Comparing Some High Speed TCP Versions under Bernoulli Losses

Alberto Blanc, Konstantin Avrachenkov
I.N.R.I.A.
2004 route des lucioles
Sophia Antipolis, France
alberto.blanc@sophia.inria.fr
k.avrachenkov@sophia.inria.fr

Denis Collange
Orange Labs
905 rue Albert Einstein
Sophia Antipolis, France
denis.collange@orange-ftgroup.com

ABSTRACT

We compare the performance of Cubic, Compound TCP, HighSpeed TCP and Reno under a simple loss model, where each packet is dropped with probability p. Modeling the evolution of the congestion window with a Markov chain, we can use efficient numerical algorithms to compute the average window size (response function), the Coefficient of Variation (CoV) of the window and the average throughput. We find that, for smaller bandwidth delay products, Cubic can have a similar throughput to Reno while for larger values the throughput of all new versions is similar and larger than Reno. The CoV of Cubic has a peak but it is otherwise the smallest one. This peak corresponds with a sharp increase in the response function.

1. INTRODUCTION

Throughout the years many studies have analyzed the performance of TCP in different scenarios. Several of these studies ([6, 8, 1] just to name a few) analyze the impact of random losses. In the simplest form these losses are modeled as a Bernoulli process: each packet is dropped with probability p and is independent of all the others. Clearly this loss model is not very sophisticated and it is not necessarily the most realistic one. Nonetheless it has been widely used in the literature and, at least for TCP Reno, it does correctly predict the performance in some cases (see, for example [8, 1]). Whether the same holds true for other versions of TCP is currently an open question, to the best of our knowledge. At the same time, thanks to its simplicity, we think that it is a reasonable first step in analyzing the behavior of some of the new versions of TCP.

In this work we are going to analyze different TCP versions under this loss model. In each case it is possible to use a Markov chain to model the evolution of the congestion window immediately after a packet loss. Given that in a real network the congestion window is always bounded by the bandwidth-delay product plus the buffer size, it is realistic to impose an upper bound on the window size. Furthermore, as the window size is usually an integer, we can use a discrete Markov chain with a finite number of states. Such a chain can easily be analyzed with numerical algorithms even for fairly large values of the maximum window size (that is the number of states).

Using such a numerical approach has several advantages over simulations. First of all, at least for smaller values of the maximum window size, roughly less than 2000, finding the steady state distribution of the Markov chain is much faster than running simulations. Second, using the steady state distribution of the Markov chain, one does not have to deal with convergence issues. With simulations it is not always easy to establish whether steady state has been reached, especially if it has to be done in an automatic way. As a fair comparison between simulations and the Markov chain approach would require to find the shortest simulation time needed to reach steady state we do not explicitly compare the running time of the two approaches. While transient phase and its duration can have significant practical consequences we believe that it is also interesting to study the steady state behavior, which should have the greatest influence on the performance of long-lived connections.

On the downside the Markov chain we have used only models the evolution of the window if no timeouts occur. But, as we are mainly interested in networks with large bandwidth delay products, on average the window will be fairly large, reducing the probability of timeouts. Also, the computation time grows as the maximum windows size increases and it depends on the protocol used. For example Reno and Compound TCP require significantly more time than Cubic and HighSpeed for values of the maximum window size around 6000. As both these protocols increment the window by one (at least in some cases), the transition matrix has more transitions than for HighSpeed and Cubic, increasing the time required to compute it.

Clearly such a method cannot replace simulations but we think that it can be a useful tool. Especially for analyzing a wide range of conditions (like large intervals of the drop probability) in order to establish the general behavior of a certain protocol and potentially highlighting interesting features that can be then analyzed, in greater details, with simulations. We have used simulations ourselves in order to check the results obtained with the Markov chain.

Building on our previous work about Compound TCP [2] we extended it to include Cubic and HighSpeed TCP, as well as Reno, used as a reference. While many other versions have been proposed, like Fast TCP [12], H-TCP [9], Illinois [7], and Scalable [5] just to name a few. We have chosen Cubic and Compound because they are used by default, respectively, in Linux and Windows Vista Server (Compound is also included in Windows Vista client but it is not activated by default). HighSpeed TCP is used in some commercial TCP accelerators, like those produced by Riverbed. Because of this we think these versions should have the largest number of users and, as such, are of interest to network operators, who, typically, do not have control over most of the end systems but deal directly with the traffic, and congestion, that they generate.

