TCP/IP Modeling and Validation

Chadi Barakat, INRIA, France

Abstract

We discuss in this article the different issues to be considered when modeling the TCP protocol in a real environment. The discussion is based on measurements we made over the Internet. We show that the Internet is so heterogeneous that a simplistic assumption about TCP congestion control or the network may lead to erroneous results. We outline some of our results in this field, and we present a novel approach for a correct validation of a model for TCP.

The modeling of TCP congestion control is an important task for improving the service provided to Internet users and the efficiency of network resource utilization. Its importance is due to the large amount of TCP traffic in the Internet: 95 percent of total bytes and 85–95 percent of total packets are of the TCP type [1]. A TCP source controls the rate of application packets as a function of the way the network treats or reacts to these packets [2]. The main objective of modeling is to come up with simple expressions of TCP throughput for a given network reaction. This has two main advantages. First, it determines the factors influencing the performance of the protocol, which gives people working on TCP insights on how the congestion control has to be improved. For example, modeling has shown that in case of drop-tail buffers and synchronized TCP flows, the throughput of a TCP connection is inversely proportional to the square of the average round-trip time (RTT) [3]. This has given an explanation for and evaluation of the bias of TCP against connections with long RTTs. Using this result, [4] proposed a modification to the standard window increase algorithm of TCP in order to improve the fairness of the protocol. Modeling has also shown that dropping packets randomly in network routers, as done by random early detection (RED) [5], improves the fairness of TCP by making the throughput of a connection inversely proportional to the average RTT instead of its square [3, 6–8]. Another possible use of an expression of TCP throughput could be the study of a change of the parameters of TCP congestion control to minimize the variations of the window without adding to the aggressiveness of the protocol. This will be useful for multimedia applications using TCP, or possibly a new version of TCP adapted to such applications.

The second advantage of TCP modeling is that it permits network designers to improve the reaction of their networks to incoming packets at the moment of congestion, given the current control policy of TCP. Important work has been done in this direction (e.g., [9–13]). The focus was and is always on the dimensioning of buffers in network routers and the management of their occupancy. The objectives [5] are to maximize the utilization of network resources, to protect the TCP transfers of one to the other, and to minimize the queuing time in routers so that delay-sensitive applications (e.g., Telnet, short Web transfers) get better service. A typical problem of this kind is the tuning of RED parameters [9, 10, 12]. RED [5] is an interesting buffer management technique presented as a solution to achieve the above objectives. For example, using the expression of TCP throughput calculated in [8], it was shown in [10] that when the number of TCP connections exceeds a certain threshold, the RED buffer gets into an unstable regime. This instability has been explained by the sudden jump in the drop probability when the average queue length reaches the maximum threshold (see [5] for more details on RED parameters). This has motivated the introduction of a new parameter to RED (the *gentle* parameter) in order to avoid such a jump. Other authors have used the explicit expressions of TCP throughput to improve other parts of the network as the link layer on a wireless interface [14].

Recently, a new application of TCP modeling has emerged [15, 16]. It consists of using the explicit expressions of TCP throughput to control the rate of real-time flows (e.g., an audio flow) in a TCP-friendly way. The reaction of the network (e.g., packet drop probability) is averaged over a certain time interval, and the rate of the real-time flow is set to the expected throughput of a TCP connection running in the same network conditions. Clearly, a trade-off exists between choosing a long averaging time interval to get a smooth variation of the rate — hence low jitter and good quality — and choosing short intervals to get a fast reaction to changes in network conditions.

The modeling of TCP can be seen as a two-step procedure. First, we model the evolution of TCP window between congestion events as well as at the moment of congestion. This includes the modeling of some particular mechanisms of TCP such as timeouts and the limitation of the rate due to the window advertised by the receiver [2]. Then comes the modeling of network reaction which consists, directly or after some transformation, of modeling the process of congestion events. The TCP connection lifetime can be seen as a succession of congestion events between which TCP increases its window and during which it reduces it. As we will see later, a model for the network also has to describe how the RTT of the connection varies. The variation of the RTT determines the variation of the window between congestion events. Note that TCP increases its window by the same amount during an RTT regardless of its length. An example of congestion events and window evolution is depicted in Fig. 4.

