INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Comparison of TCP Reno and TCP Vegas via Fluid Approximation

Thomas Bonald

N° 3563

Novembre 1998

THÈME 1

Rapport de recherche

ISSN 0249-6399

Comparison of TCP Reno and TCP Vegas via Fluid Approximation

Thomas Bonald *

Thème 1 — Réseaux et systèmes
Projet Mistral

Rapport de recherche n° 3563 — Novembre 1998 — 34 pages

Abstract: We compare the efficiency of the flow control of two versions of TCP, the transmission control protocol of the Internet: the current version called Reno, and a recently proposed version called Vegas. By means of a fluid approximation, we show that due to the use of round-trip times measurement, the window dynamics of TCP Vegas are much more stable than those of TCP Reno, resulting in a much more efficient utilization of the network resources. In addition, whereas TCP Reno discriminates against users with long propagation delays, TCP Vegas fairly shares the available bandwidth between the users, whatever their propagation delays.

Key-words: Internet, TCP, window flow control, fluid approximation, stability, performance evaluation, fairness.

* France Télécom CNET and INRIA - Email: tbonald@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Comparaison de TCP Reno et TCP Vegas par Approximation Fluide

Résumé : Nous comparons l'efficacité du contrôle de flux de deux versions de TCP, le protocole de contrôle de transmission de l'Internet : la version actuelle appelée Reno, et une version récemment proposée appelée Vegas. Par une approximation fluide, nous montrons que grâce à l'utilisation des mesures de temps de transmission des paquets, la dynamique de la fenêtre de TCP Vegas est beaucoup plus stable que celle de TCP Reno, et conduit à une utilisation bien plus efficace des ressources du réseau. De plus, alors que TCP Reno défavorise les utilisateurs ayant de long délais de propagation, TCP Vegas partage la bande passante équitablement entre les utilisateurs, quelque soient leurs délais de propagation.

Mots-clés : Internet, TCP, contrôle de flux à fenêtre, approximation fluide, stabilité, évaluation de performance, équité.

1 Introduction

Most of today's Internet traffic is generated by traditional data transfer applications such as HTTP, NNTP or FTP connections[1]. These applications which are usually not sensitive to the delivery time of each individual packet, but rather to the total transfer time of the data, are often referred to as *elastic* applications, as opposed to *real-time* applications. They use largely TCP, which provides end-to-end control over the "best-effort" service of IP. The role of TCP, apart from ensuring a reliable delivery of the packets, is to adapt the sending rate of the source to the capacity of the network. Without flow control, the source could indeed saturate one or several routers, which would cause many losses and retransmissions, resulting in a low "goodput". Thus TCP must face the trade-off of achieving a high utilization of the network resources while keeping the amount of data in the buffers as low as possible.

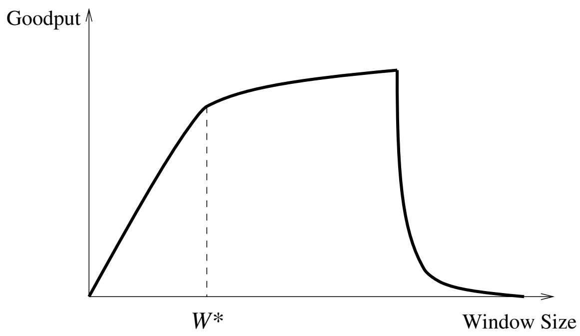

Figure 1: Typical "goodput" curve

The flow control of TCP is based on a *window* mechanism, which consists in limiting the number of packets sent by the source and not yet acknowledged by the destination. The efficiency of this mechanism depends greatly on the choice of the window size, as shown in Figure 1. Unfortunately, due to the heterogeneity of the Internet, the "good" value W^* of the window size is not known *a priori*. Moreover, it depends on dynamic parameters of the network, such as the number of connections sharing the same link. For these reasons, the window mechanism of TCP is *adaptive*. The window size W is initially set to 1 and then evolves according to the following algorithm [4], where ACK denotes the reception of an acknowledgment:

[1] See Appendix A for the main abbreviations used in the paper

Additive Increase

```
every ACK do
  if W < W_t then W := W + 1
  else W := W + 1/W
```

In the first phase called the *slow-start* phase, the window size is doubled each time W acknowledgments have been received, that is every round-trip time (RTT), resulting in an exponential increase. In the second phase called the *congestion avoidance* phase, the window size is increased by one every RTT, resulting in a linear increase. The window threshold W_t indicates when switching from one phase to the other. When a loss is detected by the expiry of a time-out (TO), the values of W and W_t are changed as follows:

Multiplicative Decrease

```
every TO do
  W_t := γ W
  W := 1
```

where γ is a fixed parameter such that $0 < \gamma < 1$, typically set to 1/2. Thus the window mechanism of TCP consists roughly of cycles where the window size is initially set to 1 and increased constantly until a loss occurs[2]. As a result, the current version of TCP called *Reno* oscillates between periods of "under-utilization" and periods of "over-utilization" of the network resources.

In order to avoid such a periodic behavior, a new version of TCP called *Vegas* was proposed in [3]. The main idea is to use the RTT measurement to stabilize the window size close to the "good" value. More precisely, the source computes the minimum of all measured round-trip times $\mathrm{RTT}_{\min}$, and evaluates the difference

$$\mathrm{Diff} = \lambda_{\mathrm{exp}} - \lambda_{\mathrm{act}},$$

where λ_{exp} is the *expected* throughput $W/\mathrm{RTT}_{\min}$ and λ_{act} is the *actual* throughput W/RTT. Defining two fixed parameters α and β, typically set to 1 and 3 or 2 and 4, the congestion avoidance phase is then changed as follows:

[2]In fact, the window size is also limited by the receiver's advertised window, typically set to 64 K-bytes, or more with the window scale option [5]. In the following, we assume that this parameter does *not* constrain the increase of the window.

Additive Increase-Decrease

$$
\begin{array}{llll}
\texttt{every} \;\; \mathrm{RTT} \;\; \texttt{do} & & & \\
\quad \texttt{if} & \mathrm{Diff} < \alpha/\mathrm{RTT}_{\min} & \texttt{then} & W := W + 1 \\
\quad \texttt{else if} & \mathrm{Diff} > \beta/\mathrm{RTT}_{\min} & \texttt{then} & W := W - 1
\end{array}
$$

Other modifications proposed in [3], which concern the retransmission mechanism or the slow-start phase, are *not* considered in this paper. However, we will refer to this version of TCP where only the congestion avoidance phase is modified as TCP Vegas for convenience. In particular, both versions considered in the following have the same behavior as far as the slow-start phase and the loss detection and retransmission mechanisms are concerned. That is the reason why in the rest of the paper, we only focus on the *steady* behavior of TCP Reno and TCP Vegas, where the congestion avoidance phase plays a crucial role.

The goal of this paper is to compare the efficiency of the *flow control* of TCP Reno and TCP Vegas, in terms of utilization of the network resources. As noted above, we are not interested in *error control* mechanisms, since both protocols do not differ at this stage. In particular, we only model losses which have an impact on the flow control, namely those which are due to buffer overflow, and cause the expiry of a time-out and the window reduction according to the multiplicative decrease algorithm.

The model and the fluid approximation used for the analysis are described in next section. In Section 3, we compare the stable behavior of TCP Reno and TCP Vegas by studying the steady state of a fixed number of connections sharing the same link. Using this, we then compare in Section 4 the performance of TCP Reno and TCP Vegas, both in terms of utilization of the available bandwidth and buffer occupation, when the number of connections is dynamic, namely when the starting times of the connections and the size of the transferred files are random. Finally, we consider in Section 5 the problem of the fair sharing of the available bandwidth, when the connections have different propagation delays. Section 6 concludes the paper.

2 Model

2.1 Network dynamics

Consider a single connection going through N FIFO routers, and controlled by a window of fixed size W. Assuming that the source can always use the transmission capacity allowed by the flow control (which is indeed the case for most connections, which consist of the transfer of a single file), and representing the cross traffic at each router by an exogenous flow, the model corresponds the open-closed queueing network of Figure 2.