2. SYSTEM DESCRIPTION AND SOLUTION METHODOLOGY

We consider a single, long-lived, TCP connection going through a single bottleneck link, of capacity μ, where each packet has the same probability (p) of being dropped. Let $\tilde{\tau}$ be the Round Trip Time when the queue is empty (that is it is the sum of all the propagation and processing delays). As previously mentioned we are only considering integer values of the congestion window w, corresponding to the number of MSS (Maximum Segment Size) and we also assume that w is upper bounded by $w_{\max}$. This upper bound can represent either the sum of the Round Trip Time and the buffer size or the advertised window, whichever is smaller. In the remainder of the paper we are going to consider two different settings for $w_{\max}$: one with “small” values ($w_{\max} = \mu\tilde{\tau}+150$), corresponding to a buffer size of 150 MSS, and one with “large” values ($5000\,\text{MSS} \leq w_{\max} \leq 7000\,\text{MSS}$). The idea behind large values is that we would like to know what happens when $w_{\max} \to \infty$, which is often the case considered when talking about the TCP response function. As we consider bandwidth delay products between 220 MSS and 1600 MSS, a value of $w_{\max}$ around 6000 MSS is not a very good approximation of infinity but, as we will discuss later, it is enough to highlight some interesting features of the protocols under discussion.

As previously mentioned, we use a discrete Markov chain to model the evolution of the congestion window immediately after a packet loss. Formally, let X_n be the value of sending window after the n-th packet loss. Clearly for each different TCP version the evolution of X_n will be different but, in each case, it is possible to construct the transition matrix of the Markov chain. To see why this is true consider that, if we know the value of the window after the last packet drop, we can construct the evolution of the window until the next packet drop, as it only depends on how the window is increased. As each packet is dropped with probability p we can compute the probability that there will be n round trip times before the next packet is dropped for any $n \geq 1$.

As in [2], after we have computed the transition matrix for X_n we use the ARPACK implementation of the Arnoldi method to compute the steady state distribution of the Markov chain X_n. From this it is possible to compute the average window size, the throughput and the Coefficient of Variation (CoV), see [2] for more details. The only difference with [2] is that, in this work, we always measure time in seconds and not in Round Trip Times. This is because Cubic does not increment its window based on the number of round trip times but on the number of seconds elapsed since the last packet drop. The coefficient of variation is linked to the jitter experienced by the connections sharing the same bottleneck link, as long as this model is applicable to more general settings with multiple connections.

We have verified a subset of our results using ns-2 in the same setting, that is with a single connection with no other traffic and a constant packet drop probability p. The main conclusion is that our model captures correctly the main mechanisms of the protocols. Due to space constraints and to the fact that the simulations share the same hypothesis with the model we do not present these results.

As the Compound TCP sending window has two components (congestion and delay) if one wants to model the evolution of both components the number of states of the Markov chain X_n increases significantly. One possible approach is to use a bi-dimensional Markov chain as we have done in [2]. Another possible solution is to simplify the model for Compound TCP and assume that the delay component (w_d) is always 0 after a packet loss. This assumption does introduce an error given that the length of the “constant window” phase of Compound TCP cannot be correctly computed. At the same time this could have an impact only when there is a significant number of packet drops during the constant window phase: if most of the packets are dropped before the constant window is reached, the value of w_d is uninfluential and if most of the packets are dropped during the “Reno” phase (that is when the window is increased by one each Round Trip Time) w_d is indeed zero after a packet drop. Comparing the results obtained with this approach to those obtained with the bi-dimensional chain in [2] we have concluded that the difference between the two approaches is negligible. Even in the cases when there is a significant number of packet drops during the constant window phase the average window size is basically the same using the two different Markov chains. This apparent independence of the average window size from the length of the constant window phase could be a consequence of our (simple) loss model; but it might be worthwhile to investigate this further, as it could lead to a simplified version of Compound TCP.

