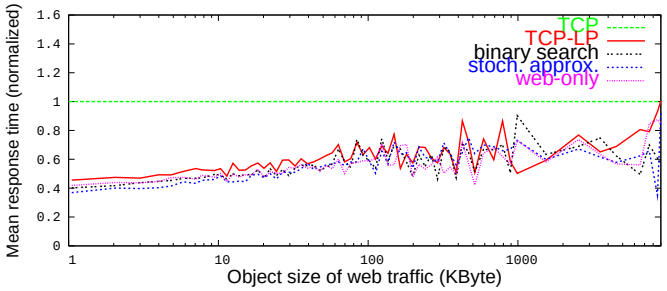

Figure 8: Normalized mean response time for web objects in the baseline topology, $T = 0.2$ second.

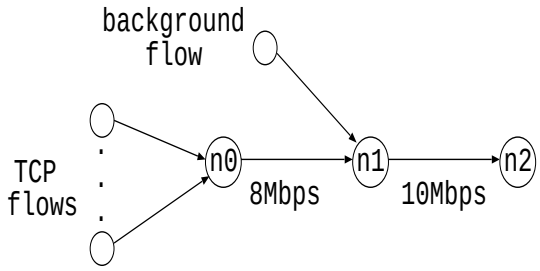

Figure 9: A group of TCP flows share link $(n1, n2)$ with the background flow.

Buffer size (Mb)	TCP v.s. TCP-LP (Mbps)	TCP v.s. binary search (Mbps)	
		$T = 0.5$ sec	$T = 0.2$ sec
1.0	8.0 v.s. 2.0	8.0 v.s. 1.9	7.9 v.s. 1.9
0.4	7.7 v.s. 1.7	8.0 v.s. 1.9	7.9 v.s. 1.9

Table 2: Throughput when the aggregate throughput of the foreground flows is constrained to be no more than 8 Mbps.

5.3 Interaction with web traffic

We next examine a realistic scenario where web traffic is dominant. The web traffic generated using the same model as in [4]. That is, each web session contains several web pages, and each web page contains 10 objects. The inter-page and inter-object time distributions are exponential with means of 1 sec and 1 msec respectively. The object size is distributed according to a Pareto distribution with shape parameter of 1.2. The capacity of the bottleneck is 1.5 Mbps. In the same direction as the web traffic, we set up one FTP connection for bulk data transfer, which uses TCP, TCP-LP, binary search or stochastic approximation. We measure the impact of the bulk data transfer on the response time of the web objects. We conduct 30 runs and average the response time for objects of different sizes over all the runs.

Figure 8 plots the average response times of the web objects normalized by the response times when using TCP for the bulk data transfer. In the figure, the control interval length is $T = 0.2$ second. The normalized response times of the web objects when using TCP-LP and our schemes are close to those with no bulk data transfer (marked as "web-only" in the figure) and much lower than 1 for most object sizes, indicating that they are much less intrusive than TCP. Furthermore, the impact on the web traffic when using our schemes is comparable to when using TCP-LP. The average throughput of the bulk data transfer is 0.69 Mbps and 0.53 Mbps when using TCP and TCP-LP respectively. When using binary search, the average throughput of the bulk data transfer is 0.46 Mbps, slightly lower than when using TCP-LP. When using stochastic approximation, the average throughput of the bulk data transfer is 0.35 Mbps, with correspondingly the least impact on the responsiveness of the web traffic as shown in Figure 8. Finally, we observe similar behavior for $T = 0.5$ second.

5.4 Multiple bottleneck links

We next explore the performance of our schemes in networks with multiple bottlenecks. The objective is to examine whether a more complex topology or the existence of multiple bottlenecks affects the performance of our schemes.

5.4.1 *Aggregate TCP throughput constrained*

We first look at a topology where a background transfer interacts with multiple TCP flows and the aggregate throughput of the TCP flows is constrained, as shown in Figure 9. 10 TCP flows traverse links $(n0, n1)$ and $(n1, n2)$. The capacity of link $(n0, n1)$ and $(n1, n2)$ is 8 Mbps and 10 Mbps respectively. The aggregate throughput of the TCP flows is therefore no more than 8 Mbps, limited by the minimum capacity of the two links. The buffer size of link $(n0, n1)$ and $(n1, n2)$ is the same, set to 1.0 Mb or 0.4 Mb. The background transfer shares link $(n1, n2)$ with the multiple TCP flows. Table 2 lists the aggregate TCP throughput versus the throughput of the background transfer. The results when using stochastic approximation are similar to those when using binary search and hence are not listed in the table. When using our schemes, the TCP flows and the background transfer achieve the desired bandwidth share, regardless of the buffer size and the length of the control interval. For TCP-LP, the performance when using a smaller buffer (e.g., 0.4 Mb) is slightly worse than using a larger buffer (e.g., 1.0 Mb).