If the network is well defined (e.g., a single RED buffer), the modeling of TCP can be achieved with a considerable degree of accuracy. One can come up with an accurate model for the network, combine it with a model for TCP, and solve the overall model for TCP performance. All work that assumes a single bottleneck router crossed by only TCP connections is of this type (e.g., [10, 11]). The difficulty arises when we want to approximate the throughput of TCP in

0890-8044/01/$10.00 © 2001 IEEE

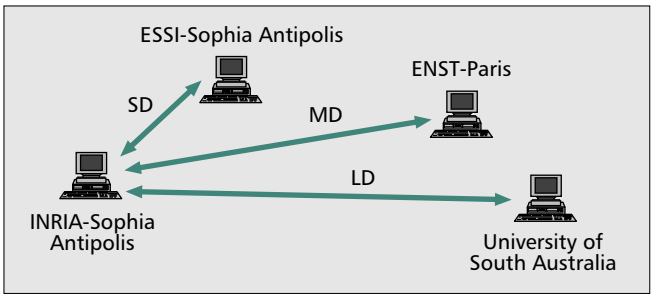


Figure 1. *The measurement testbed.*

the real Internet. Some assumptions are then required to model the reaction of this huge and heterogeneous environment. The difficulty of the analysis as well as the accuracy of the modeling change with the assumptions we make. For any given assumption we may expect our results to hold on some Internet paths while not on others. The results may hold due to the correctness of our modeling on these particular paths. They may also hold due to another phenomenon that we observed during our work: the cancellation of errors introduced by the different blocks of a model for TCP. A good model for TCP must have all its blocks validated separately, and these blocks must work well on a wide range of Internet paths.

We address in this article the problem of TCP modeling in its general form. We focus mainly on the modeling of TCP in a real network rather than a particular environment as a single drop tail buffer or a RED buffer. Based on measurements we conducted over the Internet, we discuss the different issues to be considered for correct modeling of TCP. We outline the different approaches in this domain, and show the need for a general approach that copes with the heterogeneity of network reaction we observed. We will see how the heterogeneity of the Internet can change the performance of a model from one path to another, and even from one hour to another on the same path. We introduce the notion of separate validation of each block of a model for TCP instead of overall model validation. We also discuss some issues related to the use of throughput expressions in practice, particularly in TCP-friendly applications.

In the next section, we present our measurement testbed to which we will refer in the course of the article. We then discuss the issues related to the modeling of TCP window evolution. The issues related to network modeling are also discussed. At the end, we explain our technique for the validation of a model for TCP, and then conclude the article. We refer a reader to [17] for more details.

Measurement Testbed

Our testbed consists of three long-life TCP connections run over the Internet between a machine at INRIA Sophia Antipolis in the south of France and three other machines (Fig. 1). The first machine is located at ESSI next to INRIA. The second machine is located in Paris about 800 km from INRIA. The third machine is located at the University of South Australia. These can be considered as short-distance, medium-distance, and long-distance connections, respectively. Henceforth, we use the terms SD, MD, and LD to distinguish between the three connections. The source is located at INRIA and runs the NewReno version of TCP [18]. NewReno is a recent version of TCP able to recover from multiple packet losses without timeout and with only one division of the window by two. The source of our connections is fed by a simple application that always has data to send. The connections were run for many hours on different days during the month of January 2000. We developed and ran a tool at INRIA that looks at the trace of every connection (packets and ACKs) and makes a number of statistical calculations on the connection, such as the total number of packets acknowledged, total number of retransmissions, moments at which the window is reduced, and temporal variation of window size and round-trip time. We accounted for all the mechanisms of NewReno when developing our tool, particularly the fact that a NewReno source can recover from multiple packet losses in the same window of data. We stored the statistics at fixed time intervals in separate files. We chose these intervals to get enough data in each file. We shall assume that the network conditions are approximately the same during each interval. These time intervals are set to 20 min for the SD connection, 40 min for the MD connection, and 60 min for the LD connection.

Modeling TCP Window Evolution

Most of the effort on TCP modeling [3, 6–8, 11, 12, 19] is devoted to long transfers, namely to the congestion avoidance mode of TCP [2]. The slow start phase [2], due to its fast increase of window size and short duration, is often ignored. The window is assumed to increase linearly with time between congestion events and to decrease to half its size upon congestion. The moments of congestion are determined by the underlying model for the network. These are the moments at which the source detects the loss of a packet and decides to reduce its window. Ideally (this is what the new versions of TCP try to approximate [18]), these must be the moments of detection of the first loss of data packets in a window.