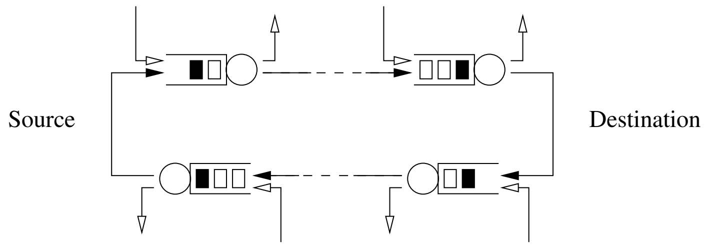

Figure 2: Discrete model with cross traffic

It turns out that this model, which was studied in [8, 9] in the particular case of exponential service times and Poisson arrival processes, is untractable in the more realistic situation where the service times are deterministic and the arrival processes generally distributed. In particular, it was shown in [2] that the throughput λ of the controlled connection depends in a crucial way on the statistical characteristics of the cross traffic. However, it is possible to give tight bounds on its value. Denote by μ the *available bandwidth* of the controlled connection, defined by

$$\mu = \min_{1 \leq i \leq N} \mu_i(1-\rho_i), \tag{1}$$

where μ_i is the service rate at queue i and ρ_i the traffic intensity of the cross flow at that queue, and let τ be the *propagation delay* of the packets of the controlled connection (that is the round-trip time minus the queueing delays). Then the following inequality always holds true:

$$\lambda \leq \min\left(\mu, \frac{W}{\tau}\right).$$

In addition, this bound is tight when the size of the packets is *small* (see [2]), that is when the transmission times $\mu_1^{-1}, \ldots, \mu_N^{-1}$ are small compared to the propagation delay τ, which is indeed the case in current high-speed communication networks. Thus in the fluid approximation, we have

$$\lambda = \min\left(\mu, \frac{W}{\tau}\right).$$

As shown in Figure 3, the model reduces then to a *single bottleneck*, namely the queue i which reaches the minimum in the right-hand side of (1). Denoting by B the *available* buffer size at this node (that is $B = B_i(1-\rho_i)$), where B_i is the buffer size of router i), the maximum allowed window size (without buffer overflow) is given by

$$W_{\max} = \mu\tau + B.$$

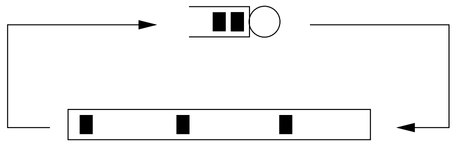

Figure 3: Fluid approximation

The resulting throughput curve is shown in Figure 4. This curve is an idealized version of that of Figure 1. It is clear that the optimal window size is the *bandwidth-delay* product $W^* = \mu\tau$. In this case, the controlled connection uses all the available bandwidth and no data packet is buffered in the network.

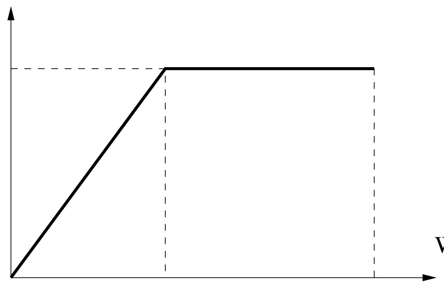

Figure 4: Throughput curve

In the following, the window size of the connection is not static but varies according to the algorithms of TCP Reno or TCP Vegas. Though the window size is now a function of time, we assume that the network dynamics are sufficiently fast compared to the window dynamics, so that the system is always in the steady state described above. In particular, the throughput of the TCP connection is still given at any time t by

$$\lambda(t) = \min\left(\mu, \frac{W(t)}{\tau}\right), \tag{2}$$

and from Little's law, we get at any time t,

$$\mathrm{RTT}(t) = \frac{W(t)}{\lambda(t)} = \max\left(\frac{W(t)}{\mu}, \tau\right). \tag{3}$$

2.2 Window dynamics

Since the slow-start phase is very short compared to the congestion avoidance phase (due to the exponential increase of the window), it can be neglected in the evaluation of long-range performance criteria we are interested in. In the congestion avoidance phase, the window size increases (or decreases) linearly at rate 1/RTT. Provided the window size is sufficiently large, it may be modeled by a continuous function of time. This leads to the following equations for the window evolution of TCP Reno:

TCP Reno

$$\begin{cases} \dfrac{dW}{dt} = \dfrac{1}{\mathrm{RTT}(t)} \\ \\ W(t) = W_{\max} \implies W(t^+) = \gamma\, W(t). \end{cases} \tag{4}$$

Concerning TCP Vegas, since the first packet of the connection experiments no queueing delay, the minimum of all measured round-trip times $\mathrm{RTT}_{\min}$ is reached immediately and is equal to the propagation delay τ. This leads to the following equations for the window evolution of TCP Vegas:

TCP Vegas

$$\begin{cases} \dfrac{dW}{dt} = \dfrac{\varepsilon(t)}{\mathrm{RTT}(t)} \\ \\ W(t) = W_{\max} \implies W(t^+) = \gamma\, W(t). \end{cases} \tag{5}$$

Here, $\varepsilon(t)$ is given by

$$\varepsilon(t) = -\frac{1}{2}\left(\mathrm{sgn}(X(t)-\alpha) + \mathrm{sgn}(X(t)-\beta)\right), \tag{6}$$

where $\mathrm{sgn}(x) = 1$ if $x > 0$ and $\mathrm{sgn}(x) = -1$ if $x < 0$ (the value of $\varepsilon(t)$ for the critical values α and β of $X(t)$ belongs respectively to the intervals $[0,1]$ and $[-1,0]$), and $X(t)$ represents the buffer occupation, performed by the source thanks to the RTT measurement, namely

$$X(t) = \mathrm{Diff(t)} \times \tau = \left(\frac{W(t)}{\tau} - \frac{W(t)}{\mathrm{RTT}(t)}\right)\tau = W(t) - \lambda(t)\tau. \tag{7}$$

Hence, the window algorithm of TCP Vegas aims at stabilizing the window size so as to have between α and β packets buffered in the network, that is to a value very close to (and above) the optimal value W^* (see Figure 5 below). In the next section, we describe in details this stable behavior in the case of multiple connections.

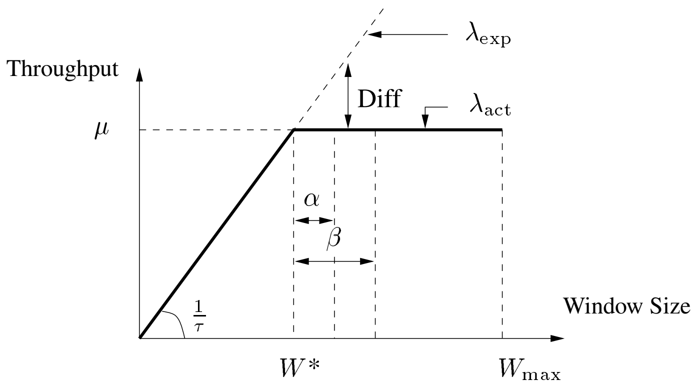

Figure 5: Window stabilization of TCP Vegas

3 Stability

In this section, we consider a fixed number of K connections which use the same version of TCP, either Reno or Vegas, to transfer a single (large) file through the same link. These connections are assumed to be *homogeneous*, that is to experiment the same propagation delay τ (the case of heterogeneous connections is investigated in Section 5). In addition, the connections are assumed to be *synchronized*, in the sense that in case of buffer overflow, all connections detect a loss and multiply their window size by γ, according to the multiplicative decrease algorithm described in Section 1. This synchronization phenomenon was reported for instance in [7, 11].

We index these connections by $k = 1, \ldots, K$, and denote by $W_k(t)$ the window size of connection k at time t. We assume that each connection k started at some negative time t_k, and we focus on the evolution of the window sizes $W_1(t), \ldots, W_K(t)$ for non-negative times t. For any $t \geq 0$, we denote by $W(t) = W_1(t) + \ldots + W_K(t)$ the total window size at time t, and by $\lambda(t)$ and $X(t)$ respectively the total throughput and the total buffer occupation at time t.

3.1 TCP Reno

From the FIFO assumption, each of the K TCP Reno connections experiments the same RTT. In view of (4), the window sizes of two connections k, l, have the same dynamics:

$$\begin{cases} \dfrac{dW_k}{dt} = \dfrac{dW_l}{dt} \\ W(t) = W_{\max} \implies W_k(t^+) = \gamma\, W_k(t) \text{ and } W_l(t^+) = \gamma\, W_l(t). \end{cases} \tag{8}$$

In addition, it follows from (3) and (4) that

$$\begin{cases} \dfrac{dW}{dt} = \dfrac{K}{\tau} & \text{when} \quad W(t) \le \mu\tau, \\ \dfrac{dW}{dt} = K\, \dfrac{\mu}{W(t)} & \text{when} \quad \mu\tau < W(t) < W_{\max}, \\ W(t^+) = \gamma\, W(t) & \text{when} \quad W(t) = W_{\max}. \end{cases} \tag{9}$$

Let t_0 be the first positive time when a loss occurs. As we shall see, losses occur then at times $t_0 + nT$, where T is the period of the unique solution $\mathcal{W}(t)$ of (9), such that $\mathcal{W}(0) = W_{\max}$. More precisely, we have

$$\forall t \ge 0, \quad W(t_0 + t) = \mathcal{W}(t).$$

In addition, it follows from (8) that for any connection k,

$$\forall t \ge 0, \quad W_k(t) = \gamma^{\lceil \frac{t-t_0}{T} \rceil} \left(W_k(0) - \frac{W(0)}{K} \right) + \frac{W(t)}{K}.$$

Thus the dynamics are completely determined by the periodic function $\mathcal{W}(t)$. The difference between the window sizes tends to zero when t tends to infinity, so that in the steady state, the throughput of each connection is a fraction $1/K$ of the total throughput λ.

We will distinguish between two cases, depending on the value of $\gamma W_{\max}$ with respect to $\mu\tau$. We define

$$\omega = \frac{\mu\tau}{B},$$

the ratio between the bandwidth-delay product and the buffer size at the bottleneck node (ω^{-1} is the *normalized buffer size*, as defined in [7]). Note that

$$\gamma W_{\max} \le \mu\tau \quad \iff \quad \omega \ge \frac{\gamma}{1-\gamma}.$$

[2]For any $x \in \mathbb{R}$, $\lceil x \rceil$ denotes the only integer belonging to $[x, x+1)$.