5.4.2 *Multi-hop background traffic*

We now examine the behavior of the background transfer in a multiple-hop network as shown in Figure 10. In the figure, links $(n0, n1)$, $(n1, n2)$ and $(n2, n3)$ are identical and form three consecutive bottleneck links. The capacity, propagation delay and buffer size of a bottleneck link are 1.5 Mbps, 10 ms and 0.15 Mb respectively. Three groups of web traffic each crosses a bottleneck link. A bulk data transfer traverses all the bottleneck links along the direction of web traffic. In this case, the round trip time of the bulk data transfer is approximately 3 times longer than that of the web traffic.

Figure 11 shows the normalized mean response time of different object sizes. We observe that the impact of TCP-LP on the response time of most object sizes is larger than binary search and stochastic approximation, all three schemes less intrusive than TCP. Correspondingly, the average throughput of the bulk data transfer when using TCP-LP (0.08 Mbps) is slightly higher than when using binary search (0.05 Mbps) and stochastic approximation (0.04 Mbps).

5.4.3 *Multi-hop web traffic*

We next consider a scenario where web traffic traverses multiple hops as shown in Figure 12. The topology and the configuration of the links are the same as in Section 5.4.2. In addition to web traffic, three FTP sessions each traverse a single hop. The round trip time of the web traffic is approximately three times higher than that of the bulk data transfer. The background transfer is hence more likely to affect the responsiveness of the web traffic in this case.

Figure 13 shows the normalized response time of different object sizes. Again we observe that, when using TCP-LP for bulk

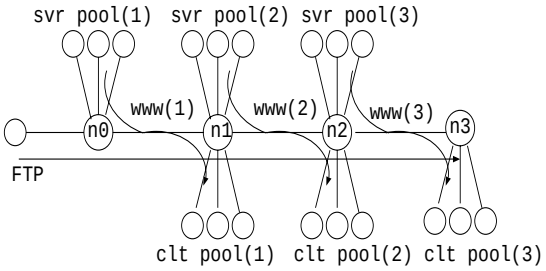

Figure 10: A bulk data transfer (using TCP, TCP-LP, binary search or stochastic approximation) traverses three consecutive bottlenecks.

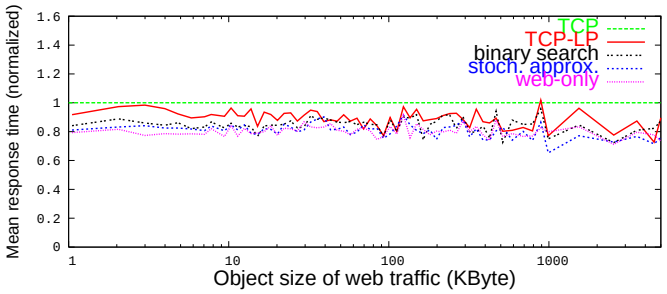

Figure 11: Normalized mean response time for file downloads in a multi-hop topology where a bulk data transfer traverses multiple bottleneck links, $T = 0.2$ second.

data transfer, the response times of most object sizes are slightly higher than when using binary search or stochastic approximation, all three schemes much less intrusive than TCP. The average throughput of the bulk data transfer is 2.04 Mbps, 1.56 Mbps and 1.41 Mbps when using TCP-LP, binary search and stochastic approximation respectively.

6. CONCLUDING REMARKS

We have looked at the problem of creating a background transfer service using an application layer reaction, adapting a receiver window to create a low-priority service. By phrasing the idealized problem as an optimization problem, we were able to characterize the solution as a dual control problem, where the aim is to control the window to a critical value which represents a change point. It is necessary to relax the ideal operating point to allow for measurement noise, and to enable us to address the problem of fairness between competing background flows. As a by-product, the relaxed problem makes the problem of searching for the optimal window easier. However, the relaxation implies some degradation for foreground flows. We have characterized this relaxation by a function $\epsilon(\cdot)$ of the measured throughput, and explored a particular parameterized form of this function. There is a tension between simplicity of the chosen parameters, and fairness between flows. The simplest parameterization just involves setting one parameter to a fraction, which in the case of delay-constrained networks corresponds to the fractional degradation that foreground flows see. However, such a choice of parameters does not guarantee fairness between flows.

We have then given two practical ways of attacking this control problem, using binary search or stochastic optimization techniques. The two-techniques have complementary strengths: binary search is quick to find a good operating point, but has inherent fluctuations, whereas stochastic optimization can be tuned for stability at the

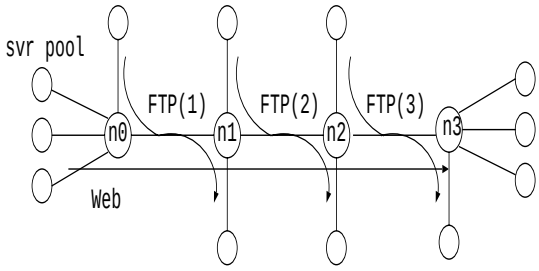

Figure 12: The web traffic traverses three consecutive bottlenecks.

