

ERROR MODELING FOR TCP SIMULATIONS

Kostas Pentikousis, Hussein Badr

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 111794-4400, USA
{kostas, badr}@cs.sunysb.edu

ABSTRACT

This work focuses on the implications for TCP performance with respect to two broad characteristics of error modeling. First, we consider rate-based error models vs. temporal error models. We show that it is difficult to confidently assess the impact of improvements to existing protocols or the performance gains of new protocols under a rate-based model. Second, we consider error models that “synchronize” losses in the forward and reverse paths vs. models that do not do so, or which model losses in the forward path only. Our results show that TCP performance can be dramatically different under a synchronized error model compared to a non-synchronized one.

KEYWORDS

TCP, ERROR MODELING, WIRED & WIRELESS COMMUNICATIONS

1 INTRODUCTION

The Transmission Control Protocol (TCP) (Postel, 1981) has been optimized over the years for wired networks (Allman et al., 1999). With the proliferation of wireless communications, TCP has to perform well in transmission-media heterogeneous infrastructures, which exhibit characteristics different from wired ones, e.g. random segment corruption over wireless links (Lee, 1993; Tabbane, 2000). Different improvement proposals attempt to alleviate TCP’s inefficiencies in a wired-cum-wireless environment (Allman et al., 2000; Bakre & Badrinath, 1995; Balakrishnan et al., 1995; Cáceres & Iftode, 1995; Parsa & Garcia-Luna-Aceves, 1999; Sinha et al., 1999). Testing the comparative efficiency of these proposals requires realistic error models (Zorzi & Rao, 1999; Pentikousis, 2000).

2 TEMPORAL VS. RATE-BASED ERROR MODELS

This work investigates the impact that two broad characteristics of error modeling have in the evaluation of TCP performance. First, we consider rate-based error models vs. temporal error models. Rate-based error models drop a specified proportion of segments. Trivial rate-based error models drop a predetermined set of segments. For example, the error model drops the 4th, 13th, 17th, and so on, incoming segments. Although such an error model does not simulate realistic situations it has nevertheless been used to point out inefficiencies in TCP Congestion Control (Fall & Floyd, 1996). This model can easily be extended to a non-deterministic one by implementing Bernoulli trials with a specified drop probability.

A rather more versatile error model uses a two-state continuous-time Markov chain. As depicted in Figure 1(a), the model alternates between two states (A and B) which can have different dropping rates. One possible configuration is to have state A set with 0% drop rate corresponding to “good” periods where the network does not corrupt packets, and state B set with a non-zero drop rate corre-

sponding to “bad” periods when packets are dropped in the network. The model uses exponential distributions to determine the sojourn times in each state. It can be configured as a Semi-Markovian model in which the sojourn times in a state are sampled from a pre-selected but arbitrary distribution. It is a rate-based model because the decision of whether to drop a packet or not in a given state is predetermined according to a fixed rate. In other words, under a rate-based model it does not matter *when*, exactly, a segment will actually arrive at the receiver: each and every segment undergoes a Bernoulli trial at the specified non-zero drop probability.

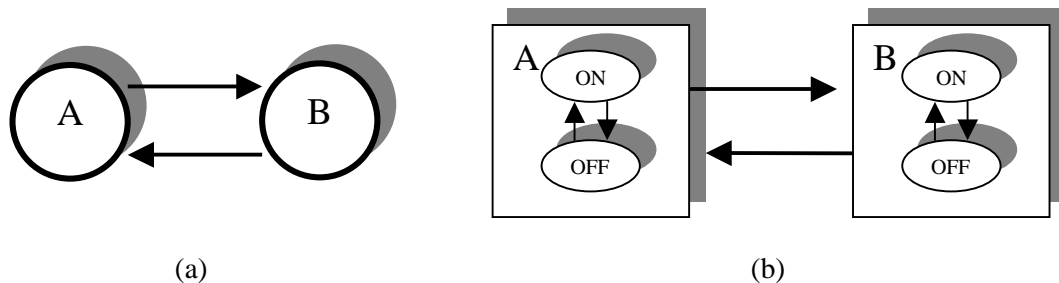


Figure 1. Markovian Error Models:
 (a) two-state continuous time chain; (b) temporal four-state model used in VDDP