In both cases, we evaluate the average throughput and the average buffer occupation, respectively defined by

$$\lambda = \frac{1}{T}\int_0^T \lambda(t)\, dt \quad \text{and} \quad X = \frac{1}{T}\int_0^T X(t)\, dt.$$

Case $\gamma W_{\max} \leq \mu\tau$. The cycles consist of two phases of lengths T_1 and T_2. During the first phase, we have

$$\forall 0 < t \leq T_1, \quad \mathcal{W}(t) = \frac{Kt}{\tau} + \gamma W_{\max},$$

and this phase ends when $\mathcal{W}(t) = \mu\tau$. During the second phase, we have

$$\mathcal{W}(t)\frac{d\mathcal{W}}{dt} = K\mu,$$

so that

$$\forall 0 \leq t \leq T_2, \quad \mathcal{W}(t + T_1) = \sqrt{(\mu\tau)^2 + 2K\mu t},$$

and this phase ends when $\mathcal{W}(t) = W_{\max}$. The total duration of a cycle is $T = T_1 + T_2$. From (2), the average throughput is given by

$$\lambda = \frac{1}{T}\left(\int_0^{T_1} \frac{\mathcal{W}(t)}{\tau} dt + \int_{T_1}^{T_1+T_2} \mu\, dt\right),$$

and the average occupation of the buffer by

$$X = \frac{1}{T}\int_{T_1}^{T_1+T_2} (\mathcal{W}(t) - \mu\tau) dt.$$

After simple calculations, we obtain

$$\lambda = \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1}\,\mu \quad \text{and} \quad X = \frac{\omega + 2/3}{2(1-\gamma)(\omega^2+\omega)+1}\,B. \tag{10}$$

Case $\gamma W_{\max} \geq \mu\tau$. The cycles consist of a single phase, namely

$$\forall 0 < t \leq T, \quad \mathcal{W}(t) = \sqrt{(\gamma W_{\max})^2 + 2K\mu t}.$$

We have

$$\lambda = \frac{1}{T}\int_0^T \mu\, dt \quad \text{and} \quad X = \frac{1}{T}\int_0^T (\mathcal{W}(t) - \mu\tau) dt,$$

that is

$$\lambda = \mu \quad \text{and} \quad X = \left(\frac{2(1+\gamma+\gamma^2)}{3(1+\gamma)}(\omega+1) - \omega\right) B. \tag{11}$$

Remark (Insensitivity). In both cases, the (total) average throughput and the average buffer occupation do not depend on the number of connections K.

Numerical example. Consider the case $\mu = 1000$ packets/s, $\tau = 100$ ms, and $B = 100$ packets. For a packet size of 500 bytes, this corresponds to a bottleneck of speed 4 Mb/s with a buffer of 50 K-bytes. Figure 6 shows the window evolution of $K = 3$ connections, starting from the initial state $W_1(0) = 0$, $W_2(0) = 40$ and $W_3(0) = 150$, when the parameter γ is equal to 1/2.

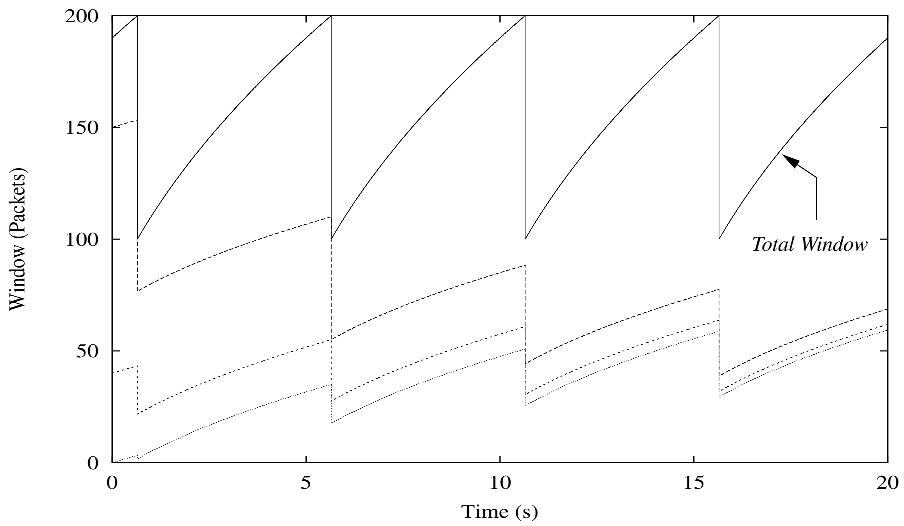

Figure 6: Window evolution of 3 TCP Reno connections

3.2 TCP Vegas

In the following, we assume that each of the K TCP Vegas connections measures accurately $\mathrm{RTT}_{\min}$, so that the dynamics of each window size $W_k(t)$ are given by (5), where the value of $\varepsilon_k(t)$ depends on the buffer occupation of connection k, namely

$$X_k(t) = \mathrm{Diff_k(t)} \times \tau = W_k(t) \left(1 - \frac{\tau}{\mathrm{RTT}(t)}\right).$$

Remark (Perfect $\mathrm{RTT}_{\min}$ measurement). This assumption, which is reasonable in the case of a single connection, may be not realistic in the case of multiple connections, since the queueing delays may be positive at the starting times of the connections. This question is discussed in details in Appendix B.

If the window sizes stabilize, it follows then from (6) that their values $w_1, \ldots, w_K$ in the steady state must satisfy the inequalities:

$$\forall k = 1, \ldots, K, \quad \alpha \leq w_k \left(1 - \frac{\tau}{\mathrm{RTT}}\right) \leq \beta.$$

In particular, RTT cannot be equal to the round-trip propagation delay τ, so that denoting by $w = w_1 + \ldots + w_K$ the total window size in the steady state, we get from (3),

$$\mathrm{RTT} = \frac{w}{\mu},$$

and

$$\forall k = 1, \ldots, K, \quad \alpha \leq w_k \left(1 - \frac{\mu\tau}{w}\right) \leq \beta. \tag{12}$$

In particular, the total throughput and the buffer occupation satisfy in this case

$$\lambda = \mu \quad \text{and} \quad K\alpha \leq X \leq K\beta. \tag{13}$$

Hence, a necessary condition for the window sizes to stabilize is that $K\alpha < B$. In fact, we will show that this condition is also sufficient. We first need the following property.

Contraction Property. *For any connections k, l, such that $W_k(0) \leq W_l(0)$, the function $W_l(t) - W_k(t)$ is non–negative and non–increasing.*

Proof. Since the window dynamics of connections k and l are the same, if the window sizes coincide at some time t, then they coincide forever. The first part of the lemma follows then from the fact that the functions $W_k(t)$ and $W_l(t)$ are piecewise continuous, and *both* multiplied by $\gamma < 1$ at the discontinuity points.

Now using the fact that $W_k(t) \leq W_l(t)$ for all $t \geq 0$, we get from (5) and (6),

$$\frac{dW_k}{dt} < 0 \implies \frac{dW_l}{dt} \leq \frac{dW_k}{dt},$$

and

$$\frac{dW_l}{dt} > 0 \implies \frac{dW_k}{dt} \geq \frac{dW_l}{dt},$$

which implies that the non–negative function $W_l(t) - W_k(t)$ is non–increasing. $\square$

Stabilization. *If $K\alpha < B$, there exists a finite time from which no loss occurs. In addition, the window sizes stabilize in finite time, that is for any initial condition, there exists $t_0 \geq 0$ and a K–uple $w_1, \dots, w_K$, which satisfies (12), such that*

$$\forall t \geq t_0, \quad \forall k = 1, \dots, K, \quad W_k(t) = w_k.$$

Proof. Let $W_k(0)$ and $W_l(0)$ be respectively the minimum and the maximum of all window sizes at time 0. From the above property, $W_k(t)$ and $W_l(t)$ remain respectively the minimum and the maximum of all window sizes at any time $t \geq 0$, and we can define

$$\theta = \lim_{t \to +\infty} (W_l(t) - W_k(t)).$$

We first show that there exists a finite time t_1 from which no loss occurs. Note that when a loss occurs, all windows are multiplied by γ, so that the result is immediate if $\theta > 0$. In the case $\theta = 0$, define

$$\eta = \frac{1}{2}\left(\frac{B}{K} - \alpha\right).$$

Note that $\eta > 0$ by hypothesis. Define t_1 such that

$$\forall t \geq t_1, \quad W_l(t) - W_k(t) \leq \eta,$$

and assume that for some $t \geq t_1$,

$$B - K\eta < X(t) < B.$$

We will show that in this case, the total window size decreases at time t, and this will conclude the first part of the proof. Since $X(t) > 0$, we have

$$\mathrm{RTT}(t) = \frac{W(t)}{\mu}.$$

Using the inequalities

$$\frac{W(t)}{K} - \eta \leq W_k(t) \leq \frac{W(t)}{K},$$

we obtain

$$X_k(t) = W_k(t)\left(1 - \frac{\mu\tau}{W(t)}\right) \geq \frac{W(t)}{K} - \eta - \frac{\mu\tau}{K} = \frac{X(t)}{K} - \eta.$$

Hence, $X_k(t) > \alpha$, and since $X_k(t)$ is the minimum of all buffer occupations at time t, all window sizes decrease at time t.

