Compound TCP with Random Losses

Alberto Blanc, Konstantin Avrachenkov, Denis Collange, Giovanni Neglia

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Performance de Compound TCP en présence de pertes aléatoires

Alberto Blanc — Konstantin Avrachenkov — Denis Collange — Giovanni Neglia

N° 6736

Decembre 2008

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6736--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Performance de Compound TCP en présence de pertes aléatoires

Alberto Blanc [*] , Konstantin Avrachenkov [†] , Denis Collange [*] ,
Giovanni Neglia [†]

Thème COM — Systèmes communicants
Équipes-Projets Maestro

Rapport de recherche n° 6736 — Decembre 2008 — 17 pages

Résumé : Nous analysons les performances d'une longue connexion Compound TCP isolée en présence de pertes de paquets aléatoires. Compound TCP est une nouvelle version de TCP implémentée sur Microsoft Windows pour améliorer les performances des transferts sur des réseaux à grand produit délai-bande passante. Nous calculons grâce à un modèle markovien l'évolution la fenêtre d'émission Compound TCP, et nous en déduisons la distribution stationnaire et le débit moyen d'une connexion TCP. Nous remarquons que l'approximation usuelle de ce système, basée sur un "cycle typique", sous-estime la fenêtre moyenne et sa variance, tandis que le modèle Markovien donne des résultats plus proches des simulations. Nous utilisons ce même modèle pour comparer Compound TCP et TCP Reno. Nous notons que Compound TCP donne toujours un débit supérieur ou égal à Reno, tandis que la performance relative en termes de gigue dépend du taux de perte. La gigue générée par Compound TCP est plus élevée que Reno pour des taux de perte élevés, mais plus petite que Reno pour des faibles taux de perte.

Mots-clés : TCP, Compound TCP, Processus de pertes de Bernoulli, Modèle markovien

[*] Orange Labs, 905 rue Albert Einstein, 06921 Sophia Antipolis, France, email: `FirstName.LastName@sophia.inria.fr`

[†] INRIA, 2004 Route des Lucioles, Sophia-Antipolis, France, email: `FirstName.LastName@sophia.inria.fr`

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Compound TCP with Random Losses

Abstract: We analyze the performance of a single, long-lived, Compound TCP (CTCP) connection in the presence of random packet losses. CTCP is a new version of TCP implemented in Microsoft Windows to improve the performance on networks with large bandwidth delay-products. We derive a Markovian model for the CTCP sending window and compute the steady state distribution of the window and the average throughput of a CTCP connection. We observe that the previous approximation, using a "typical cycle," underestimates the average window and its variance while the Markovian model gives more accurate results. We use our model to compare CTCP and TCP Reno. We notice that CTCP gives always a throughput equal or greater than Reno, while relative performance in terms of jitter depends on the specific network scenario: CTCP generates more jitter for moderate-high drop rate values, while the opposite is true for low drop rate values.

Key-words: TCP, Comopund TCP, Bernoulli losses, Markovian Models

1 Introduction

With the increasing popularity of faster access links like Fiber To The Home [9], the current standard TCP is not always ideal. As indicated by Floyd [10] the current standard is not able to reach high rates in realistic environments, i.e. with typical packet loss rates. Many new transport protocols have been proposed and are currently being studied to replace it. Some of them are already implemented in the latest versions of some operating systems, like Compound TCP (CTCP) on Windows, and Cubic (and others) on Linux. For a survey and comparative analysis of several high speed TCP versions see, for example, [12, 15, 14]. For the next few years, the new high speed TCP versions will play an increasing role in resource sharing among flows in the Internet. Yet the behavior, the performance, and the impact on the network of these protocols are not well-known. In particular, there is no comprehensive analytical study of CTCP.

CTCP has been presented by Microsoft Research in [20] and [21] in 2006. It is currently submitted as a draft to the IETF Network Working group with minor differences [18]. CTCP is enabled by default in computers running Windows Server 2008 and disabled by default in computers running Windows Vista [8]. It is also possible to add support for CTCP to Windows XP. An implementation of CTCP, based on [21, 18], is also available for Linux [2]. The main objective of the authors of CTCP [21] is to specify a transport protocol which is efficient, using all the available bandwidth, fair and conservative, limiting its impact on the network. They propose to combine the fairness of a delay-based approach with the aggressiveness of a loss-based approach. As the proposal of CTCP is still recent, there are only a few published evaluations of it. The only analytical model of CTCP in [21] is based on a de facto deterministic model.

In the present work we study the performance of CTCP under random losses. There are at least two important motivations to analyze TCP performance under random losses : random losses harm the performance of the current New Reno TCP on high speed optical links and random losses are inherent in wireless networks (WiMax can provide significantly high transmission rates in wireless networks).

The outline of the paper and of our results is as follows. In Section 2 we give a brief overview CTCP. In Section 3 we present a two-dimensional Markov chain model for CTCP. With the help of this model we obtain the long-run average throughput of CTCP and the distribution of its congestion window at congestion events. Then, in Section 3.2 we use Palm calculus to obtain the distribution of the congestion window at arbitrary time moments. In Section 3.3 we propose some heuristics and compare them with the accurate two-dimension Markov chain model. Finally, in Section 4 we provide numerical and simulation results which confirm our theoretical findings. In particular, we conclude that CTCP provides a higher throughput than TCP New Reno and , even if more aggressive, it causes less traffic jitter than TCP New Reno on high speed links with small random losses.

Due to space constraints, some technical details are provided in a companion technical report [3].

2 Compound TCP Overview

In this section we give a very brief overview of CTCP (see [21] for a complete description). The main idea of CTCP is to quickly increase the window as long as the network path is not fully utilized, then to keep it constant for a certain period of time and finally to increase it by one MSS (Maximum Segment Size) per round trip time just like TCP Reno. We will often use the term "phases" for these three different behaviors.