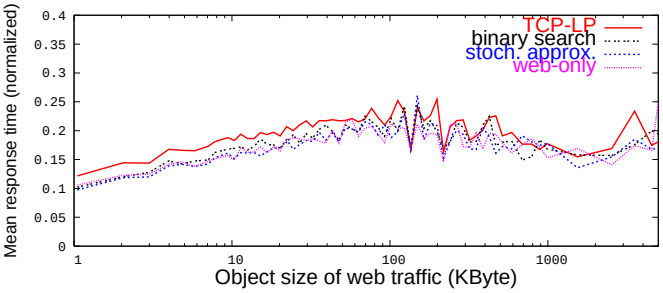

Figure 13: Normalized mean response time for file downloads in a multi-hop topology where web traffic traverses multiple bottleneck links, $T = 0.2$ second.

expense of convergence speed. It may be possible to exploit the strengths of both approaches by combining them.

Simulations have given credence to our framework and approach, and initial results on the algorithms are encouraging, especially since we have not sought to fine-tune parameters. Indeed, the results suggest that such an application level reaction is able to work almost as well as a transport layer reaction, such as TCP-LP.

7. ACKNOWLEDGEMENT

We thank A. Kuzmanovic for many helps of using TCP-LP. We also thank the anonymous reviewers for their insightful comments. The last part of Bing Wang's work was supported in part by the National Science Foundation under NSF ANI-9980552 and EEC-0313747 001.

8. REFERENCES

[1] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin. The potential costs and benefits of long term prefetching for content distribution. *Computer Communication Journal*, 25(4):367–375, 2002.

[2] Antony I. T. Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Symposium on Operating Systems Principles*, pages 188–201, 2001.

[3] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A mechanism for background transfers. In *Proc. Operating Systems Design and Implementation*, December 2002.

[4] A. Kuzmanovic and E. Knightly. TCP-LP: A distributed algorithm for low priority data transfer. In *Proc. IEEE INFOCOM*, 2003.

[5] B. Huffaker, M. Fomenkov, D. Moore, and K. Claffy. Macroscopic analyses of the infrastructure: Measurement and visualization of Internet connectivity and performance.

In *A Workshop on passive and active measurements*, April, 2001.

[6] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, and R. G. Baraniuk. Multifractal cross-traffic estimation. In *Proc. ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 2000.

[7] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proc. IEEE GLOBECOM*, November 2000.

[8] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *Proc. ACM SIGCOMM*, 2002.

[9] Neil T. Spring, Maureen Chesire, Mark Berryman, Vivek Sahasranaman, Thomas Anderson, and Brian N. Bershad. Receiver based management of low bandwidth access links. In *Proc. IEEE INFOCOM*, volume 1, pages 245–254, 2000.

[10] Puneet Mehra, Avideh Zakhor, and Christophe De Vleeschouwer. Receiver-driven bandwidth sharing for TCP. In *Proc. IEEE INFOCOM*, 2003.

[11] M. Crovella and P. Barford. The network effects of prefetching. In *Proc. IEEE INFOCOM*, 1998.

[12] Jon Crowcroft and Philippe Oechslin. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. *ACM Computer Communication Review*, 28(3), July 1998.

[13] Y. Dong, R. Rohit, and Z. Zhang. A practical technique to support controlled quality assurance in video streaming across the Internet. In *Proc. Packet Video Workshop*, 2002.

[14] Jeffrey Semke, Jamshid Mahdavi, and Matthew Mathis. Automatic TCP buffer tuning. In *Proc. ACM SIGCOMM*, pages 315–323, 1998.

[15] Manish Jain, Ravi Prasad, and Constantinos Dovrolis. The TCP bandwidth-delay product revisited: network buffering, cross traffic, and socket buffer auto-sizing. Technical Report GIT-CERCS-03-02, College of Computing, Georgia Tech, 2003.

[16] Harold J. Kushner and George Yin. *Stochastic Approximation Algorithms and Applications*. Springer Verlag, 1997.

[17] Laurent Massoulie and James Roberts. Bandwidth sharing: Objectives and algorithms. In *Proc. IEEE INFOCOM*, volume 3, pages 1395–1403, 1999.

[18] T.R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[19] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: Utility functions, random losses and ECN marks. In *INFOCOM 2000*, 2000.

[20] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999. http://www.statslab.cam.ac.uk/∼frank/PAPERS/evol.html.

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

[22] F. P. Kelly. Mathematical modelling of the Internet. In *Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, 2000.

[23] S. McCanne and S. Floyd. ns-LBNL network simulator. http://www-nrg.ee.lbl.gov/ns/.

[24] Richard M. Karp, Elias Koutsoupias, Christos H. Papadimitriou, and Scott Shenker. Optimization problems in congestion control. In *IEEE Symposium on Foundations of Computer Science*, pages 66–74, 2000.

[25] H. Robbins and S Munro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[26] Jean-Claude Fort and Gilles Pages. Asymptotic behavior of a Markovian stochastic algorithm with constant step. *SIAM J. Control*, 37(5):1456–1482, 1999.

[27] Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, 2000.