For Compound TCP we assume that the sender has a perfect estimate of the queue size so that the value of the window during the “constant window” phase is exactly $\theta \triangleq \mu\tilde{\tau} + \gamma$ (as suggested in [11] we set $\gamma = 30\,\text{MSS}$). To the best of our knowledge there are no definitive studies on the robustness of the queue size estimate used by Compound, which is the same estimate used by Vegas. While it is possible to construct cases in which this estimate performs poorly it is not clear how frequent they are in reality. At the same time one could argue that the Bernoulli losses unduly penalize Cubic as this protocol uses the value of the window at the last packet loss as the target value for the window. (Recall that Cubic puts the inflection point of the cubic function at this value, so that the window is quickly increased until this value and then its growth rate is reduced until the max-probing phase.) With Bernoulli losses such target value is random with a potentially negative impact on Cubic's performance. While this is an unfortunate aspect of the loss model, whenever one has to compare multiple protocols it is not always easy to find a scenario which does not penalize, or favor, any of the protocols. Comparing different TCP versions is not an easy task and this study examines only one specific case and it is by no means the definitive answer. Even though it does have some limitations we think that such a case is interesting and informative about the behavior of the protocols considered.

One last comment about the Markov chain X_n is that we do not model the TUBE algorithm for Compound [10] nor Fast Convergence for Cubic. While it is possible to model both by modifying the Markov chain this would lead to a significant increase in the number of states, and hence solution time. In both cases, given the random losses, we think that the influence of these algorithms is negligible. We have run a limited number of simulations with and without

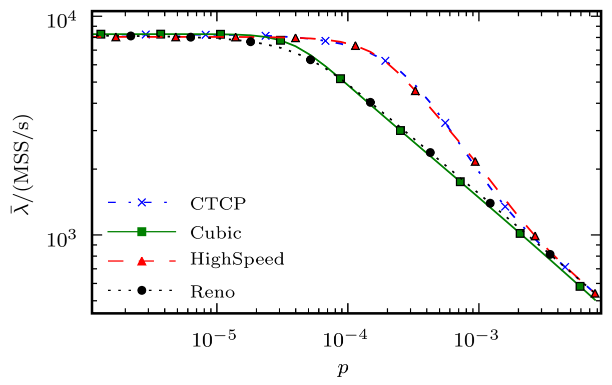

Figure 1: Average throughput for $\mu\tilde{\tau} = 220\,\mathrm{MSS}$

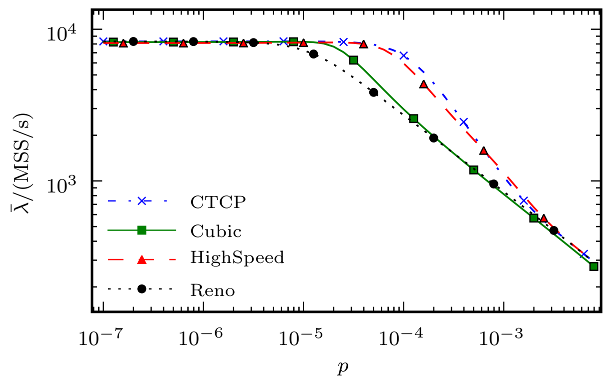

Figure 2: Average throughput for $\mu\tilde{\tau} = 800\,\mathrm{MSS}$

these algorithms, without changing anything else, and the steady state distribution of X_n did not change in any of them.