During phase 1 the sender computes the value at the $(i+1)$-th round trip as $w_{i+1} = w_i + \alpha w_i^k$ (as suggested in [21] we use $\alpha = 1/8$ and $k = 3/4$). At each round trip the sender estimates the bandwidth-delay product and the amount of data backlogged in the network using the same method adopted by TCP Vegas [7]. If the amount of backlogged data is greater than a certain threshold (γ, usually set to 30 [21, 19]) the sender switches to phase 2 and keeps the window constant. This constant value corresponds to the sum of the estimated bandwidth-delay product and the estimated amount of backlogged data. We consider an ideal behaviour of CTCP, assuming that such estimates are correct. In such case the window in phase 2 is equal to $\theta \triangleq \mu\tilde{\tau}+\gamma$, where μ is the capacity of the bottleneck link and $\tilde{\tau}$ is the round trip propagation delay. In reality any queue size estimate available at the sender is outdated due to feedback delays, this fact combined with the CTCP algorithm presented in [21] causes the window to oscillate during this phase as we analyzed in [5]. The length of phase 2 is dictated by the "congestion component" of the window. In fact in CTCP the congestion window w is the sum of two components : the delay window w_d and the congestion window w_c. The congestion component is incremented by one every round trip (just like the TCP Reno congestion window) and when this component reaches θ phase 2 ends. The delay component is set such that the value of the total window ($w_i = w_{c_i} + w_{d_i}$) at the i-th round trip time follows the following evolution in absence of packet losses :

$$w_i = \begin{cases} w_{i-1} + \delta_i & \text{, if } w_{i-1} + \delta_i < \theta \\ \theta & \text{, if } w_{i-1} + \delta_i \geq \theta \text{ and } w_{c_0} + i < \theta \\ w_{i-1} + 1 & \text{, otherwise} \end{cases} \tag{1}$$

where $\delta_i = \max\left\{ \left\lfloor \alpha w_{i-1}^k \right\rfloor, 1 \right\}$. Figure 1 shows the three phases of the window evolution. When a loss occurs both the total window and the congestion window are halved.

3 Performances with Random Losses

We consider a single long lived TCP compound flow using a path with $\mu\tilde{\tau}$ bandwidth delay product and buffer size equal to b. The flow will experience a loss every time that its window size reaches the value $\mu\tilde{\tau} + b$. For this reason, we can consider $w_{max} = \mu\tilde{\tau} + b$ as an upper bound for the the window size. Beside the deterministic losses due to buffer overflow, we consider also that each packet can be dropped with some probability p, independently from all other packets, i.e. according to a Bernoulli process. In what follows we derive the throughput and the window distribution in steady state. We are going to assume that w can only take integer values.

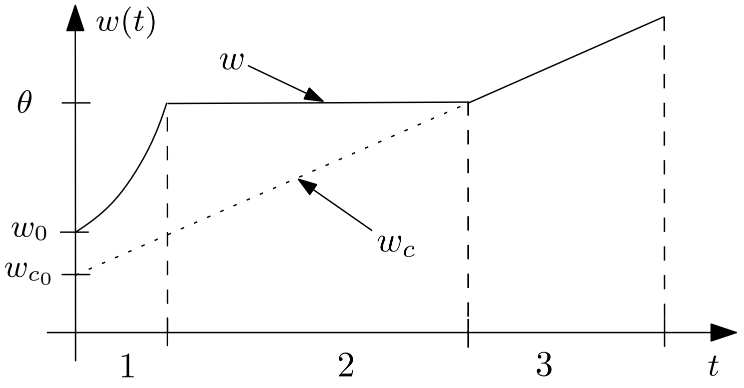

FIG. 1 – The evolution of w and w_c in CTCP

3.1 Throughput calculation

We define a cycle as the time interval between two consecutive losses. We denote as $w_{c_n}^t$ (respectively $w_{d_n}^t$) the congestion (respectively delay) window at the begin of the $(n+1)$-th round trip time of the t-th cycle. We will omit the superscript t whenever it is clear which cycle is being considered.

We observe that in our framework the evolution of the window in each cycle t depends from previous cycles only through the window value at the begin of the cycle, or, more precisely, through the two initial values $w_{c_0}^t$ and $w_{d_0}^t$, which can be determined by the final value of the windows at the $t-1$-th cycle. This also implies that it is possible to use the renewal reward theorem to compute the average throughput as (see [16]) :

$$E[\lambda] = \frac{E[S]}{E[T]} \tag{2}$$

where λ is the throughput (in MSS/s), S is the total number of packets sent during a cycle and T is the duration of the cycle.

Both $E[T]$ and $E[S]$ can be evaluated starting from the knowledge of the distribution size of the two (correlated) random variables W_{c_0} and W_{d_0}. We denote $g(w_{c_0}, w_{d_0})$, the probability mass function (pmf) of these random variables. We first show how to derive the distribution of cycle duration from $g(w_{c_0}, w_{d_0})$ and then derive the pmf $g()$ itself. $E[S]$ can be evaluated similarly to $E[T]$.

A cycle has length equal to n if there is a loss at the n-th round trip time. As we are assuming that there is a loss whenever $w = w_{\max}$, all cycles have a finite length. Let $m(w_0) = \min\{n | w_n \geq w_{\max}\}$ be the maximum possible length (in round trips) of a cycle starting at w_0. For $n < m(w_0)$ the probability of a cycle having length equal to n can be derived from the Bernoulli loss process as :

$$\begin{aligned} P[T = n | W_{c_0} = w_{c_0}, W_{d_0} = w_{d_0}] &= (1-p)^{V_{n-1}(w_0)} - (1-p)^{V_n(w_0)} \\ &\triangleq a_n(w_{c_0}, w_{d_0}), \end{aligned} \tag{3}$$

where

$$\mathrm{V}_n(w_0) \triangleq \sum_{i=0}^{n-1} w_i,\ V_0 \triangleq 0,$$

and w_i is computed as in (1) so that V_n is the number of packets sent during the n-th round trip of a cycle starting with $w = w_0$. Both V_n and a_n, as most quantities used in this section, depend on the initial window $w_0 = w_{c_0} + w_{d_0}$ as highlighted by the notation $a_n(w_0)$ and $V_n(w_0)$, even though we will also use the simplified notation a_n and V_n.