TCP is known to increase its window size in congestion avoidance mode in a linear manner but as a function of RTT number rather than time. This increase is approximately equal to $1/b$ packets/RTT, where b is the number of packets covered by an ACK [8]. For the linear increase with time to hold, the RTT is often supposed to be constant or to vary independently of window size, so it is substituted in the analysis by its average during the transfer [3, 7, 8].

For the multiplicative decrease factor, it is indeed equal to one half when the source receives three duplicate ACKs for congestion detection and when the fast recovery phase succeeds [18]. If this is not the case, a timeout occurs, the window is set to one packet, and slow start is called to quickly reach the slow start threshold [2]. Some authors [6, 8] keep the decrease factor in the case of timeout equal to one half, while others [19] set the window to one packet and assume a linear increase from this low value. We believe that there will be no difference between the two approaches in the future given the expected low probability of timeout with the new enhancements proposed to TCP congestion control [20]. Our measurements showed that on the LD connection, most of the losses are detected with timeout; this is due to the small size of the TCP window. The same result is noticed in [8]. On the SD and MD connections, the window is large and the timeout phenomenon is absent. In our work on TCP modeling [6], we set the multiplicative decrease factor to a constant value v ($v < 1$) for both kinds of congestion detection methods. This factor v, together with the general window increase rate α we considered, permits our results to be used for the study of other congestion control policies.

Clearly, the above model is simple since it only accounts for the linear-increase multiplicative-decrease part of TCP congestion control. Other issues have to be considered for better modeling of TCP. We will enumerate those that are the most important in the following sections and explain when possible how they can be introduced into the above model.

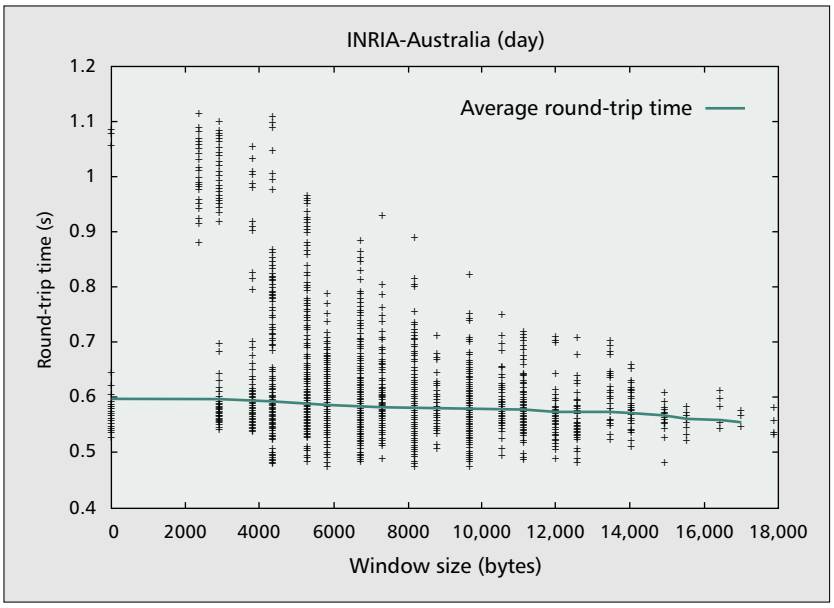


Figure 2. *The LD connection: RTT vs. window size.*

The Dependence between Window Size and RTT

On paths where the window of the TCP connection is small compared to the bandwidth-delay product and packets cross multiple congested routers, one should expect that the assumption on the linearity of the window increase will hold. Indeed, on such paths the connection does not contribute to the queuing time in network routers, and congestion is mostly caused by other aggressive connections. The RTT then varies independently of the window size and can be substituted in the analysis by its average [8], resulting in time-linear window growth at rate $1/(bRTT)$, with RTT being the average rtt and b the number of packets covered by an ACK. However, this independence is not expected to hold on paths where the window of the connection is large compared to the bandwidth-delay product. The increase in the window in this latter case will result in an increase in the RTT, which in turn will result in a sublinear increase in the window with time [3]. One should expect that, in the presence of a linear model for TCP window evolution which uses the average RTT for the calculation of the window slope, this sublinearity will result in an overestimation of the real throughput.