Temporal error models, on the other hand, capture the effect of the channel going “bad” for a specified proportion of time, during which transmitted segments will be lost only if they happen to be in transit at that time. The two-state model of Figure 1(a) can be configured as time-based. Here, instead of assigning a drop rate to each state, we designate one as a “good” state where no segments are dropped and the other as “bad” where *all* segments are dropped. The sojourn times in each state can be sampled from arbitrarily chosen distributions. Exponential distributions are often used (Zorzi & Rao, 1999), yielding a Markovian model. This temporal error model, of course, is simply an On/Off model. The capability of such models to capture network performance as perceived by the transport layer is less than satisfactory (Xylomenos & Polyzos, 1999). On the other hand, the two-state rate-based model is not necessarily just an On/Off model, e.g. segments can be dropped partially in either state. For example, if the drop rate in state B is 10%, then 90% of the segments received while the model is in this state will *not* be dropped. In order to have a comparable temporal model we need to allow partial dropping in a state as well.

Figure 1(b) illustrates a temporal four-state Semi-Markovian-based model. The model has two main states, A and B. Each state has two sub-states ON and OFF. While the model is in an OFF sub-state (A OFF or B OFF) no incoming packets are dropped (dropping OFF), whereas during the ON sub-states all incoming packets are dropped. Clearly this model is more versatile than the time-based two-state model of Figure 1(a), and allows partial dropping in each main state. For example, state B can be configured with a mean overall sojourn time of 10 seconds, and a “drop rate” of 10%. When talking about the main states of this model (state A and state B), we use the “drop rate” to signify the proportion of time the model spends in the ON sub-state of the main state, as opposed to the OFF sub-state. This, for example, therefore means that for a drop rate of 10% in state B, the mean sojourn times will be in the ratio of 1:9 for the B ON and B OFF sub-states, respectively.

We used the model in Figure 1(b) to define the Virtual Dropping and Delaying Protocol (VDDP) (Pentikousis, 2000) for the *x*-kernel protocol framework (*x*-kernel, 2000) and for ns-2. VDDP uses exponential distributions for the sojourn times of the main states A and B, and exponentially-based distributions for the sojourn times of the ON/OFF sub-states, as follows. When the model transits from one sub-state of a main state to the other, the sojourn time in the new sub-state is sampled from an exponential distribution. If this sojourn time will cause the total, exponentially-sampled, sojourn time in the main state to be exceeded, the residence time in the sub-state is truncated at the appropriate point. The model “remembers” the sub-state and truncated residual sojourn time. When the model next revisits the main state, it does so by reentering that same sub-state and completing the residual sojourn time there. This residual time is accounted as part of the exponentially-sampled residence time for the new visit into the main state. The model can, of course, easily be modified to use distributions other than the exponential. It aims at providing a flexible structure, which can be calibrated to yield a large range of more realistic error dynamics.

In order to demonstrate the difference between a rate-based and a temporal error model derived from the same Markovian chain, i.e. using the same exponentially-distributed sojourn times for states A and B, respectively, in both models, we conducted a series of tests using the *x*-kernel protocol framework. The rate-based model works as shown in Figure 1(a), whereas VDDP was used as the temporal model. Both models were configured to simulate “good” and “bad” phases on the link. For our testing purposes we created an application-level protocol for *x*-kernel which makes a bulk transfer of 5 MB from one dedicated host to another in a low bandwidth environment (Pentikousis, 2000). TCP Tahoe was used in all experiments. The mean sojourn times for states A and B are set equal to each other, and sampled from the same exponential distribution. The drop rate in state A was 0% for both models, and varied as shown in Figure 2 for state B. Therefore, each model drops segments only in state B. The values in Figure 2 give the total connection times to transfer the 5-MB data set, and are all based on averages of 10 batches (i.e. 10 serial transfers of 5-MB data sets).