To prove the second part of the result, we first note that since the function $W_l(t) - W_k(t)$ is non–negative and non–increasing, its derivative tends to zero, and there exists $t_2 \geq t_1$ such that

$$\forall t \geq t_2, \quad 0 \leq \frac{dW_k}{dt}(t) - \frac{dW_l}{dt}(t) < \frac{1}{\mathrm{RTT}_{\max}},$$

where $\mathrm{RTT}_{\max}$ is the maximum RTT, namely

$$\mathrm{RTT}_{\max} = \tau + \frac{B}{\mu}.$$

Let t be a fixed time larger than t_2. If $\alpha \leq X_k(t) \leq X_l(t) \leq \beta$, the system is in equilibrium at time t, with

$$w_1 = W_1(t), \ldots, w_K = W_K(t).$$

If $X_k(t) < \alpha$, we will show that $W_k(t) = W_l(t)$, so that all window sizes are equal at time t, and the system reaches in finite time $t_0 \leq t + w_1 \mathrm{RTT}_{\max}$, the equilibrium

$$w_1 = \ldots = w_K = \frac{\mu\tau}{K} + \alpha.$$

Assume that $W_k(t) < W_l(t)$ (equivalently $X_k(t) < X_l(t)$). If $X_l(t) < \alpha$, then from (5),

$$\frac{dW_k}{dt}(t) = \frac{dW_l}{dt}(t),$$

so that there exists $s \geq t$ such that

$$X_k(s) < \alpha \quad \text{and} \quad X_l(s) \geq \alpha.$$

If $X_l(s) = \alpha$, then $\varepsilon_l(s) \geq 0$, so that

$$\frac{dX_l}{dt}(s) = \frac{dW_l}{dt}(s)\left(1 - \frac{\mu\tau}{W(s)}\right) + W_l(s)\frac{\mu\tau}{W(s)^2}\frac{dW}{dt}(s) > 0,$$

and there actually exists $s \geq t$ such that

$$X_k(s) < \alpha \quad \text{and} \quad X_l(s) > \alpha.$$

This implies

$$\frac{dW_k}{dt}(s) - \frac{dW_l}{dt}(s) \geq \frac{1}{\mathrm{RTT}(s)} \geq \frac{1}{\mathrm{RTT}_{\max}},$$

a contradiction. We show by a similar argument that $X_l(t) > \beta$ implies $W_k(t) = W_l(t)$, so that the system reaches in finite time the equilibrium

$$w_1 = \ldots = w_K = \frac{\mu\tau}{K} + \beta.$$

□

Periodic Regime *If $K\alpha \geq B$, the system reaches in finite time the same periodic regime as that of K TCP Reno connections, that is there exists $t_0 \geq 0$, such that*

$$\forall t \geq t_0, \quad W(t) = \mathcal{W}(t - t_0),$$

where $\mathcal{W}(t)$ is the periodic function of period T defined in §3.1, and

$$\forall t \geq t_0, \quad W_k(t) = \gamma^{\lceil \frac{t-t_0}{T} \rceil} \left(W_k(t_0) - \frac{W(t_0)}{K} \right) + \frac{W(t)}{K}.$$

Proof. We use the same notations as above. In particular, $W_k(t)$ and $W_l(t)$ denote respectively the minimum and the maximum of all window sizes at any time $t \geq 0$. Let t_1 be such that

$$\forall t \geq t_1, \quad 0 \leq \frac{dW_k}{dt}(t) - \frac{dW_l}{dt}(t) < \frac{1}{\mathrm{RTT}_{\max}}.$$

and let $t \geq t_1$ be such that no loss occurs at time t. Since $K\alpha \geq B$, we have

$$X_k(t) \leq \frac{X(t)}{K} < \alpha.$$

If $X_l(t) \geq \alpha$, then by the same argument as that used in the proof of stabilization, there exists $s \geq t$ such that no loss occurs at time s, $X_k(s) < \alpha$ and $X_l(s) > \alpha$. Hence,

$$\frac{dW_k}{dt}(s) - \frac{dW_l}{dt}(s) \geq \frac{1}{\mathrm{RTT}(t)} \geq \frac{1}{\mathrm{RTT}_{\max}},$$

a contradiction. Therefore, $X_l(t) < \alpha$, and from (6), $\varepsilon_1(t) = \ldots = \varepsilon_K(t) = 1$. Thus at any time $t \geq t_1$, either a loss occurs or all window sizes increase at rate 1/RTT. In particular, each TCP Vegas connection behaves exactly like a TCP Reno connection, and the result follows from §3.1. □

In view of the above results, the ratio[3]

$$L = \left\lfloor \frac{B}{\alpha} \right\rfloor, \tag{14}$$

is a critical value for the number of TCP Vegas connections. If $K \leq L$, the window sizes stabilize in finite time, whereas if $K > L$, the mechanisms introduced in the congestion avoidance phase of TCP Vegas are not effective, and after a transient period, the system behaves exactly like K TCP Reno connections.

[3]For any $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the only integer belonging to $[x - 1, x)$.

It is worth noting that when the window sizes stabilize, there is a continuum of possible equilibria, depending on the initial state. In view of (12), denoting respectively by w_k and w_l the minimum and the maximum of all window sizes in the steady state, we have

$$1 \leq \frac{w_l}{w_k} \leq \frac{\beta}{\alpha},$$

so that the *fairness* of TCP Vegas depends in a crucial way on the ratio β/α. In the case $\alpha = 1$ and $\beta = 3$ for instance, some users could receive three times more bandwidth than other users. On the other hand, when $\alpha = \beta$, TCP Vegas is stable and fair, since the window sizes converge in finite time to the single equilibrium, given by

$$w_1 = \ldots = w_K = \frac{\mu\tau}{K} + \alpha. \tag{15}$$

In view of the above results, the main parameter of TCP Vegas is α, since it determines the maximum number of connections under which the system stabilizes, according to (14). It is clear that a positive value for α is required, but it should not be chosen too large, so as to keep the amount of data per connection buffered in the network as low as possible. Thus reasonable values for α are 1 or 2, as proposed in [3]. Concerning the parameter β, it was introduced to allow the window to stabilize and to make the protocol less sensitive to small variations in the available bandwidth. In fact, there is a *trade-off* on the choice of the difference $\delta = \beta - \alpha$.

Large *vs* **small** δ**.** *For large values of δ (compared to α), TCP Vegas is robust but unfair, whereas for small values of δ, TCP Vegas is fair (at least for homogeneous connections, the general case is treated in Section 5), but not robust, in the sense that small variations in the available bandwidth may lead to changes in the window size.*

In the limiting case $\delta = 0$, the steady state would actually consist of ± 1 variations of the window sizes around the equilibrium (due to the packetization effect), a phenomenon which is not captured by the fluid approximation used. We argue that these oscillations which are very *slow* (due to the linear increasing-decreasing rate of the window) and of *small magnitude*, do not significantly damage the performance of the flow control. Furthermore, these oscillations may be suitable to make the protocol responsive to sporadic changes in the available bandwidth. Note that in the worst case where these oscillations are *in phase*, the maximum number of connections under which the window sizes stabilize would be changed into

$$L' = \left\lfloor \frac{B}{\alpha + 1} \right\rfloor.$$

Thus in the following, we will always consider the case $\alpha = \beta$, and the subsequent analysis is valid only for practical implementations of TCP Vegas where δ is very small compared to α. However, it is clear from (12) that in other cases (for instance when (α, β) is equal to (1,3) or (2,4), as proposed in [3]), the following results provide *upper* and *lower* bounds on the performance of the protocol.

Example. Consider the same example as above, but in the case of 3 TCP Vegas connections, and with the parameters $\alpha = \beta = 1$. As shown in Figure 7, the total window size stabilizes to the value $\mu\tau + K\alpha = 103$, which is very close to the optimal window size $\mu\tau$.

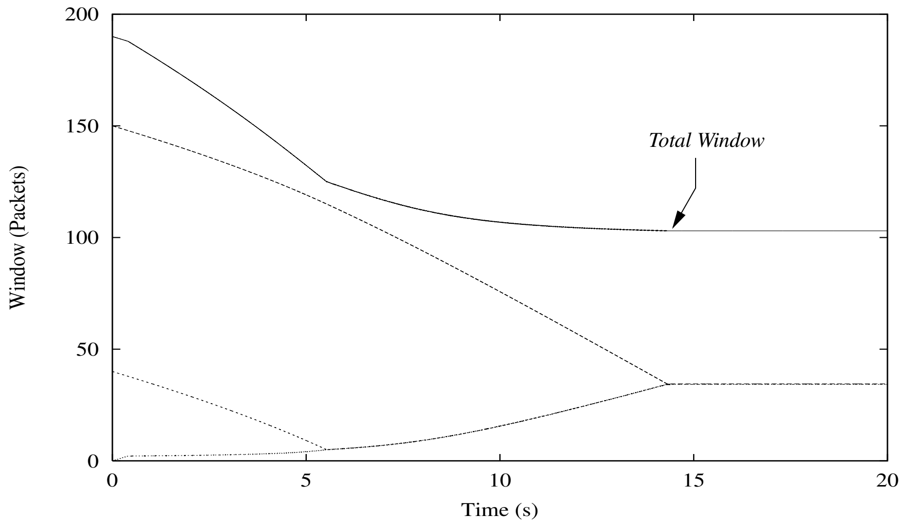

Figure 7: Window evolution of 3 TCP Vegas connections

4 Performance Evaluation

In this section, we consider a model where the number of connections sharing the same link is *dynamic*. More precisely, we assume that the starting times of the connections form a Poisson process of intensity ν (this assumption is reasonable as soon as these connections are generated by a large number of users which have mutually independent behaviors), and that the sizes of the files are i.i.d., with a general distribution of mean σ.