The expression of a_n follows from the cumulative distribution function of the geometric random variable $F(x) = 1 - (1-p)^k$ and corresponds to the probability of at least one packet being dropped during the n-th round trip.

The probability that a loss occurs at the $m(w_0)$-th round trip can be evaluated simply considering that $\sum_{i=1}^{m} P[T = i] = 1$:

$$P[T = m_(w_0)|W_{c_0} = w_{c_0}, W_{d_0} = w_{d_0}] = 1 - \sum_{i=1}^{m_(w_0)-1} a_i(w_{c_0}, w_{d_0}).$$

Finally, being that the support of the discrete random variables W_{c_0} and W_{d_0} is finite, we can use a finite sum to compute $P[T]$:

$$P[T = n] = \sum_{w_{c_0}, w_{d_0}} a_n(w_{c_0}, w_{d_0}) g(w_{c_0}, w_{d_0}). \tag{4}$$

In order to compute $g(w_{c_0}, w_{d_0}$ we model the evolution of the window (at the beginning of each cycle) with a Markov chain. The evolution of the window of TCP Reno at the begin of a cycle (w_0^t) has been modeled in other works as a Markov chain (see, for example, [13, 16]). In fact w_0^t is equal to half of the window value at the end of the $(t-1)$-th cycle, which depends only on w_0^{t-1} and on packet loss probability p.

In order to model CTCP we use a two-dimensional discrete Markov chain X_t to account for $w_{c_0}^t$ and $w_{d_0}^t$. For each state (i, j) the first index represents $w_{c_0}^t$ and the second $w_{d_0}^t$. For any pair of states it is possible to compute the transition probability as :

$$P[X_{t+1} = (k, l)|X_t = (i, j)] = \sum_{n \in B} a_n(i, j)$$

with $w_{c_0}^t = i$, $w_{d_0}^t = j$, $B = \left\{n | \left\lfloor w_{c_n}^t/2 \right\rfloor = k, \left\lfloor w_{d_n}^t/2 \right\rfloor = l\right\}$, where $w_{c_n}^t$ and $w_{d_n}^t$ are evaluated according to (1). The sum on the right hand side is needed because different pairs ($w_{c_n}^t$,$w_{d_n}^t$) can originate, after a loss, the same pair ($w_{d_0}^{t+1}$,$w_{d_0}^{t+1}$) as we use integer values for the window. As $w \leq w_{\max}$ we have that $w_{c_0} \leq \lfloor w_{\max}/2 \rfloor \triangleq N$ and, if θ is the value of the window during the constant window phase, $w_{d_0} \leq \lfloor \theta/2 \rfloor \triangleq M$ (as $w_d \leq \theta$). Combining these two bounds we obtain that the number of states in the Markov chain is NM. Using the ARPACK implementation of the Arnoldi method [17] it is possible to efficiently calculate the steady state distribution of the Markov chain X_t even for large values of NM. The more time consuming step is actually to compute the transition matrix for X. The complexity of the algorithm we used is $O(MN^2)$; we believe that it is not possible to decrease the complexity of the algorithm given that it has to compute all the possible transitions and these grow like MN^2. Note that the number of possible transitions for a Markov chain with NM states is N^2M^2 therefore we already take into account that, in this case, some transitions are not possible.

Once the steady state distribution of the Markov chain X_t $(g(w_{c_0}, w_{d_0}))$ is derived, we can use it to compute $E[T]$, $E[S]$ and then the average throughput using (2).

If the queueing delays are negligible with respect to the propagation delays it is possible to convert round trip times into seconds by simply multiplying the number or round trip times by the total propagation delay. If this approximation is not acceptable we can account for the queueing delays by estimating the queue size q as $q = w - \mu\tilde{\tau}$ (this is true as long as $w > \mu\tilde{\tau}$ and the source has been sending date at a rate greater than μ over the last round trip). Using this estimate we have that the round trip time is w/μ whenever $w \geq \mu\tilde{\tau}$.

For a given n, w_{c_0} and w_{d_0}, the right hand side of (4) corresponds to a cycle starting with $w = w_{c_0} + w_{d_0}$ and lasting n round trip times. For each value of n we know the window size w and we can compute the length in seconds of each round trip and, hence, the total length of each cycle in seconds. Summing over all the possible cycles we can compute the expected value of T in seconds, and, by using (2), the average throughput in MSS/s.

3.2 Steady State Distribution of the Window

In the previous section we have described how to compute the steady state distribution of w_{c_0} and w_{d_0}, and consequently also the value of the window $w_0 = w_{c_0} + w_d$ *at the beginning of each cycle.* In this section we are interested in the steady state distribution of the window as a function of time. We denote as Y_n the value of the window at the begin of the $(n-1)$-th round trip time. Note that Y_n is different both from X_t, which is the value of the window after a packet loss, and from w_n^t, which is the value of the window at the begin of the $(n-1)$-th round trip time in the t-th cycle. Clearly X_t represents a subsequence of the sequence Y_n. We observe that Y_n can also be modeled as a discrete time Markov chain where a transition occurs every round trip. Also this Markov chain is ergodic, hence it admits a steady state and we assume that it is in steady state at time 0.