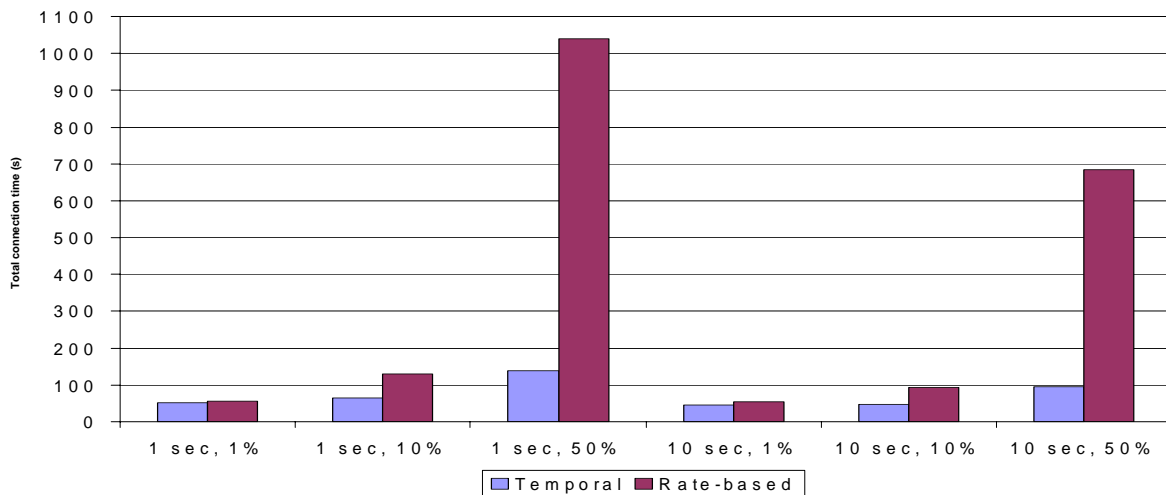


Figure 2. Temporal- vs. rate-based error models

There are three main parameters that affect the performance of TCP in our tests: the mean sojourn times of the error model states, the drop rates, and the type of error modeling (temporal vs. rate-based). TCP Tahoe performs better when the sojourn times are greater, taking advantage of the “good” periods, and effectively shutting down when the channel is corrupting packets. This way the longer the “good” periods, the smaller the total time to complete the 5-MB transfer. Of course, the bigger the drop rates the worse TCP performance is in terms of total connection time.

However, the impact of the type of error modeling on TCP performance is more predominant than that of drop rates and sojourn times. When drop rates are small (1%), the difference in TCP performance under the two types of error models is insignificant. When the drop rates are higher, the impact of the underlying error model on TCP measured performance is significant. When we compare TCP performance under the two models for the same sojourn times and drop rates, TCP Tahoe completes the file transfer faster under the temporal error model. This is consistent across all configurations. For example, for sojourn times of one second and a drop rate of 10%, TCP needs more than double the amount of time to finish the 5-MB transfer. The effect of error modeling is even more striking when the drop rate is high (50%): there is a seven-fold improvement in measured performance under the temporal than under the rate-based model.

Moreover, note that the total time to finish the 5-MB file transfer is almost the same under a rate-based model configured to drop with a probability of 10%, and a temporal model configured to drop with 50% probability, for mean sojourn times of both one and ten seconds. In another set of tests (not shown here), TCP Tahoe finished the transfer in about the same time under the rate-based model with one second mean sojourn times and a 10% drop rate, and under the temporal model with the same sojourn times but 44% drop rate. Effectively, this means that, from the perspective of TCP performance, an overall drop rate of 5% under a rate-based model translates to a channel that is corrupting packets 22% of the time.