To understand such dependence, we plot the variation of the RTT as a function of the window size on our LD and SD connections (Figs. 2 and 3). The LD connection is a typical example of the first connection, whereas the SD connection is a typical example of the second one. We measure the RTT for one packet per window of packets and note the window size during this measurement. We then average the RTTs obtained for close window sizes. This gives the thick line in both figures. We see clearly how, on the LD connection, the RTT is on average constant and independent of window size. However, it is an increasing function of the window size on the SD connection, and a sublinear window increase should be seen on this connection. Indeed, Fig. 4 shows this sublinear behavior well. We plot in this latter figure the variation of the real window on the SD connection for some seconds. The straight line corresponds to the expected window evolution if the RTT were independent of the window size. This line is given by the time-linear model for TCP window evolution.

The problem with the sublinearity of window increase is that it complicates the analysis and makes it difficult (if not impossible) to obtain simple explicit expressions for TCP throughput. The literature that considers such sublinearity (e.g., [11, 12]) gives the throughput as a solution of some implicit differential equations. Only the work considering linear models (e.g., [6, 8]) succeeds in finding closed-form expressions for TCP throughput. Moreover, it is not clear at the moment how such sublinearity can be modeled in so complicated a network as the Internet. The importance of the sublinearity depends on many factors, such as the intensity of exogenous traffic sharing the path with the TCP connection, the buffering capacity in network routers, and the available bandwidth on the path. In a real environment, one must look at the traces of RTT and window size, and try to infer from these traces some model for the dependence between RTT and window size. In previous work [11, 12], this inference was not necessary since the content of the network was well defined (e.g., a single RED buffer with given bandwidth and propagation delay); hence, the variation of the RTT as a function of the window size was well known. The model for RTT, together with that for window evolution and that for network reaction, provide the required information for the calculation of TCP throughput. For example, one can use the models developed in [12, 21] for such calculation.

A Model for RTT Variation — Using the traces of our SD connection, we propose a technique for the inference of the dependence between window size and RTT. First, we have to define a model for the variation of the RTT. Then we have to use some fitting technique to infer the parameters of such a model from the traces of the connection. It seems that the most appropriate model for RTT variation is the one used in