Using Palm calculus, we first compute $P[Y_n = k]$ starting from $P[W_0 = w_0]$, where Y_n represents the window after n round trip times starting from some arbitrary value (given that all the Markov chains involved are ergodic, the initial value is irrelevant).

Let Z_n be the (discrete) time of the n-th packet drop after time 0. Using the intensity and inversion formulas of Palm calculus [6] we can compute $P[Y_n = k]$ as a function of $P[W_0 = w_0]$:

$$P[Y_n = k] = E\left[1_{\{Y_n=k\}}\right] = P[Z_0 = 0]E^0\left[\sum_{s=1}^{Z_1} 1_{\{Y_s=k\}}\right] \tag{5}$$

$$= \eta E^0\left[\sum_{s=1}^{Z_1} 1_{\{Y_s=k\}}\right] \tag{6}$$

$$= \eta \sum_{w_0,l}\left[P[W_0 = w_0]P[Z_1 = l|W_0 = w_0]\sum_{s=1}^{l} 1_{\{Y_s=k|W_0=w_0\}}\right] \tag{7}$$

where E^0 is the Palm expectation, $1_{\{Y_n=k\}}$ is the indicator function for the event $Y_n = k$ and η is the intensity of the process Z_n. The second (5) and third (6) equalities follow from the inversion and intensity formulas, respectively, while (7) follows from the total probability theorem, conditioning on all the possible values of W_0 and l. Given that Z_n is an ergodic process $P[Z_0 = 0]$ can be computed, using the intensity formula, as the inverse of the expected value of $T \overset{\mathrm{d}}{=} Z_n - Z_{n-1}$ that is as the average length of a cycle (in round trips) so that $\eta = 1/E[T]$ where $E[T]$ can be computed using (4).

If the queueing delays are non-negligible with respect to the propagation delays it is possible to re-normalize $\pi_k \triangleq P[Y_n = k]$ as follows in order to take this into account :

$$\hat{\pi}_k \triangleq \frac{\pi_k q_k}{\sum_{i=1}^{w_{\max}} \pi_k q_k}$$

where

$$q_k \triangleq \max\left\{\tilde{\tau}, \frac{k}{\mu}\right\}.$$

3.3 A Simple Approximation and the Deterministic Response Function

The method described in the previous section provides an exact solution, but it can be computationally expensive for medium and large values of $w_{\max}$ and θ. Using the same method as in [13] it is possible to quickly find an approximate solution for the average window size. The idea is to consider a sequence of "typical" or "average" cycle. If p is the probability that a packet is dropped, the average cycle has exactly $1/p$ packets (provided the probability of reachind $w_{\max}$). Let us denote w_0 the initial window of an average cycle and $w_n(w_0)$ the final window, after n round trip times during which $1/p$ packets are transmitted. Being that the following average cycle has to be identical and that CTCP, as Reno, halves the window at the end of each cycle, it has to be :

$$w_n(w_0) = 2w_0. \tag{8}$$

Imposing the constraint $V_n(w_0) = 1/p$ ($V_n(w_0)$ is defined in section 3.1), we can identify the unique possible value of w_0 and then the unique possible window evolution corresponding to p. Once the window evolution is known, the average window can be obtained and can be plotted as a function of the drop rate.

As previously noted in [1] this approach corresponds to the calculation of what is usually called the "deterministic response function." This function is often used in the literature on TCP. For example, [10] defines the response function as "the function mapping the steady-state packet drop rate to TCP's average sending rate in packets per round-trip time." Where the drop rate is simply the inverse of the number of packets sent during each cycle. In this case the term "drop rate" always refers to this quantity and not to any probabilistic model, while this usage is not the most appropriate one it is, nonetheless, common in the literature. From another point of view, we observe that the average cycle corresponds to the actual evolution of the window under our loss model, when there is no random loss, but the bandwidth delay product and the buffer size are such to cause a deterministic loss every $1/p$ packets.

When the window update rule operates on a round trip time basis -as in CTCP but also in Reno and HighSpeed for example- the deterministic response function does not depend on physical parameters like capacity or propagation delay. In other words it can be considered as an intrinsic property of the specific growth function used to increment the window. On the contrary, for TCP Cubic the growth of the window depends on real time and hence also on link capacities and propagation delays, so that a comparison with the TCP versions indicated above would need special care.

Figure 2 shows the response function for different values of θ, between 100 MSS and 1000 MSS (Maximum Segment Size). The two lines correspond to two different ways to express the window evolution in (8). The dotted line corresponds to a fluid model where the growth of the window is approximated with a continuous function (see [4]). More precisely

$$w_n(w_0) = \left((1-k)\alpha n + w_0^{*1-k}\right)^{\frac{1}{1-k}} .$$

The solid line, instead, considers only integer values of the window and computes the increment of the window as according to (1) with $w_{c_0} = w_0$. In this case the last round trip time of the cycle -the n-th one- is evaluated as $n = \min\{i : w_i \geq 2w_0\}$. In both cases we have considered w_0 as the independent value. For each value of w_0 we have then computed S, the total amount of packets sent during a cycle starting with $w = w_0$ and then we have plotted the average window as a function of $1/S = p$. Clearly the total number of packets sent during such a cycle is a monotonically increasing function of w_0 (as the TCP window is monotonically increasing during each cycle) so that increasing values of w_0 correspond to increasing values of S, and decreasing values of $p = 1/S$.

Regarding the integer approximation we observe that, for $\alpha = 1/8$ and $k = 3/4$ (values suggested in [21]), $\alpha w^k \geq 2$ only if $w > 30$. That is the CTCP window grows by one each round trip, the same as Reno, as long as $w < 30$. This is somewhat consistent with what suggested in [21] where the authors call for the delay component to be used (that is to increment the window by αw^k) only if the window is larger than *lowwnd* which they set equal to 41. This observation explains why for small values of w the fluid and integer approximation are different, with the integer approximation giving a larger average value of the window.