TCP attempts to adjust its behavior in response to perceived drops. In particular, much of the on-going research aims at making TCP respond in an intelligent manner to prevailing network conditions. But under a rate-based model, a specified proportion of packets will be dropped irrespective of how the protocol tries to adjust to perceived error conditions. It is therefore difficult, under such a model, to confidently assess the impact of the improvements proposed to existing protocols, or the performance gains of new protocols. As such, temporal models provide a more realistic representation of error occurrences over a real-world network link or end-to-end connection, since error conditions are not modeled to induce a predetermined proportion of drops.

3 THE EFFECT OF SYNCHRONIZATION

We now consider error models that “synchronize” losses in the forward and reverse paths vs. models that do not do so, or which model losses in the forward path only. The forward path is the network path where data are flowing from the sender to the receiver; the reverse path is the network path where acknowledgements (ACKs) flow back from receiver to sender. A TCP connection between two hosts could have the same forward and reverse paths, or could have at least one different link in the two paths. In the former case, whenever the forward path is in the bad state so should the reverse path, and vice versa. In this sense, the error model has to be “synchronized” in both directions. On the other hand, if the forward and reverse paths are different, even if they exhibit the same overall error characteristics, the model should not be synchronized. Since TCP sender behavior is “self-clocking”, largely governed by the pattern of ACKs the sender receives for in-flight data, using a synchronized or non-synchronized error model can have significant performance implications. Furthermore, one of the most widely used network simulation tools, ns-2, has a default error model that yields specified drops in the forward direction only. The reverse path is error-free. This, too, can have significant performance implications. Note that the Bernoulli trials of a rate-based model inherently yield a non-synchronized dropping model.

The original VDDP introduces synchronized drops in both forward and reverse paths. With minor modifications VDDP can be made to yield non-synchronized drops, as well as drops in the forward path only. We used three versions of TCP, namely Tahoe, Reno (Allman et al., 1999), and NewReno (Floyd & Henderson, 1999), for our testing. The tests involve a 8-MB file transfer between two dedicated hosts (Pentikousis, 2000). All three VDDP variations in this set of tests were configured as simple, two-state, Markovian On/Off models: No segments are dropped while in state A; all segments are dropped while in state B. The mean sojourn time is 1.8 seconds for state A and 0.2 seconds for state B. We first experimented with a scenario that involved LAN-scale roundtrip times (RTTs). Table 1 presents the total connection time (averages of 10 batches) to complete the 8-MB data transfer under different synchronization scenarios for the three TCP versions considered.

Table 1. Comparison of On/Off models in a LAN environment.
Entries show total connection time in seconds

	Tahoe	Reno	NewReno
No lost ACKs	71.798	69.800	70.649
Synchronized	72.765	71.704	75.286
Not synchronized	97.277	97.887	100.079

Lack of synchronization in the error model leads to worse results (Table 1). TCP needs approximately 30% more time to finish the transfer under a non-synchronized model than under a synchronized one. Note that there is a slight reordering in terms of relative performance amongst the three TCP versions under the two variations of the error model.

The effect of not losing ACKs on the other hand, is not significant, although it leads to a change in the relative order of performance. For example, NewReno seems to perform better than Tahoe under a synchronized temporal error model that does not drop ACKs. However, when ACKs are lost, NewReno performs worse than Tahoe.

In order to simulate a scenario where the two hosts are communicating over the Internet, we replicated the same tests with simulated latency. We introduced a fixed 50-ms delay to each incoming segment, yielding RTTs in the order of 100 ms. The increased RTT means that the delay \times bandwidth product will be much larger than in the case of a LAN. A larger delay \times bandwidth product implies that the TCP sender's window can become proportionately larger, with more outstanding data than in the LAN scenario discussed above. We made sure that only the congestion window limits the sender's window by making the receiver's advertised window sufficiently large. The same testing configuration was used as for Table 1 above; results are shown in Figure 3(a).