3. NUMERICAL RESULTS

3.1 Average Throughput

We start by looking at the efficiency of the different protocols. Figures 1-3 show the average throughput for different values of the bandwidth delay product. In all cases the bottleneck capacity is 100 Mbit/s which is equivalent to 8333 MSS/s with packets of 1500 B and $w_{\max} = \mu\tilde{\tau} + 150\,\mathrm{MSS}$. The Round Trip Time is 26.4 ms, 96 ms and 192 ms, respectively. In each case the maximum throughput (i.e. the capacity of the link) is achieved for sufficiently small drop probabilities. Eventually, as p increases, the throughput decreases for all protocols, but this transition does not take place for all the protocols at the same time. For example, for $\mu\tilde{\tau} = 220\,\mathrm{MSS}$ Compound and HighSpeed achieve a higher throughput, that is they are more efficient, than Cubic and Reno for larger values of p (in this case there is almost no difference between Cubic and Reno). When $\mu\tilde{\tau} = 800\,\mathrm{MSS}$ Cubic is still fairly close to Reno, while for $\mu\tilde{\tau} = 1600\,\mathrm{MSS}$ it is very close to the other new versions. Recall that, as the rate at which Cubic increases its window depends on real time, these results are a function of the Round Trip Time and not only of the bandwidth-delay product.

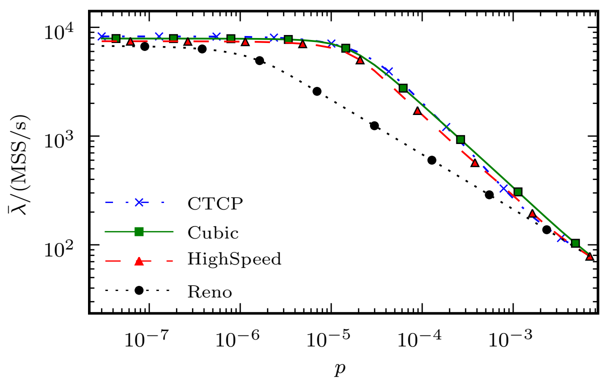

Figure 3: Average throughput for $\mu\tilde{\tau} = 1600\,\mathrm{MSS}$

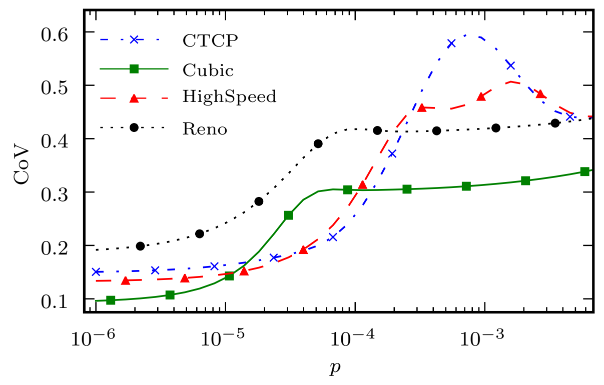

Figure 4: The coefficient of variation for $\mu\tilde{\tau} = 220\,\mathrm{MSS}$

For $p > 10^{-3}$ all the protocols behave in a similar way, this is not too surprising given that Compound and HighSpeed are designed to behave like Reno for small values of the window. Similarly Cubic increases its window at least by the same amount by which a Reno connection in the same circumstances would increase its window. Note that it has been reported in [4] that the end to end drop rate experience by ADSL customers is normally between 10^{-4} and 10^{-3}.

3.2 Coefficient of Variation of the Congestion Window

The Coefficient of Variation (CoV) is defined as the ratio between the standard deviation and the mean and it is related to the jitter experienced by the connection in question as well as by other flows sharing the same bottleneck link. Figures 4-6 show the CoV for the same settings used in the previous section (3.1). When $\mu\tilde{\tau} = 220\,\mathrm{MSS}$ for small loss rates (roughly less than 10^{-5}) all new versions of TCP have a smaller values than Reno. For larger drop probabilities Cubic has the smallest values while Compound has a peak: this is probably due to the fact that Cubic adapts its target value for the window (the inflection point of the cubic function) while Compound keeps quickly increasing the window without reaching its target value of $\mu\tilde{\tau} + \gamma$ (as suggested in [11] we set $\gamma = 30$) causing larger oscillations of the window.