In the case of the integer approximation and for large drop rates (more than 10^{-3}) the response function is the same as Reno. The same is true, regardless of the approximation, for small drop rates (less than 10^{-6}) given that, in this latter case, the evolution of the window is the same in CTCP and Reno. For drop rates roughly between $4 \cdot 10^{-4}$ and 10^{-3} only the rapid increase phase of CTCP is used so that the response function is steeper. For drop rates between 10^{-6} and $4 \cdot 10^{-4}$ the "constant" window phase is used along with all the other phases and this explains why the response function increases more slowly until it reaches the Reno response function.

While for Reno and HighSpeed TCP the response function is only a function of the drop rate, for CTCP the response function is also a function of θ (the value of the window during the constant window phase). The larger the value of θ the longer the rapid increase phase can be. In the extreme case of $\theta = \infty$ the other phases would not take place at all and the response function would be much steeper. Note that in [21] the authors compute the response function

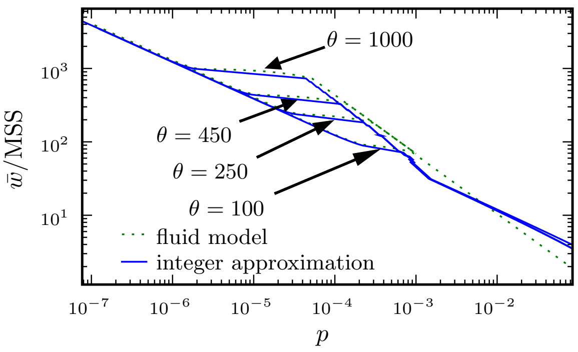

FIG. 2 – The deterministic response function for $\theta = 100, 250, 450, 1000$

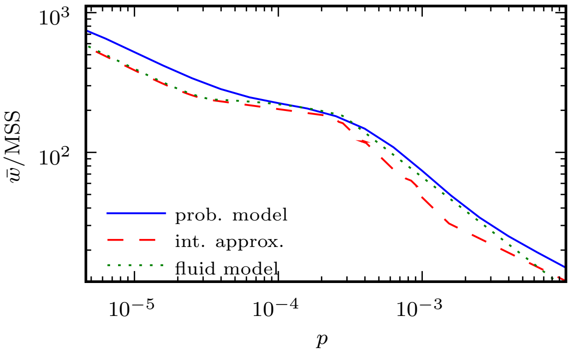

FIG. 3 – CTCP response function

for exactly this case ($\theta = \infty$). Given that they are interested in limiting the aggressiveness of CTCP, they are in effect considering the worst case, by only considering the rapid increase phase. At the same time it can be argued that, as a consequence, CTCP is *less* aggressive than HighSpeed TCP as, for small values of p, CTCP is as aggressive Reno, which is less aggressive than HighSpeed TCP.

Finally it is worth noting that, as p decreases, the difference between the fluid and integer models for the rapid increase phase becomes negligible.

4 Numerical Results

Using the models presented in the previous sections we can compute the average throughput and the average window size for different values of the drop probability p. Figure 3 shows the CTCP response function computed using the two deterministic models presented in section 3.3 and the probabilistic model discussed in section 3.1, for $\theta = 250\,$MSS. For the same "drop probability" the deterministic model with integer approximations gives a smaller average window than the probabilistic model, which uses the very same integer approximations, as already observed in [1]. The fluid model, instead, does agree with the proba-

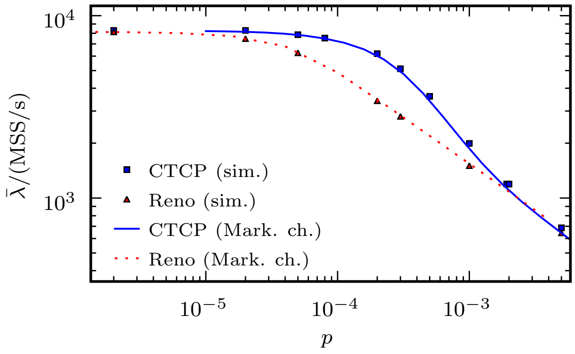

Fig. 4 – Average Throughput ($\theta = 250\,\text{MSS}$)

bilistic model, for some values of p, but we believe that this is just a coincidence. One should take some care in comparing these two models : in the case of the deterministic model the buffer size at the bottleneck is fixed (so that all the cycles have the same size) while in the case of the probabilistic model the buffer size at the bottleneck link is much larger (in theory infinite, set to 1600 MSS for the numerical results) and allows the window to reach larger values. If we used the same buffer size in both models the average window would be smaller in the probabilistic case.

Figure 4 shows the average throughput for CTCP and TCP Reno. For CTCP we have used the probabilistic model introduced in section 3.1 while for Reno we have used an equivalent model but with a “one dimensional” Markov chain as Reno does not have two components in the congestion window. The squares and triangles in Figure 4 correspond to simulation results. For both versions of TCP there is a good match between the probabilistic model and the simulations, obtained using ns-2 with a Linux implementation of CTCP [2]. In the simulations there is a single TCP connection with no cross traffic going through a bottleneck of 100 Mb/s with propagation delay of 26.4 ms and where each packet of 1500 B is dropped with probability p. As the difference between different simulation runs is very small (less than 1%) we did not plot errorbars.

In Figure 4 $w_{\max} = 370\,\text{MSS}$, the bandwidth-delay-product is 220 MSS and the buffer size is 150 MSS. While in Figure 3 $w_{\max} = 1600\,\text{MSS}$ this explains why in Figure 4 the throughput is constant for small drop probabilities (most of the packets are dropped when $w = w_{\max}$) a similar behavior takes place for larger values of $w_{\max}$ but for drop probabilities smaller than those included in Figure 3.