First of all, it is clear that TCP performance suffers more under a non-synchronized model than under a synchronized one. In fact, the total connection time doubles when non-synchronized drops occur, as compared to synchronized drops. In addition, the relative performance of the three TCP versions is significantly different under the two categories of models. NewReno performs better than the other two TCP versions when no ACKs are lost; Reno performs better than the others under a synchronized model which introduces losses in the reverse path, and under a non-synchronized model. This means that the error modeling used in performance evaluation studies can affect the measured TCP performance in a non-uniform fashion, and can lead to different conclusions on the efficiency of different TCP improvement proposals. Results from other sets of tests, not included here for lack of space, further confirm this.

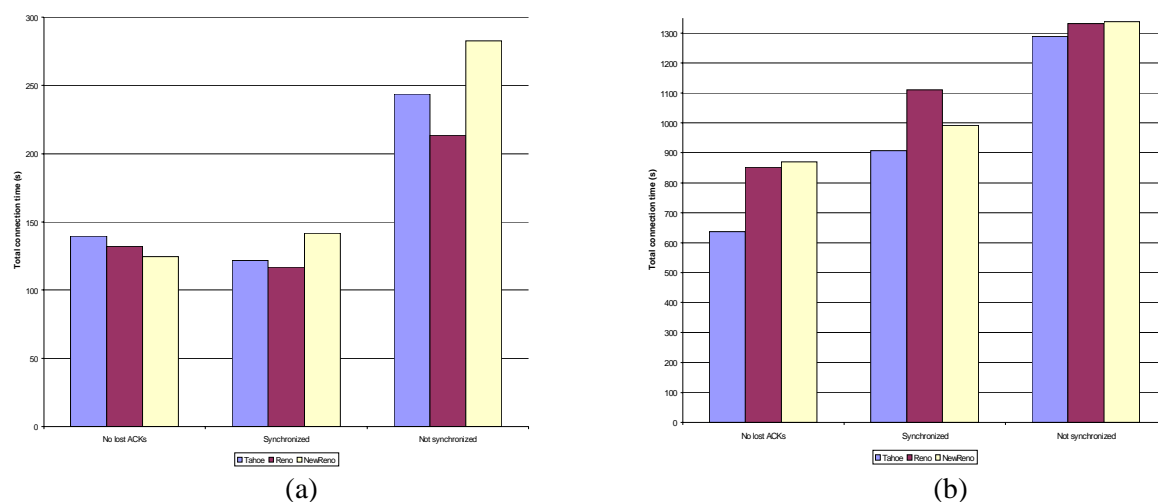


Figure 3. Comparison of On/Off models in (a) an Internet environment, and (b) in an environment with long delays

Finally, we tested the TCP versions under the three synchronization scenarios with even longer latencies (Figure 3(b)). Each incoming segment experienced an added fixed delay of 300 ms, leading to a minimum RTT of 600 ms. The effect of a non-synchronized error model is significant, but not as extreme as in the previous case. The long delays do not allow the sender to time segment transmission well, so the effect of non-synchronized segment corruption does not yield more-than-doubled total transmission times, as was the case for the "Internet scenario" of Figure 3(a). All three TCP versions perform better under a model that does not drop ACKs than under a model that corrupts ACKs. In particular, Tahoe seems to benefit the most from this, achieving the best performance overall. Clearly, the fluctuations in relative performance can lead to completely opposite conclusions about the most efficient TCP version.

4 CONCLUSIONS

This paper attempts to draw attention to the effects of two broad characteristics of error modeling in TCP simulations. First, we considered rate-based vs. temporal error models. We showed that the error model used affects measured TCP performance, and under certain scenarios the performance can be

strikingly different. We show that rate-based error models do not allow TCP (and proposed modifications to it, as well other new transport protocols) to exhibit their potential efficiency. Temporal error models provide a better simulation environment.