All connections use the same version of TCP, either Reno or Vegas. We assume that the sizes of the files are sufficiently *large*, so that at any time $t \geq 0$, the connections can be assumed to be in the steady state described in Section 3. Under some condition on the *load* of the network, defined by $\rho = \nu\sigma/\mu$, we will show that this system is stable. Let K, λ and X be respectively the number of connections, the total throughput and the buffer occupation in the steady state. We will evaluate the performance of TCP Reno and TCP Vegas both in terms of *utilization* and *congestion* of the network, respectively defined by

$$U = \frac{1}{\mu}\,\mathbb{E}(\lambda \mid K > 0) \quad \text{and} \quad C = \frac{1}{B}\,\mathbb{E}(X \mid K > 0).$$

We will distinguish between two cases, depending on the value of the bandwidth–delay product $\mu\tau$ compared to buffer size B. As in §3.1, we denote by ω the ratio between bandwidth–delay product and the buffer size.

4.1 Small bandwidth–delay product

When $\omega \leq \gamma/(1-\gamma)$, both TCP Reno and TCP Vegas use all the available bandwidth, that is the total throughput is equal to μ whatever the number of connections, and

$$U_{\text{Reno}} = U_{\text{Vegas}} = 1.$$

In addition, the number of connections present in the system corresponds to the number of customers in a $M/G/1/PS$ queue, so that the stability condition is $\rho < 1$, and in the steady state, the law of the number of connections is geometric:

$$\forall k \in \mathbb{N}, \quad \mathbb{P}(K = k) = \rho^k(1-\rho).$$

Since the buffer occupation of TCP Reno is not sensitive to the number of connections, we immediately get from (11),

$$C_{\text{Reno}} = \frac{2(1+\gamma+\gamma^2)}{3(1+\gamma)}(\omega+1) - \omega. \tag{16}$$

Using (15), we obtain

$$C_{\text{Vegas}} = \frac{1}{B}\sum_{k=1}^{L} k\alpha\,\mathbb{P}(K = k \mid K > 0) + C_{\text{Reno}}\,\mathbb{P}(K > L \mid K > 0),$$

that is

$$C_{\text{Vegas}} = \frac{\alpha}{B}\left(\frac{1-\rho^{L+1}}{1-\rho} - (L+1)\rho^L\right) + C_{\text{Reno}}\,\rho^L.$$

4.2 Large bandwidth–delay product

In the case $\omega \geq \gamma/(1-\gamma)$, we have seen in §3.1 that TCP Reno does not use all the available bandwidth, but only a fraction of it, given in view of (10) by

$$\mu_0 = \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1}\,\mu \leq \mu.$$

Hence, the number of TCP Reno connections present in the system still corresponds to the number of customers in a $M/G/1/PS$ queue, but for a server of speed μ_0 instead of μ. In particular, the stability condition of the system is now

$$\rho < \frac{\mu_0}{\mu}.$$

Under this condition, we have

$$U_{\text{Reno}} = \frac{\mu_0}{\mu} = \frac{(1-\gamma^2)(\omega+1)^2}{2(1-\gamma)(\omega^2+\omega)+1}, \tag{17}$$

and from (10),

$$C_{\text{Reno}} = \frac{\omega+2/3}{2(1-\gamma)(\omega^2+\omega)+1}.$$

Concerning TCP Vegas, the total throughput depends on the number of connections. More precisely,

$$\lambda = \begin{cases} \mu & \text{when} \quad K < L, \\ \mu_0 & \text{when} \quad K \geq L. \end{cases}$$

Hence, the stability condition of the system is still $\rho < \mu_0/\mu$, but the number of connections K in the steady state is not geometric, and depends on the distribution of the file sizes. However, when L is sufficiently large compared to $(1-\rho)^{-1}$, it is likely that K is not very sensitive to the distribution of the file sizes. Thus to get analytical results, we will assume that the law of the file sizes is *exponential* in this particular case. The number of connections is then a birth–death process, shown in Figure 8. It follows from the Chapman-Kolmogorov equations of this Markov chain that

$$\forall k \in \mathbb{N}, \quad \mathbb{P}(K=k) = \begin{cases} \mathbb{P}(K=0)\,\rho^k & \text{if} \quad k \leq L, \\ \mathbb{P}(K=0)\,\rho^L \rho_0^{k-L} & \text{if} \quad k > L, \end{cases}$$

where $\rho_0 = \rho(\mu/\mu_0)$, and

$$\mathbb{P}(K=0) = \frac{(1-\rho)(1-\rho_0)}{1-\rho_0+\rho^L(\rho_0-\rho)}.$$

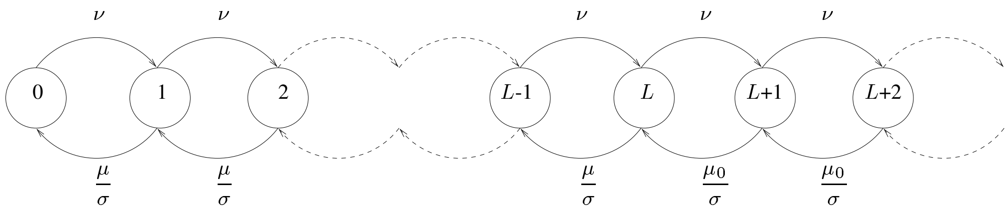

Figure 8: Birth–death process

Using this, we get from the equality

$$U_{\text{Vegas}} = 1 \times \mathbb{P}(K \leq L \mid K > 0) + U_{\text{Reno}}\, \mathbb{P}(K > L \mid K > 0),$$

that

$$U_{\text{Vegas}} = \frac{1 - \rho_0 + \rho^L(\rho_0 - \rho)}{1 - \rho_0 + \rho^{L-1}(\rho_0 - \rho)}.$$

In the same way, we get from the equality

$$C_{\text{Vegas}} = \frac{1}{B} \sum_{k=1}^{L} k\alpha\, \mathbb{P}(K = k \mid K > 0) + C_{\text{Reno}}\, \mathbb{P}(K > L \mid K > 0),$$

that

$$C_{\text{Vegas}} = \frac{1 - \rho_0}{1 - \rho_0 + \rho^{L-1}(\rho_0 - \rho)} \times \left[\frac{\alpha}{B} \left(\frac{1 - \rho^{L+1}}{1 - \rho} - (L+1)\rho^L \right) + C_{\text{Reno}}\, \frac{\rho_0(1 - \rho)}{\rho(1 - \rho_0)}\, \rho^L \right].$$

4.3 Discussion

The results obtained are illustrated by Figures 9 and 10 for a buffer size $B = 100$, and with the usual parameters $\gamma = 1/2$ and $\alpha = 1$. Note that in this case, the critical value of the bandwidth–delay product is $\gamma B/(1 - \gamma) = B$. The stability condition $\rho < \rho_{\max}$, where

$$\rho_{\max} = \begin{cases} 1 & \text{if } \mu\tau \leq B, \\ \dfrac{\mu_0}{\mu} & \text{otherwise,} \end{cases}$$

is also represented in both figures. When the load ρ tends to its maximal value $\rho_{\max}$, the mean number of connections becomes large, so that TCP Vegas has the same behavior as TCP Reno (see §3.2) and thus the same performance. Except for this case, TCP Vegas clearly outperforms TCP Reno, both in terms of buffer occupation and utilization of the available bandwidth.

For small bandwidth–delay products ($\mu\tau \leq 100$), both TCP Reno and TCP Vegas fully utilize the available bandwidth. The major difference is that the buffer occupation of TCP Vegas is much lower than the buffer occupation of TCP Reno. In particular, whereas the buffer occupation of TCP Vegas is very close to zero whatever the load of the network, we have in view of (16),

$$C_{\mathrm{Reno}} \longrightarrow \frac{2(1+\gamma+\gamma^2)}{3(1+\gamma)} \quad \text{when} \quad \omega \to 0,$$

that is $C_{\mathrm{Reno}} \sim 78\ \%$ in the case $\gamma = 1/2$ for small values of the bandwidth–delay product. Note that the congestion would be even more severe for higher values of γ. This gap between both protocols reflects the benefits of the window mechanism of TCP Vegas, which consists in stabilizing the windows instead of discovering the available bandwidth by filling the buffer.