Figures 5, 6, 7 and 8 show the distributions for w_0, w_{c_0} and w_{d_0} when $\theta = 430\,\text{MSS}$ for $p = 3 \cdot 10^{-3}$, $3 \cdot 10^{-4}$, $4 \cdot 10^{-5}$ and $5 \cdot 10^{-6}$ respectively. Corresponding to the four different regions of the response function in Figure 2. In each case the dotted lines represent the corresponding distribution obtained from ns-2 simulations.

Figure 5 corresponds to the case where, most of the time, the window is smaller than 41 so that CTCP behaves similarly to Reno. This is confirmed by the fact that $w_{d_0} = 0$ most of the time.

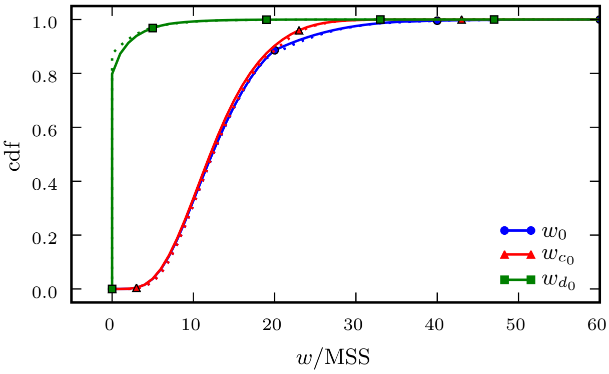

FIG. 5 – Distribution of w_0,w_{c_0} and w_{d_0} ($\theta = 430\,\text{MSS}, p = 3 \cdot 10^{-3}$)

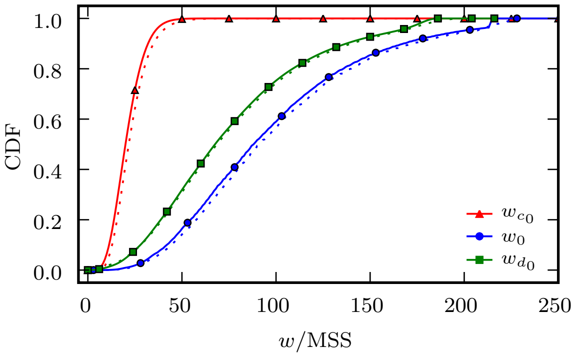

FIG. 6 – Distribution of w_0,w_{c_0} and w_{d_0} ($\theta = 430\,\text{MSS}, p = 3 \cdot 10^{-4}$)

Figure 6 corresponds to the case where most of the packet are dropped when the window is quickly growing as $w_{n+1} = w_n + \alpha w_n^k$. In this case the distribution of w_{c_0} is greater than the distribution of w_{d_0}, so that, on average, $w_{c_0} < w_{d_0}$.

Figure 7 corresponds to the case where most of the packets are dropped during the constant window phase as confirmed by the jump in the distribution of w_0 when $w_0 = 215\,\text{MSS}$ which correspond to $\theta/2$ (as $\theta = 430\,\text{MSS}$ in this case). The smaller jumps in the distribution for the ns-2 simulations are caused by the oscillations of the window during the constant window phase. These oscillations are not taken into account by the probabilistic model.

Figure 8 corresponds to the case where a significant fraction of packets is dropped during the Reno phase (almost 50%). Again the jump at $w_0 = 215\,\text{MSS}$ represents the packets dropped during the constant window phase (with the simulations having smaller jumps caused by the oscillations of the window). The jump at $w = 300\,\text{MSS}$ correspond to the case when packets are dropped due to a buffer overflow (in this case $w_{\max} = 600\,\text{MSS}$ and the buffer size is $200\,\text{MSS}$ for the simulations). In this case the distribution of w_{c_0} is smaller than the distribution of w_{d_0}, the opposite of what happens in the previous two cases.

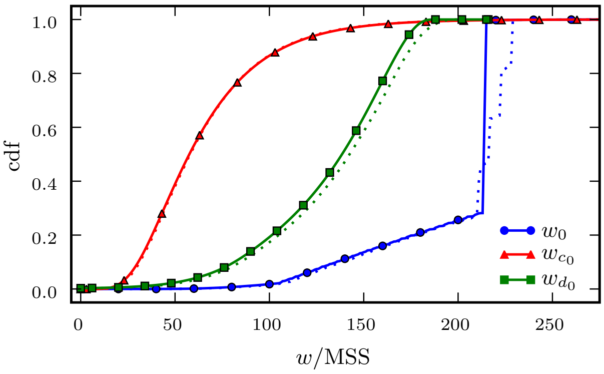

FIG. 7 – Distribution of w_0,w_{c_0} and w_{d_0} $(\theta = 430\,\mathrm{MSS}, p = 4 \cdot 10^{-5})$

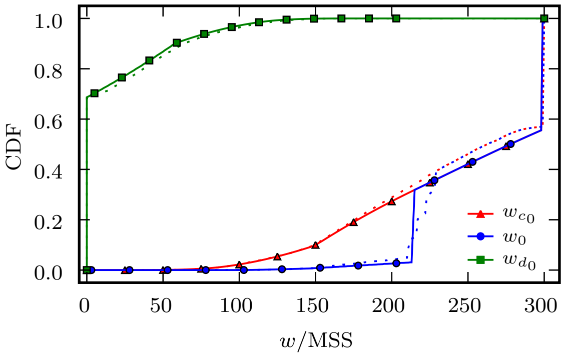

FIG. 8 – Distribution of w_0,w_{c_0} and w_{d_0} $(\theta = 430\,\mathrm{MSS}, p = 5 \cdot 10^{-6})$