For larger bandwidth delay products, the general shapes of the curves does not change much and they are all shifted

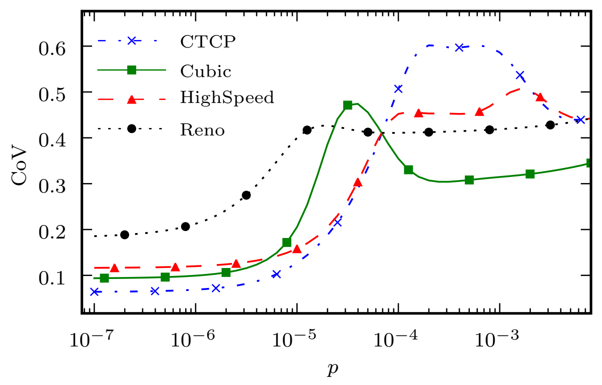

Figure 5: The coefficient of variation for $\mu\tilde{\tau}$ = 800 MSS

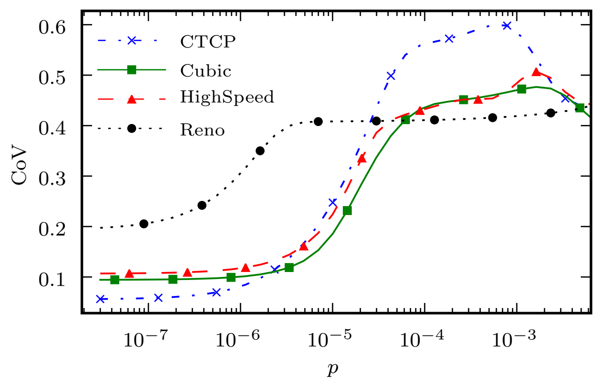

Figure 6: The coefficient of variation for $\mu\tilde{\tau}$ = 1600 MSS

to the left. The exception is Cubic, that has a peak around $p = 3 \cdot 10^{-5}$ when $\mu\tilde{\tau} = 800$ MSS. (We will discuss this in section 3.4.)

3.3 Impact of $w_{\max}$ on the Average Window Size and CoV

In the previous two sections we have used $w_{\max} = \mu\tilde{\tau} + 150$ MSS. In this section we investigate the influence of larger values for this parameter. Figures 7 and 8 show the average window size for $w_{\max} = 950$ MSS and 6000 MSS, respectively. In both cases the total propagation delay is 96 ms and the capacity is 8333 MSS/s ($\mu\tilde{\tau} = 800$ MSS). Figure 8 shows how HighSpeed and Cubic reach $w_{\max}$ for larger drop probabilities than Compound and Reno. As previously discussed, this is probably a consequence of our assumptions: Bernoulli losses can adversely affect Cubic by randomizing the target value for the window while, for Compound, we assume that the sender has a perfect estimate of the queue size so that the window will be equal to $\mu\tilde{\tau} + \gamma$ during the constant window phase. Figure 8 shows, as well, that Cubic has a smaller average window than Compound for larger drop probabilities, but the opposite is true for smaller drop probabilities.

The fact that one protocol has a larger window than another is sometimes rephrased as "one protocol is more aggressive" than the other. In this case we could say that

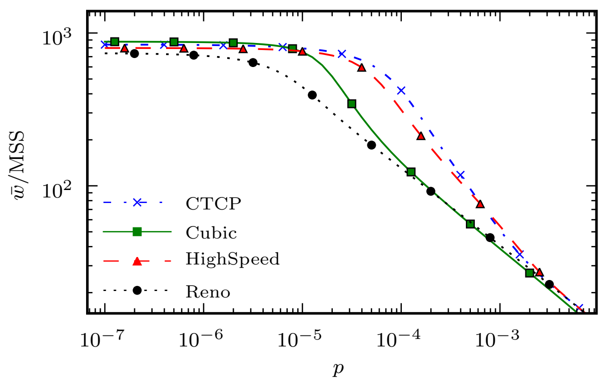

Figure 7: The average window size for $\mu\tilde{\tau} = 800$ MSS with $w_{\max} = 950$ MSS