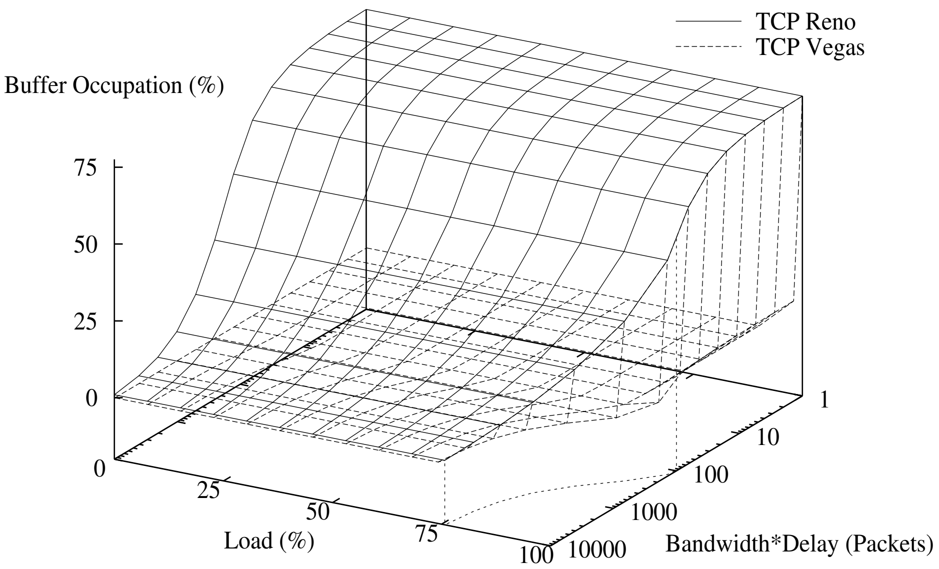

Figure 9: Buffer occupation

For large bandwidth–delay products ($\mu\tau \geq 100$), TCP Reno under-utilizes the network resources, due to the fact that when a loss occurs, all windows are multiplied by γ, and there is not enough fluid to "fill the pipe" $\mu\tau$. In particular, we have in view of (17),

$$U_{\text{Reno}} \longrightarrow \frac{1+\gamma}{2} \quad \text{when} \quad \omega \to \infty,$$

that is $U_{\text{Reno}} \sim 75\ \%$ in the case $\gamma = 1/2$ when $\mu\tau$ is close to 10^4 packets. On the other hand, except when the load of the network is very high, TCP Vegas fully utilizes the available bandwidth.

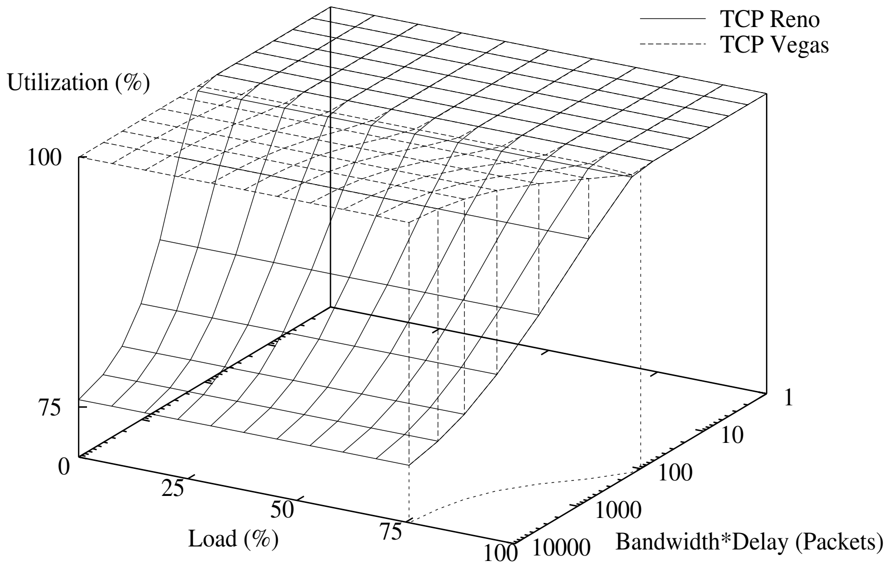

Figure 10: Utilization of the available bandwidth

5 Fairness

In this section, we focus on the case of *heterogeneous* connections, which do not experiment the same propagation delay. As in Section 3, we consider a fixed number of connections K sharing the same bottleneck. We denote by τ_k the propagation delay of connection k. Denoting by X the total buffer occupation as above, we have

$$X = 0 \quad \Longleftrightarrow \quad \sum_{k=1}^{K} \frac{W_k}{\tau_k} \leq \mu.$$

When the buffer is non–empty ($X > 0$), the RTT of connection k is given by

$$\mathrm{RTT}_k = \tau_k + \frac{X}{\mu}.$$

By Little's law, we have $\lambda_k \mathrm{RTT}_k = W_k$, where λ_k is the throughput of connection k. Using the fact that $\lambda_1 + \ldots + \lambda_K = \mu$, we obtain the following implicit expression for X:

$$\sum_{k=1}^{K} \frac{W_k}{X + \mu\tau_k} = 1.$$

5.1 TCP Reno

The dynamics of the system consist of previous expressions and the window dynamics of each connection, namely

$$\begin{cases} \dfrac{dW_k}{dt} = \dfrac{1}{\mathrm{RTT}_k(t)} \\ \\ X(t) = B \implies W_k(t^+) = \gamma\, W_k(t). \end{cases}$$

Since this system is not tractable in general, we will consider asymptotic results in the ratios $\omega_1, \ldots, \omega_K$ of the bandwidth–delay products $\mu\tau_1, \ldots, \mu\tau_K$ with respect of the buffer size.

We first assume that $\omega_1, \ldots, \omega_K$ are very *small*. After a transient period, the buffer is never empty and the RTT are roughly the same (equal to the queueing delay). In particular, the difference between window sizes vanishes as in §3.1, and by Little's law, we get in the steady state,

$$\forall k, l = 1, \ldots, K, \quad \frac{\lambda_k(t)}{\lambda_l(t)} \approx \frac{\mathrm{RTT}_l(t)}{\mathrm{RTT}_k(t)} \approx 1.$$

Now assume that $\omega_1, \ldots, \omega_K$ are very *large*. In this case, the buffer is almost always empty and the RTT are roughly equal to the propagation delays. Hence, we have in the steady state,

$$\forall k, l = 1, \ldots, K, \quad \frac{W_k(t)}{W_l(t)} \approx \frac{\tau_l}{\tau_k},$$

and by Little's law,

$$\frac{\lambda_k(t)}{\lambda_l(t)} \approx \frac{\tau_l}{\tau_k} \frac{\mathrm{RTT}_l(t)}{\mathrm{RTT}_k(t)} \approx \left(\frac{\tau_l}{\tau_k}\right)^2.$$

Thus in the case of large bandwidth-delay products, TCP Reno significantly discriminates against connections with *larger* propagation delays, as already observed in [7].

Finally, we consider the case of K' connections (denoted with a prime) with *small* bandwidth-delay products, and K connections with *large* bandwidth-delay products, still denoted by $\omega_1, \ldots, \omega_K$. We use the same approximations as above, namely

$$\mathrm{RTT}'_k \approx \frac{\mu}{X} \quad \text{and} \quad \mathrm{RTT}_k \approx \tau_k.$$

From previous analysis, the connections with small propagation delays have the same window dynamics, and thus will have the same throughput in the steady state, roughly equal to $\lambda' \approx \mu/K'$. In addition, the buffer occupation satisfies the differential equation

$$\frac{dX}{dt} \approx \frac{K'\mu}{X}.$$

As in §3.1, the steady state of the system is periodic with period

$$T \approx \frac{(1-\gamma^2)B^2}{2K'\mu}.$$

On the other hand, since the window sizes of the connections with large propagation delays increase linearly, respectively at rates $1/\tau_1, \ldots, 1/\tau_K$, we have

$$T \approx (1-\gamma)W_1\tau_1 \approx \ldots \approx (1-\gamma)W_K\tau_K,$$

where $W_1, \ldots, W_K$ are the window sizes of these connections when a loss occurs. Since the average throughput of connection k is given by

$$\lambda_k \approx \frac{1+\gamma}{2}\frac{W_k}{\tau_k},$$

we finally get

$$\forall k = 1, \ldots, K, \quad \frac{\lambda_k}{\mu} \approx \frac{1}{K'}\left(\frac{1+\gamma}{2\omega_k}\right)^2.$$

Numerical example. Consider as above a bottleneck of speed $\mu = 1000$ packets/s, with a buffer size of $B = 100$ packets. The parameter γ is equal to 1/2. Figure 11 shows the throughput evolution of $K' = 2$ connections with small propagation delays ($\tau'_1 = 10$ ms, $\tau'_2 = 20$ ms), starting from the initial window sizes $W'_1(0) = 0$, $W'_2(0) = 40$, and a single connection with a large propagation delay ($\tau_1 = 500$ ms), starting from the initial window size $W_1(0) = 150$. By simulation [12], we find the following bandwidth sharing in the steady state:

$$\frac{\lambda'_1}{\mu} \approx 55\ \%,\ \frac{\lambda'_2}{\mu} \approx 44\ \% \quad \text{and} \quad \frac{\lambda_1}{\mu} \approx 1\ \%.$$

As expected, both connections with small propagation delays receive roughly half of the available bandwidth, whereas the third connection receives only a small fraction of it, namely

$$\frac{\lambda_1}{\mu} \approx \frac{1}{2}\left(\frac{1+\gamma}{2 \times 5}\right)^2 \approx 1\ \%.$$