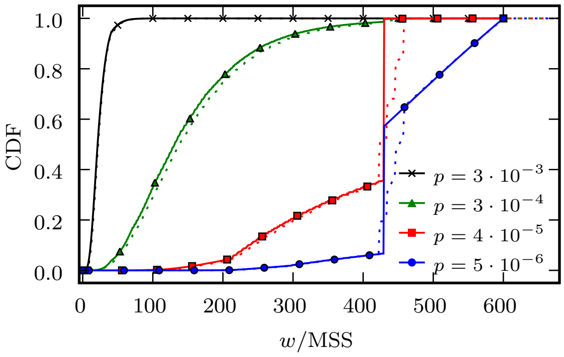

FIG. 9 – Steady state distribution of Y_n $(\theta = 430\,\mathrm{MSS})$, $w_{\max} = 600\,\mathrm{MSS}$

Figure 9 shows the steady state distribution of Y_n for different values of p with $\theta = 430\,\mathrm{MSS}$ and $w_{\max} = 2000\,\mathrm{MSS}$. Again the dotted lines represent

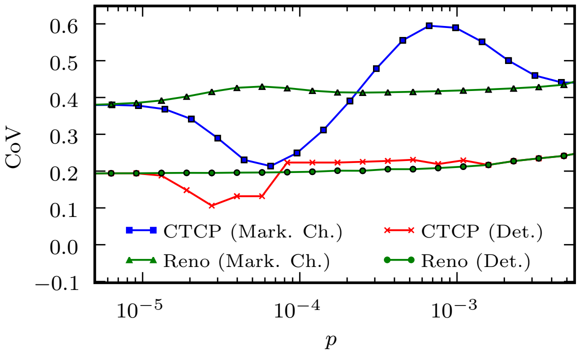

FIG. 10 – Coefficient of variation of Y_n for CTCP and Reno

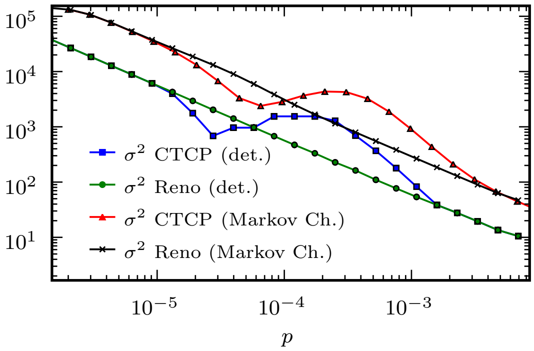

FIG. 11 – Variance of Y_n for CTCP and Reno

the same distribution for the corresponding ns-2 simulations. As expected, with increasing drop probabilities, each distribution is strictly greater than all the previous ones. In Figures 8 and 9 while the probabilistic models have a sharp jump for $w = 215\,\mathrm{MSS}$ and $w = 430\,\mathrm{MSS}$ the simulations (dotted lines) have smaller jumps. This is caused the oscillations of the sending window during phase 2 in the simulations. As mentioned in section 2 this is consistent with the CTCP algorithm but it is not taken into account by the probabilistic model. While it is, at least in principle, possible to incorporate this aspect into the model, we prefer using a simpler model with a constant value during phase 2 given that the differences between this simplified model and the simulations are not significant (especially as far as the throughput is concerned).

Figure 12 shows the average value and the standard deviation of Y_n for CTCP.

Figures 11 and 10 show the variance and the coefficient of variations (CoV) for CTCP and Reno. In both cases the difference between the deterministic and probabilistic model is more pronounced than in the case of the average window (response function). This can be explained by the fact that the average window

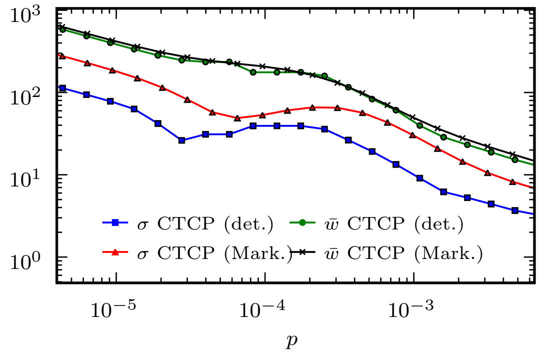

FIG. 12 – Standard deviation and average of Y_n for CTCP

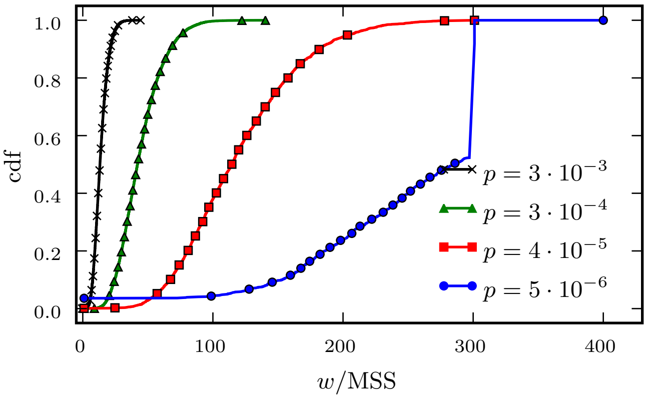

FIG. 13 – Distribution of w_0 for TCP Reno ($\theta = 430\,\text{MSS}$)

depends only the first moment of Y_n while the variance and the CoV depend on the second moment as well. For the CoV, in particular, the difference between the two models is significant. For small values of p the CoV of CTCP is smaller than Reno but for larger values of p the opposite is true indicating that for $p > 10^{-4}$ CTCP might not be the best solution.

Figures 13 and 14 show the distribution (from ns-2 simulations) of w_0 and w_t for TCP Reno under the same conditions : $\theta = 430\,\text{MSS}$ and buffer size $200\,\text{MSS}$.