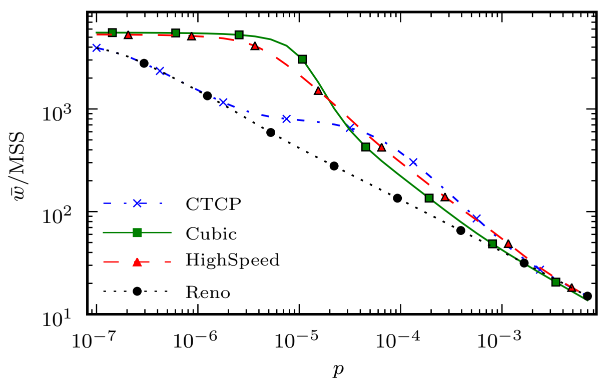

Figure 8: The average window size for $\mu\tilde{\tau} = 800$ MSS with $w_{\max} = 6000$ MSS

for certain drop probabilities Cubic is more aggressive than Compound while the opposite is true for other values of p. It might be worthwhile to point out that, as Cubic grows its window based on real time while Compound uses round trip times, it is not possible to establish which of the two protocols has the larger average window (is more aggressive) without knowing both the bandwidth and the propagation delay. As we will discuss in section 3.4 Cubic response function depends both on the round trip time as well as on the capacity and not only on the bandwidth delay product as for Compound, Reno and HighSpeed.

Some care should be taken when in interpreting this "aggressiveness:" even if two connections share the same bottleneck link it is not necessarily true that they will experience the same drop rate. Therefore it not always appropriate use the response function to predict what happens when two connections share the same link, unless it is possible to compute the drop probability for each protocol.

Figures 9 and 10 show the CoV for $\mu\tilde{\tau} = 220$ MSS and $\mu\tilde{\tau} = 1600$ MSS with $w_{\max} = 5000$ MSS and 7000 MSS respectively. In both cases Cubic has a peak and its CoV is not always the smallest one among all the protocols considered. (We did run several simulations to verify this results and the height and location of the peak did correspond well with the Markov chain results.) It is interesting to compare this with what reported in [3]: using a different loss model

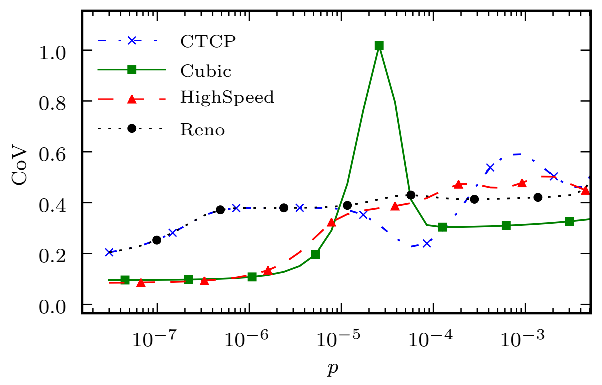

Figure 9: The coefficient of variation for $\mu\tilde{\tau}$ = 220 MSS with $w_{\max}$ = 5000 MSS

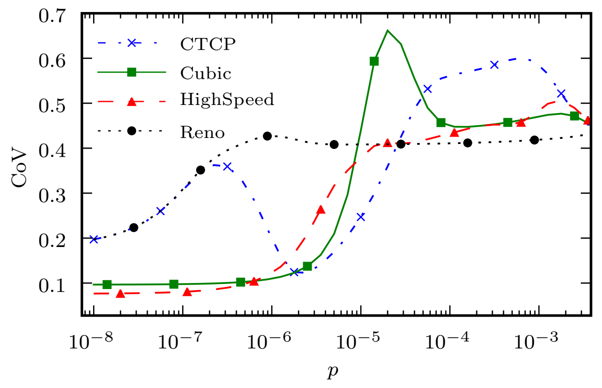

Figure 10: The coefficient of variation for $\mu\tilde{\tau}$ = 1600 MSS with $w_{\max}$ = 7000 MSS

the authors conclude that Cubic has a smaller CoV than other TCP versions. Note that, as we are using a different loss model, these results do not contradict each other.

3.4 Compound and Cubic Response Functions

Traditionally the response function is defined as the average window size as a function of the drop probability, where the average is computed measuring time in Round Trip Times. This definition arose in the context of Reno and other versions of TCP which increment the window once each Round Trip Time. As Cubic increases its window based on real time and not Round Trip Times we decided to always express time in seconds and compute the average window size using seconds and not Round Trip Times. Figure 11 shows the Compound response function for different values of the bandwidth delay product (with $\mu = 100\,\text{Mbit/s}$, $w_{\max} = 5000\,\text{MSS}$ and $\tilde{\tau} = 26.4\,\text{ms}, 96\,\text{ms}, 192\,\text{ms}$). Comparing this figure with those in [2] it looks like that measuring time in seconds or in Round Trip Times does not change the response function significantly and that, for Compound, the response function is only a function of the bandwidth delay product.