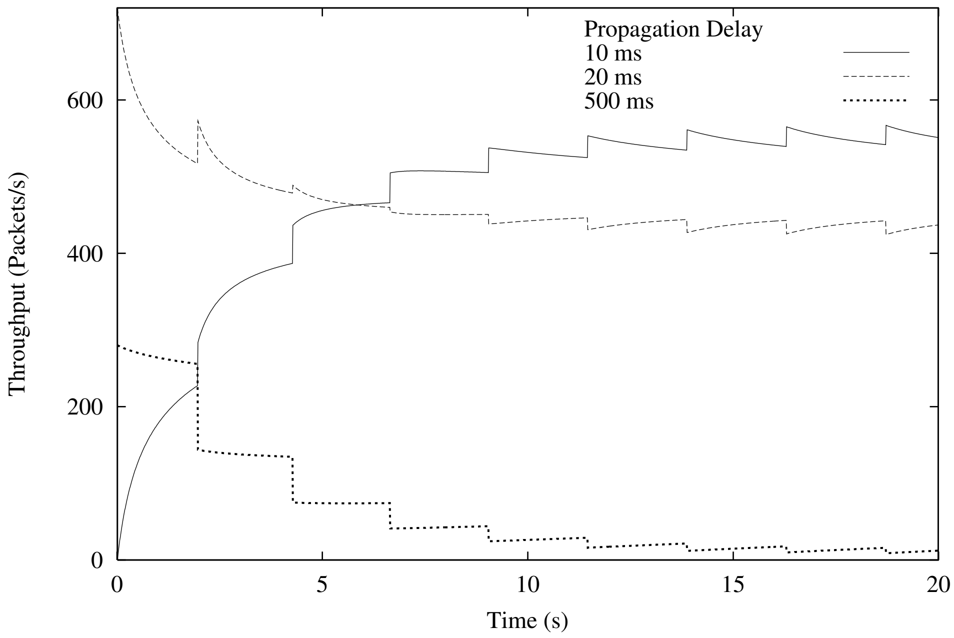

Figure 11: Discrimination of TCP Reno against connections with large propagation delays

5.2 TCP Vegas

As in §3.2, we first note that if the windows stabilize, there is a unique possible equilibrium $w_1, \ldots, w_K$, characterized by the equations

$$\forall k = 1, \ldots, K, \quad w_k \left(1 - \frac{\tau_k}{\mathrm{RTT}_k}\right) = \alpha,$$

so that

$$\lambda_k = \frac{w_k}{\mathrm{RTT}_k} = \frac{\alpha}{\mathrm{RTT}_k - \tau_k} = \frac{\alpha}{X}\,\mu.$$

In particular, each connection receives the same throughput μ/K in the steady state, whatever its propagation delay, and there is a total of $K\alpha$ packets buffered at the bottleneck node, as in the case of homogeneous connections. Thus $K\alpha < B$ is clearly a necessary condition for stabilization. We guess that this condition is also sufficient. However, we will prove only the following partial result. By convention, we assume that $\tau_1 \le \tau_2 \le \ldots \le \tau_K$.

Stabilization. *Assume that at time $t = 0$, the connections $1, \ldots, K-1$, are in equilibrium, that is*

$$\forall k = 1, \ldots, K-1, \quad X_k(0) = \alpha.$$

If $B > K\alpha$, the windows stabilize in finite time, that is there exists $t_0 \ge 0$ such that

$$\forall t \ge t_0, \quad \forall k = 1, \ldots, K, \quad W_k(t) = w_k.$$

Proof. From the expression

$$\frac{X_k}{X} = \frac{\lambda_k}{\mu} = \frac{W_k}{X + \tau_k\mu}, \tag{18}$$

we get using (5),

$$\forall k = 1, \ldots, K, \quad \frac{dX_k}{dt} = \frac{\tau_k\mu}{X + \tau_k\mu}\frac{X_k}{X}\frac{dX}{dt} + \varepsilon_k(t)\frac{X\mu}{(X + \tau_k\mu)^2}.$$

Hence, a sufficient condition for $X_1, \ldots, X_{K-1}$, to remain equal to α is that

$$\forall k = 1, \ldots, K-1, \quad \left|\frac{dX}{dt}\right| \le \frac{X^2}{\alpha\tau_k(X + \tau_k\mu)}.$$

But in this case, we have

$$\frac{dX}{dt} = \frac{dX_K}{dt} = \varepsilon_K(t)\frac{X^2}{(X^2 + \alpha\tau_K\mu)(X + \tau_K\mu)}\mu.$$

In particular, previous condition is satisfied, no loss can occur due to the fact that $B > K\alpha$, and X_K converges in finite time to α. Thus in the steady state, the buffer occupation of each connection is equal to α, and the window sizes are equal to $w_1, \ldots, w_K$, where in view of expression (18),

$$\forall k = 1, \ldots, K, \quad w_k = \frac{K\alpha + \tau_k \mu}{K\alpha}\alpha = \frac{\tau_k \mu}{K} + \alpha.$$

□

Example. Consider the same example as above, where $K = 3$ TCP Vegas connections with propagation delays $\tau_1 = 10$ ms, $\tau_2 = 20$ ms and $\tau_3 = 500$ ms, start with initial window sizes $W_1(0) = 0$, $W_2(0) = 40$, and $W_3(0) = 150$. As shown in Figure 12, after a transient period, the available bandwidth is equally shared between the 3 connections.

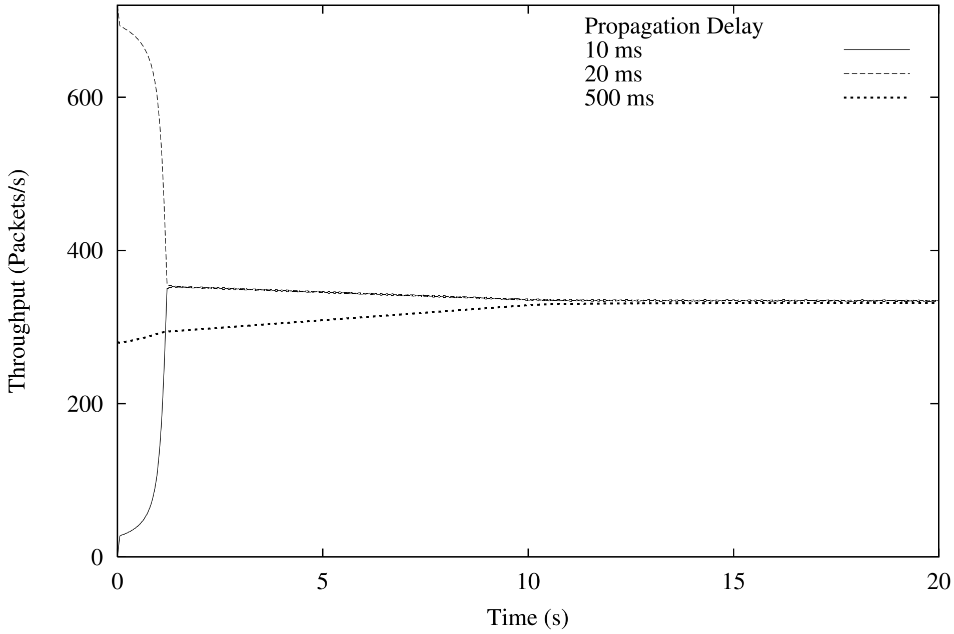

Figure 12: Fairness of TCP Vegas

6 Conclusion

We have used a fluid approximation to compare the efficiency of the flow control of TCP Reno and TCP Vegas. Since these protocols differ essentially in their steady behavior (due to different congestion avoidance phases), we have focused on long-term performance criteria such as the average throughput and the average buffer occupation. The main conclusion is that TCP Vegas, the window mechanism of which consists in stabilizing the window size to the optimal value plus a number of "extra" packets comprised between α and β, is much more *stable*, *efficient* and *fair* than TCP Reno. It is clear that the model used for the analysis is an idealized representation of a TCP connection, as explained in Section 2. However, it gives some insights on the steady behavior of both protocols for sufficiently large windows (so that the discrete nature of the window mechanism may be neglected), and for large file transfers (so that the steady state described in Section 3 may be effectively reached).

Another interesting result is that the window mechanism TCP Vegas is much more *conservative* than that of TCP Reno, and that in the worst case, TCP Vegas behaves exactly as TCP Reno, namely when the number of connections sharing the same bottleneck is larger than B/α, where B is the buffer size of this bottleneck. As a result, the performance of the Internet cannot *a priori* be damaged by the use of TCP Vegas instead of TCP Reno. On the other hand, the main expected benefits of TCP Vegas are the following:

- Since the buffers are not filled up by TCP Vegas connections, the queueing delays in the Internet may be significantly decreased, and this would be of great interest, especially for real-time applications. As illustrated by Figure 9, this effect should be significant when the bandwidth–delay product of the link is small;

- When the bandwidth–delay product of the link is large, TCP Vegas fully utilizes the available bandwidth, whereas TCP Reno utilizes only a fraction of it (see Figure 10). Equivalently, the buffer requirements of TCP Vegas do not depend on the bandwidth–delay product of the link, but only on the number of connections sharing this link, which is a much more convenient and realistic design rule.

It is worth noting that this last feature makes TCP Vegas suitable for *satellite* links. In this case, the bandwidth–delay product is large, and it is crucial to use all the (costly) available bandwidth. In addition, losses which are due to buffer overflow must be avoided, since starting a slow-start phase takes a long time due to long propagation delays. Finally, since the congestion avoidance phase of TCP Vegas allows the window to decrease (this is clearly illustrated by Figure 7), it might not be necessary to fix an arbitrary limit to the window size depending on the bandwidth–delay product, as proposed in [5] with the window scale option.