5 Conclusions and Future Work

In this paper we have presented a Markovian model of CTCP under random losses. This kind of model is a first attempt to roughly assess the impact of varying network conditions on a CTCP connection. The network is seen as a black box randomly dropping packets, due to buffer overflows. This model could

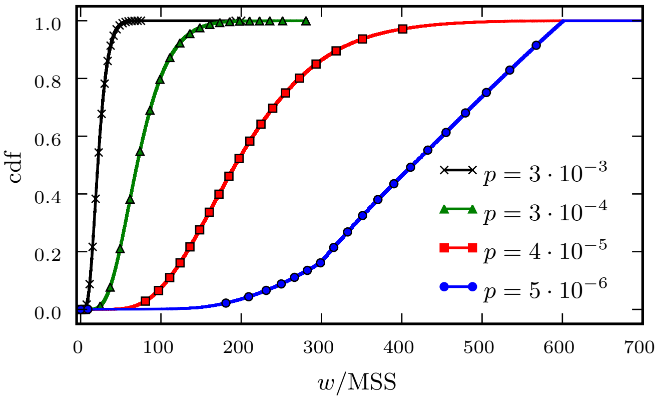

FIG. 14 – Distribution of w for TCP Reno ($\mu\tilde{\tau} = 400\,\mathrm{MSS}$)

also be used to describe the impact of transmission errors in some "challenging environments" (e.g. wireless networks) as expected from new TCP versions [11].

In this first analysis, we have assumed that the loss arrivals follow a simple Bernoulli process. We have computed the distribution of the sending window on loss events with a Markovian model, and then the average throughput. Using Palm Calculus we have computed the steady state distribution of the window. Its value has a direct influence on the buffer occupancy and on the jitter experienced by all the flows sharing the same bottleneck link.

This analysis can be extended in many ways. The Bernoulli loss process could be replaced with a more bursty and realistic process. In this case, multiple losses could take place during the same round-trip time, and the recovery time could be longer. We could also consider time-outs in the case of high loss rates, as in [16]. A similar analytical study could also be applied to the other TCP versions, currently under standardization, comparing their efficiency and their robustness.

Références

[1] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of tcp/ip with stationary random losses. *IEEE/ACM Trans. Netw.*, 13(2) :356–369, 2005.

[2] L. Andrew. Compound TCP Linux module. available at `http://netlab.caltech.edu/lachlan/ctcp/`, April 2008.

[3] A. Blanc, K. Avrachenkov, D. Collange, and G. Neglia. Compound tcp with random losses. INRIA Research Report 6736, August 2008.

[4] A. Blanc, D. Collange, and K. Avrachenkov. Modelling an isolated Compound TCP connection. Tech. Report DE/DIAM/08.65, France Telecom, July 2008. avaliable at `http://www-sop.inria.fr/maestro/personnel/Alberto.Blanc/bca-diam-865.pdf`.

[5] A. Blanc, D. Collange, and K. Avrachenkov. Oscillations of the sending window in Compound TCP. In *Proc. 2nd NetCoop Workshop*, 2008.

[6] Jean-Yves Le Boudec. Understanding the simulation of mobility models with palm calculus. *Perform. Eval.*, 64(2) :126–147, 2007.

[7] L. S. Brakmo and L. L. Peterson. Tcp vegas : end to end congestion avoidance on a global internet. *IEEE JSAC*, 13(8) :1465–1480, 1995.

[8] J. Davies. Performance enhancements in the next generation TCP/IP stack. The Cable Guy `http://www.microsoft.com/technet/community/columns/cableguy/cg1105.mspx`, 2007.

[9] Fiber to the home council. `http://www.ftthcouncil.org/`.

[10] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[11] S. Floyd and M. Allman. Specifying New Congestion Control Algorithms. RFC 5033 (Best Current Practice), August 2007.

[12] A. Kherani, B. Prabhu, K. Avrachenkov, and E. Altman. Comparative study of different adaptive window protocols. *Telecomm. Systems*, 30(4) :321–350, 2005.

[13] T.V. Lakshman and U. Madhow. The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *Networking, IEEE/ACM Transactions on*, 5(3) :336–350, Jun 1997.

[14] Y.T. Li, D. Leith, and R. Shorten. Experimental evaluation of tcp protocols for high-speed networks. *IEEE/ACM Trans. Netw.*, 15(5) :1109–1122, 2007.

[15] Y.T. Li, D.J. Leith, and B. Even. Evaluating the performance of TCP stacks for high-speed networks. In *Proc. 4th Int. Workshop on Protocols for FAST Long-Distance Networks*, February 2006.

[16] J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose. Modeling tcp reno performance : a simple model and its empirical validation. *Networking, IEEE/ACM Transactions on*, 8(2) :133–145, Apr 2000.

[17] D. Sorensen, R. Lehoucq, C. Yang, and Maschhoff K. ARPACK. available at `http://www.caam.rice.edu/software/ARPACK/`.

[18] M. Sridharan, K. Tan, D. Bansal, and D. Thaler. Compound TCP : A new TCP congestion control for high-speed and long distance networks. Internet draft, Internet Engineering Task Force, October 2007. (Work in progress).

[19] K. Tan, J. Song, M. Sridharan, and C.Y. Ho. CTCP-TUBE : Improving TCP-friendliness over low-buffered network links. In *Proc. 6th Int. Workshop on Protocols for FAST Long-Distance Networks*, March 2008.

[20] K. Tan, J. Song, Q. Zhang, and M. Sridharan. Compound TCP : A scalable and TCP-friendly congestion control for high-speed networks. In *Proc. 4th Int. Workshop on Protocols for FAST Long-Distance Networks*, March 2006.

[21] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. In *INFOCOM*, 2006.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

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