For Cubic, instead, the response function depends on both parameters. Figures 12 and 13 show the response function in the case of constant capacity and constant round trip time, respectively. Changing the bandwidth or the delay affects

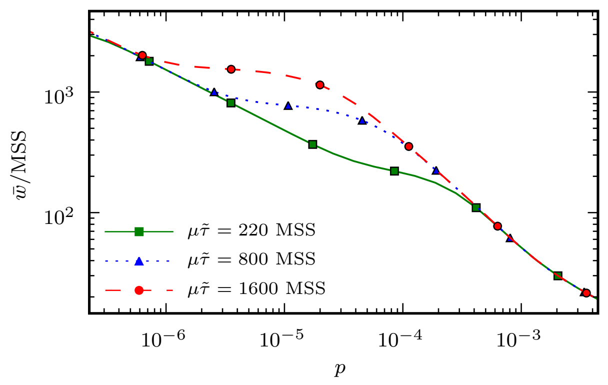

Figure 11: Compound $\bar{w}(t)$ for different values of $\mu\tilde{\tau}$

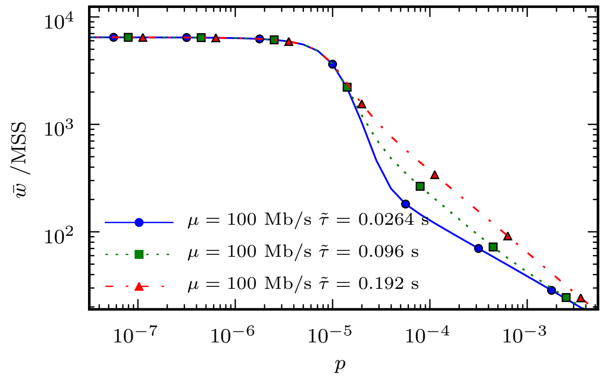

Figure 12: The average window size for μ = 100 Mbit/s, and different values of $\tilde{\tau}$ ($w_{\max}$ = 7000 MSS)

the response function in different ways. Due to space constraints we did not include the figure, but it is possible to have different response functions for the same value of the bandwidth delay product. It is interesting to note that, even if Cubic uses real time to increase its window, the response function does change when the round trip time changes (Figure 12).

In the case of $\mu = 100\,\text{Mbit/s}$ and $\tilde{\tau} = 26.4\,\text{ms}$ the average window size quickly decreases when $p \simeq 10^{-5}$. Figures 14 and 15 show $w(t)$ (from a simulation) for $p = 1.2 \cdot 10^{-5}$ and $p = 2.58 \cdot 10^{-5}$, respectively. For the smaller value of p, the window reaches $w_{\max}$ more often and the average window size is larger as well. This also explains why the CoV has a peak around these values of p. We have not been able to explain why the behavior of Cubic changes so rapidly for these values of the drop probability, nor we can find the value of p for which this transition takes place. At the same time this might be an issue worth investigating further, as it clearly has non-negligible consequence on the average window size and on the CoV.

4. CONCLUSIONS

We have used a Markovian model to compute the throughput, average window size and CoV for different TCP versions, with Bernoulli losses. Such a method is, often, much

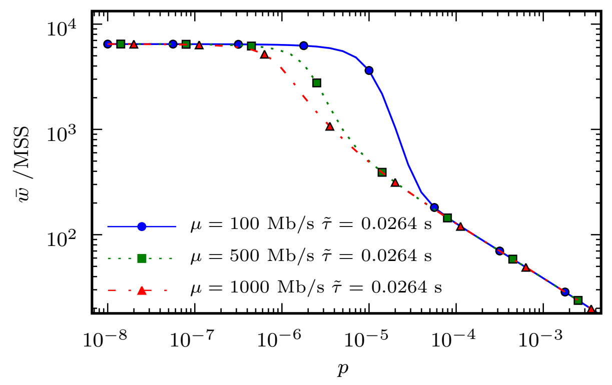

Figure 13: The average window size for $\tilde{\tau} = 0.0264\,\text{s}$, and different values of μ ($w_{\mathbf{max}} = 7000\,\text{MSS}$)