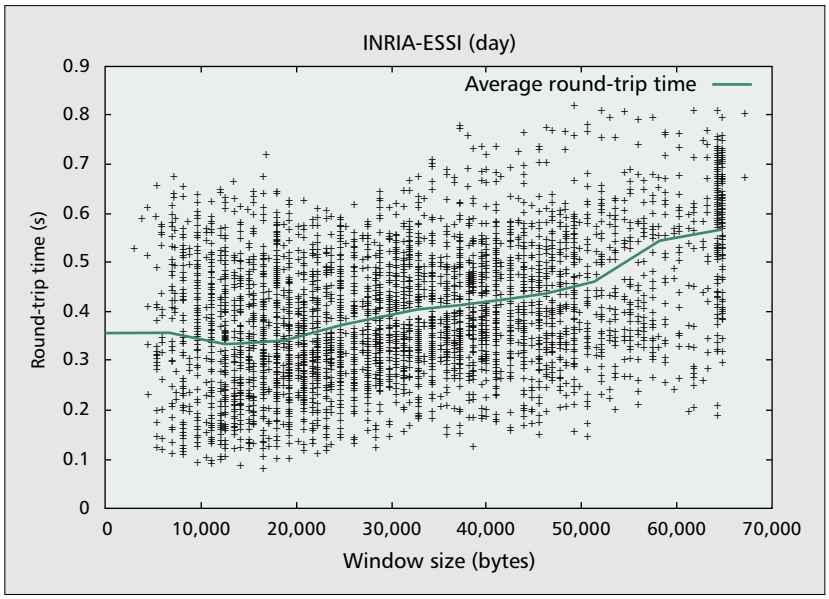


Figure 3. *The SD connection: RTT vs. window size.*

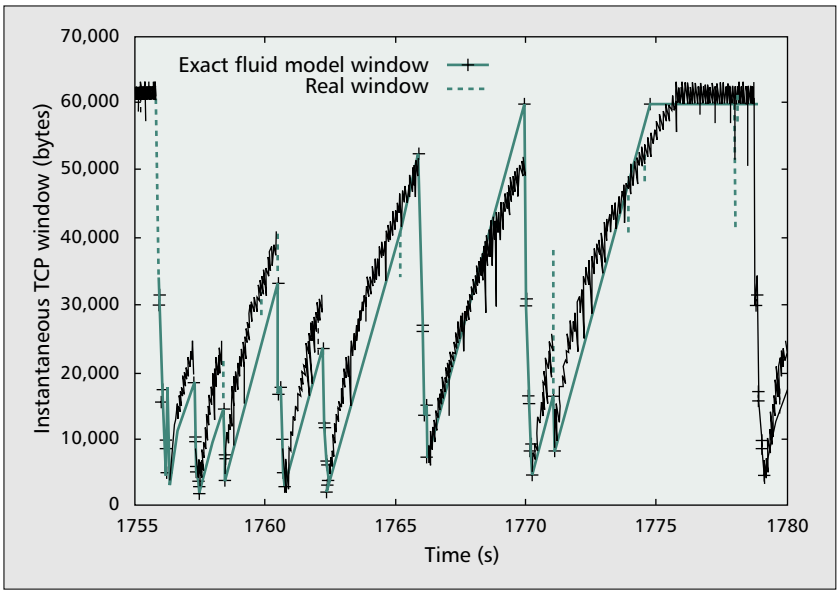


Figure 4. *SDC: Window size vs. time.*

the literature for networks of one router (e.g., [3]). Denote by W the window of the TCP connection. The model considers the RTT to be constant whenever W is less than a certain window size W_0. For windows larger than W_0, the RTT increases linearly with W at a rate β. We write the model as follows:

$$RTT_e = RTT_0 + 1\{W > W_0\}\ \beta\ (W - W_0).$$

RTT_0 can be seen as the contribution of the propagation delay and the exogenous traffic to the total RTT. $\beta(W - W_0)$ represents the contribution of the connection to the total RTT when its window is large. $1/\beta$ represents the bottleneck bandwidth on the path of the connection.

Given this model for the RTT, we use the nonlinear least square technique to infer the three parameters RTT_0, β, and W_0 of the model from the traces of the connection. Let RTT_m be an RTT measurement that corresponds to a window W. We can write RTT_m as follows:

$$RTT_m = RTT_e + \epsilon,$$

where ϵ is a certain error. Let ϵ_n be the error introduced by the nth RTT measurement. We propose to find the parameters of the model that minimize the sum

$$\sum_{n=1}^{N} \epsilon_n^2,$$

where N is the total number of RTT measurements.

We solve the minimization problem for the traces of our SD connection shown in Fig. 3. We get the thick line in Fig. 5. We also plot in Fig. 5 the 95 percent confidence intervals as well as the measurements we obtained. Our results show clearly that there is a dependence between window size and RTT on this particular path. For small windows, the expected RTT is constant and equal to 0.35 s. Once the window of the connection exceeds 2246 bytes, the expected RTT starts to increase linearly by 0.0493 s every 10,000 bytes.

Modeling Timeouts and Fast Recovery

The above modeling does not consider these two mechanisms of TCP [2, 18]. Due to the coarse granularity of TCP timers (500 ms in most implementations), the first mechanism introduces a certain idle time between congestion and its detection. The second mechanism also introduces a certain time between the detection of congestion with duplicate ACKs and the resumption of window increase. During this latter time, the source is supposed to recover from losses and transmit new packets in order for the ACK clock not to stop.

The modeling of these two mechanisms of TCP requires a detailed description of the protocol behavior at the packet level. This behavior is quite complicated to model and varies from one version to another [18]. The fast recovery phase is always ignored due to the complexity of its modeling, and because it adds a negligible contribution to the throughput when it works well. It is assumed in [8] that once the source receives three duplicate ACKs, the window resumes its linear increase until the next congestion event. This seems reasonable with the new versions of TCP (e.g., TCP-SACK [18]) where the fast recovery phase is quite robust and fast, but it is not true with other versions such as Reno [18], where fast recovery may fail due to multiple losses per window. A failure of fast recovery results in a timeout and a slow start from a window of one packet. A model that does not account for the fast recovery phase will lead to throughput overestimation if such failures are frequent.