Temporal error models can also have variations. Our results show that TCP performance is dramatically different under a synchronized error model compared to a non-synchronized one. In addition, we compared a synchronized model with a model that introduces errors only in the forward path, in order to test the implications of the assumption, commonly made in analytic TCP performance modeling, that ACKs are not lost. TCP performance under the latter version of the model is not significantly different from the former to the extent that error rates and/or the RTTs are small. Under the synchronized model, when the error rate is small, enough ACKs are received to allow the sender to correctly estimate network conditions. On the other hand, when the error rate increases, and under a model that does not drop ACKs, the sender is unable to correctly gauge these conditions because the flow of ACKs is, at best, a delayed and possibly obsolete indication of current conditions in the forward direction. Thus, a TCP sender would continue transmission, and possibly lose segments. The size of the RTT also plays a crucial role in the performance of TCP under these two versions of the error model: the difference in the measured TCP performance between the two increases with the RTT delays.

An important aspect of our results is that, with respect to the two broad characteristics considered, and to which insufficient attention has heretofore been given by researchers and practitioners in the field, error modeling affects measured TCP performance significantly and thus can play an important role in judging the efficiency of proposed modifications to TCP or new transport protocols. At least in the cases presented in this paper, the relative performance of three versions of TCP was shown to depend, sometimes heavily, on the error model.

REFERENCES

- Allman, M., Paxson, V., and Stevens, W. R., (1999). *TCP Congestion Control*, RFC 2581.
- Allman, M., Balakrishnan, H., and Floyd, S., (2000). *Enhancing TCP's Loss Recovery Using Early Duplicate Acknowledgment Response*, Internet Draft, June 2000.
- Bakre, A., & Badrinath, B. R. (1995). "I-TCP: Indirect TCP for Mobile Hosts". In *Proceedings of the 15th International Conference on Distributed Computing Systems (IDCS)*, May 1995.
- Balakrishnan, H., Seshan, S., Amir, E., & Katz, R. (1995). "Improving TCP/IP Performance over Wireless Networks". In *Proceedings of ACM MobiCom*, November 1995.
- Cáceres, R. & Iftode, L. (1995). "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments". In *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 5.
- Fall, K. & Floyd, S. (1996). "Simulation-based Comparisons of Tahoe, Reno and SACK TCP". In *Computer Communication Review*, Volume 26 No. 3, pp. 36-46.
- Floyd, S. & Henderson, T. (1999). *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 2582, April 1999.
- Lee, W. C. Y. (1993). *Mobile Communications Design Fundamentals, 2nd edition*. John Wiley and Sons, 1993.
- UCB/LBNL/VINT Network Simulator – ns-2 (2000), available at <http://www-mash.cs.berkeley.edu/ns>.
- Parsa, C. & Garcia-Luna-Aceves, J.J. (1999). "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media", In *Proceedings of IEEE ICNP '99*, Toronto, October 1999.
- Pentikousis, K. (2000). *Error Modeling for TCP Performance Evaluation*. Technical Report, TR-00-0510, Department of Computer Science, State University of New York at Stony Brook, May 2000.
- Postel, J. B (1981). *Transmission Control Protocol*. RFC 793, September 1981.
- Sinha, P., Venkitaraman, N., Sivakumar, R., Bharghavan, V. (1999). "WTCP: a reliable transport protocol for wireless wide-area networks", In *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999, Seattle, WA USA.
- Tabbane, S. (2000). *Handbook of Mobile Radio Networks*, Artech House Mobile Communications Library, 2000.
- x-kernel Programmer's Manual (2000), available at <http://www.cs.arizona.edu/xkernel>.
- Xylomenos, G. & Polyzos, G. (1999). "TCP and UDP Performance over a Wireless LAN." In *Proceedings of IEEE INFOCOM 1999*.
- Zorzi, M., Rao, R. (1999). "Perspectives on the Impact of Error Statistics on Protocols for Wireless Networks." In *IEEE Personal Communications*, vol. 6, Oct. 1999.