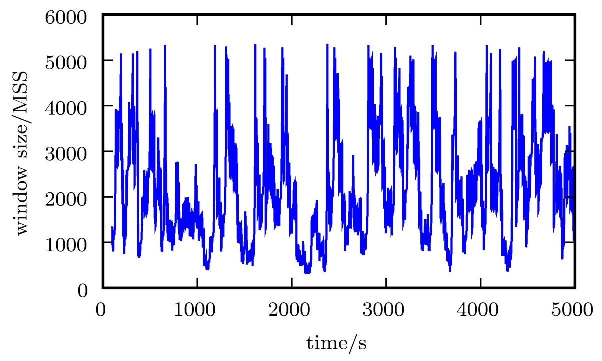

Figure 14: The evolution of $w(t)$ for Cubic with $p = 1.2 \cdot 10^{-5}$

more time efficient than simulations allowing us to cover more scenarios. Among these we were somewhat surprised to find that, in some cases, Cubic behaves similarly to Reno, with a lower throughput than Compound and HighSpeed. Also the CoV of the Cubic window has a peak, which is more pronounced for larger values of the bandwidth delay products and of $w_{\max}$. As the response function of Cubic depends both on the bandwidth as well as the delay (and not only on their product as in the case of Compound, HighSpeed and Reno), it is not possible to say that “Cubic is more aggressive than Compound” or vice-versa. Both cases are possible and the answer also depends on the drop probability.

We believe that some of these aspects could be further investigated but we did not find any glaring problem with any of these protocols, which should be safe to use in most cases. Finally it is worth pointing out that it should be possible to modify the Markovian model in order to consider different loss models. It should be possible to used any loss model for which it is possible to compute the probability of reaching a certain value of the window, knowing only the value of the window right after the last packet drop.

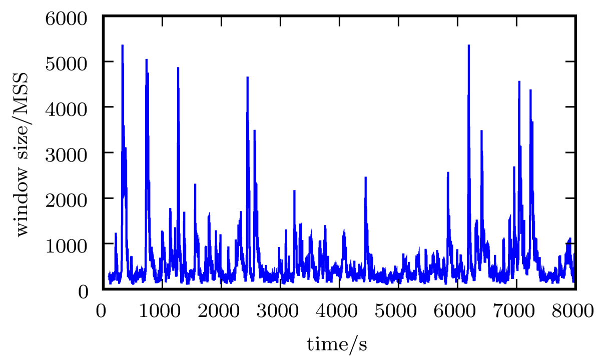

Figure 15: The evolution of $w(t)$ for Cubic with $p = 2.58 \cdot 10^{-5}$

5. REFERENCES

[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] A. Blanc, K. Avrachenkov, D. Collange, and G. Neglia. Compound TCP with random losses. In *IFIP/TC6 NETWORKING 2009*, Aachen, Germany, May 2009 (extended version available as INRIA technical report 7636).

[3] H. Cai, D. Y. Eun, S. Ha, I. Rhee, and L. Xu. Stochastic ordering for internet congestion control and its applications. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 910–918, May 2007.

[4] D. Collange and J.-L. Costeux. Passive estimation of quality of experience. *Journal of Universal Computer Science (j-ucs)*, 14(5):625–641, 2008.

[5] T. Kelly. Scalable TCP: improving performance in highspeed wide area networks. *SIGCOMM Comput. Commun. Rev.*, 33(2):83–91, 2003.

[6] T. 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.

[7] S. Liu, T. Başar, and R. Srikant. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Perform. Eval.*, 2008.

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

[9] R. Shorten and D. Leith. H-TCP: TCP for high-speed and long-distance networks. In *Proc. 4th Int. Workshop on Protocols for FAST Long-Distance Networks*, Feb. 2004.

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

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

[12] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 14(6):1246–1259, 2006.