The last issue, which was not addressed in this paper, concerns the deploying of TCP Vegas in the Internet. It may be argued that due to its conservative strategy, a TCP Vegas user will be severely disadvantaged compared to TCP Reno users, as illustrated in Figure 13 for the same numerical values as above ($\mu = 1000$ packets/s, $\tau = 100$ ms and $B = 100$ packets). If such a discrimination arises, the only way to incite users to behave “socially” (that is to use TCP Vegas instead of TCP Reno) would be to adopt a pricing scheme which penalizes resource-consuming behaviors. However, it is still unclear whether the expected drop in unnecessary retransmissions due to the conservative window mechanism would not finally result in a higher “goodput” for TCP Vegas users, as observed in [1, 3] by simulations and measurements in the Internet. If this turns out to be the general case (here analytical results would be of great help), it is likely that TCP Vegas, which improves both the individual utility of the users and the global utility of the network, will gradually replace TCP Reno.

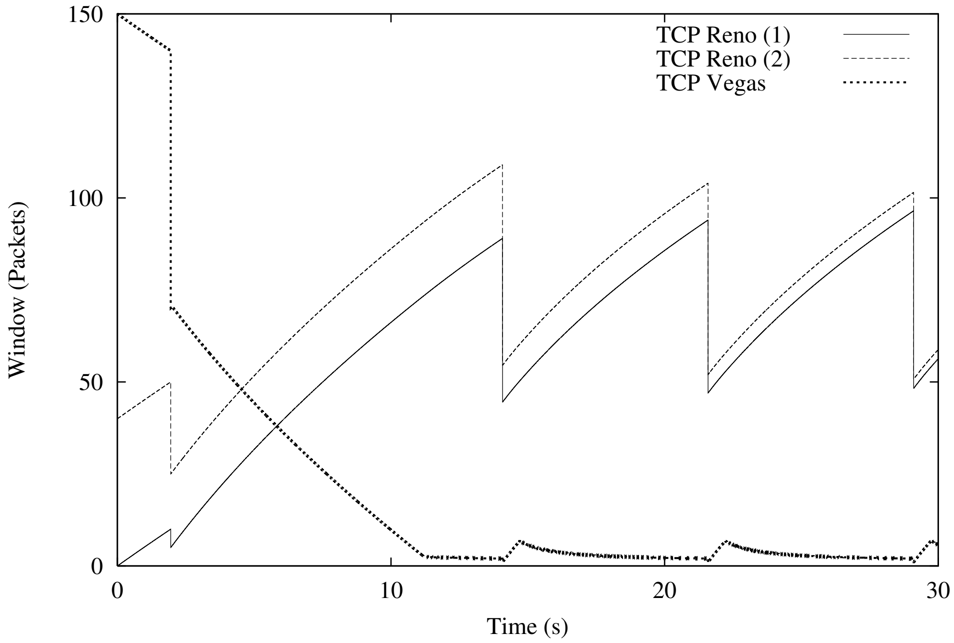

Figure 13: Competition between TCP Reno and TCP Vegas

A Abbreviations

Internet Protocols

IP	Internet Protocol
TCP	Transmission Control Protocol
FTP	File Transfer Protocol
HTTP	Hyper-Text Transfer Protocol
NNTP	News Network Transport Protocol

Window Dynamics

ACK	Acknowledgment
RTT	Round-Trip Time
TO	Time-Out
RTTmin	Minimum Round-Trip Time

B Measurement of RTTmin

Throughout the paper, we have made the assumption that the measurement of the minimum RTT on which the window mechanism of TCP Vegas is based was perfect. In particular, $\mathrm{RTT}_{\min}$ was always taken equal to the round-trip propagation delay τ. In this appendix, we discuss the validity of this assumption.

First note that the measurement of the propagation delay is robust in the sense that it concerns a *static* parameter of the connection[4] and it can only improve with time. However, in the particular (and rare) case where the route changes during the connection time, the propagation delay may increase and thus be *under–estimated* by the source, resulting in a poor utilization of the network resources. To avoid such a misbehavior, one possible solution would be to evaluate the minimum of the M last RTT measures instead of the minimum of *all* RTT measures. But the parameter M should then be carefully chosen in order to discriminate between changes in the route of the packets and transient congestion of the network. In the general case where the route remains the same during the connection time, the propagation delay can only be *over–estimated* (due to the queueing delays), thus cannot affect the performance of TCP Vegas in terms of utilization of the available bandwidth, but only in terms of buffer occupation.

[4]This is not the case of the available bandwidth for instance. The fact that this *dynamic* parameter of the connection varies with respect to the intensity of the cross traffic makes its estimation very difficult [6].

To evaluate the effect of biased $\mathrm{RTT}_{\min}$ measurement on the buffer occupation, we consider as in Section 3.2 the case of K TCP Vegas connections sharing the same link, and starting respectively at times $t_1 \le \ldots \le t_K$. We denote by τ_k the $\mathrm{RTT}_{\min}$ measure of connection k, and consider the worst case where this measure is equal to the RTT of the first packet of this connection. In the case $\alpha = \beta$, the steady state of these K connections is characterized by the equations:

$$\forall k = 1, \ldots, K, \quad w_k \left(1 - \frac{\tau_k}{\mathrm{RTT}}\right) = \alpha.$$

In particular, the RTT is larger than $\tau_1, \ldots, \tau_K$, thus larger than the round-trip propagation delay τ. Denoting by $w = w_1 + \ldots + w_K$ the total window, we have $\mathrm{RTT} = w/\mu$, so that

$$\sum_{k=1}^{K} \frac{\alpha}{\mathrm{RTT} - \tau_k} = \mu. \tag{19}$$

Hence, there exists a single equilibrium $\mathrm{RTT} = f_K(\tau_1, \ldots, \tau_K)$, and it can be shown as in §3.2 that for a sufficiently large buffer size, this equilibrium is reached in finite time.

Now assume that at time t_k, connections $1, \ldots, k-1$, are in equilibrium. In this case, the $\mathrm{RTT}_{\min}$ measure τ_k of connection k is equal to the RTT in the steady state reached by connections $1, \ldots, k-1$, namely $\tau_1 = \tau$ and

$$\forall k = 1, \ldots, K-1, \quad \tau_{k+1} = f_k(\tau_1, \ldots, \tau_k).$$

Let us show by induction that

$$\forall k = 1, \ldots, K, \quad \tau_k \le \tau + \frac{\alpha}{\mu}(k-1)S_{k-1},$$

where $S_0 = 0$ and for all $k \ge 1$,

$$S_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}.$$

The property holds for $K = 1$. Assuming that it is true for $K \ge 1$, we get

$$\sum_{k=1}^{K} \frac{\alpha}{\left(\tau + \frac{\alpha}{\mu} K S_K\right) - \tau_k} \le \sum_{k=1}^{K} \frac{\mu}{K S_K - (k-1) S_K} = \mu,$$

and it follows then from (19) that

$$\tau_{K+1} \le \tau + \frac{\alpha}{\mu} K S_K.$$

Therefore, the buffer occupation in presence of K connections is smaller than $KS_K\alpha \sim K\log(K)\alpha$. With the assumption of a perfect $\mathrm{RTT}_{\min}$ measurement, we have obtained a buffer occupation equal to $K\alpha$. Thus we can consider that the conclusions of the paper are not significantly biased by this assumption, as far as the performance criteria are concerned. Concerning the fairness, this last result tends to show that TCP Vegas discriminates against older connections since their estimation of the propagation delay is more accurate thus smaller than that of recent connections. In other words, the shortest connections are favoured, and it may be argued that this is a desirable feature of a transmission control protocol [10].

References

[1] J.S. Ahn, P.B. Danzig, Z. Liu and L. Yan, Experience with TCP Vegas: Emulation and experiment, in: *Proceedings of ACM SIGCOMM'95*, 1995.

[2] F. Baccelli and T. Bonald, Window flow control in FIFO networks with cross traffic, to appear in Queueing Systems, special issue on Stochastic Stability, 1999.
Available as INRIA Research Report:
`http://www.inria.fr/rapports/sophia/RR-3434.html`

[3] L.S. Brakmo and L.L. Peterson, TCP Vegas: End to end congestion avoidance on a global Internet, IEEE J. of Selected Areas in Communication 13 (1995), 1465-1480.

[4] V. Jacobson, Congestion avoidance and control, in: *Proceedings of ACM SIGCOMM'88*, 1988.

[5] V. Jacobson, R. Braden and D. Borman, TCP Extensions for High Performance, RFC 1323, 1992.

[6] S. Keshav, A control-theoretic approach to flow control, in: *Proceedings of ACM SIGCOMM'91*, 1991.

[7] T.V. Lakshman and U. Madhow, Performance analysis of window-based flow control using TCP/IP: The effect of high bandwidth-delay products and random loss, in: *Proceedings of HPN'94*, 1994.

[8] A.A. Lazar, Optimal flow control of a class of queueing networks in equilibrium, IEEE Transactions on Automatic Control 28 (1983), 1001-1007.

[9] D. Mitra, Asymptotically optimal design of congestion control for high speed data networks, IEEE Transactions on Communications 40 (1992), 301-311.

[10] J.W. Roberts and L. Massoulié, Bandwidth sharing and admission control for elastic traffic, in: *Proceedings of ITC Specialist Seminar*, Yokohama, 1998.

[11] S. Shenker, L. Zhang and D.D Clark, Some observations on the dynamics of a congestion control algorithm, ACM Computer Communication Review, 20 (1990), 30-39.

[12] `http://www.inria.fr/mistral/pub/vegas.html`

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399