The timeout mechanism is studied in [8] using the probability that a packet is dropped. Denote this probability by p. The authors focus on the calculation of the probability that the source fails to receive three duplicate ACKs to trigger fast recovery. This has been assumed to be the necessary and sufficient condition for the occurrence of a timeout. Timeouts are assumed not to occur during the fast recovery phase. We believe that this will be the case with the new versions of TCP which use the SACK option [18], and have a robust and quick fast recovery phase. We also believe that with the modifications proposed to TCP error recovery in [20], failure to

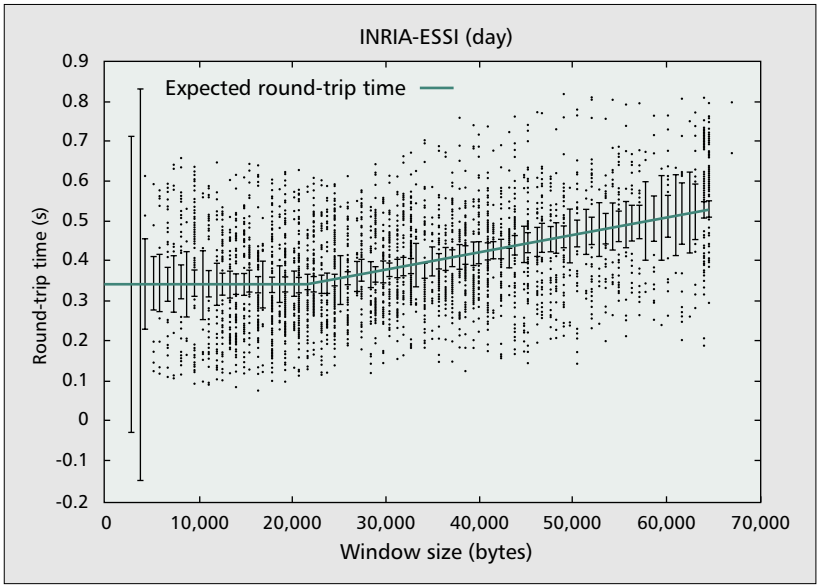


Figure 5. *The SD connection: expected RTT vs. window size.*

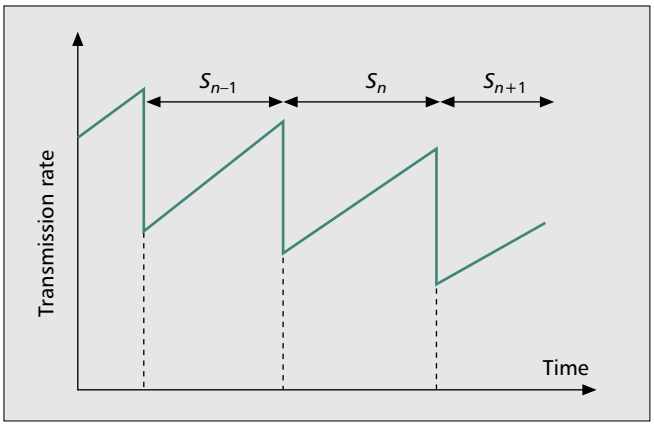


Figure 6. *The model without timeout intervals.*

receive three duplicate ACKs will no longer be a sufficient condition to get a timeout. Sources will be able to recover from losses when more than one ACK are received per RTT. In anticipation of the deployment of these modifications, we will show in the following a heuristic for the addition of the timeout mechanism to the above model according to the condition in [8].

To simplify the analysis, it is assumed in [8] that when a packet is dropped, the subsequent packets in the same window are also dropped. The probability to get a timeout ($Q(p)$) as well as the average duration of a timeout period ($Z(p)$) are calculated as a function of p,

$$Q(p) = \min\left(\frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{W-3}))}{1-(1-p)^W}, 1\right),$$

$$Z(p) = T_0 \frac{1+p+2p^2+4p^3+8p^4+16p^5+32p^6}{1-p},$$

with

$$W = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2},$$

and T_0 being the basic timeout interval which is doubled when the source backs off its timer after the loss of a retransmission [2]. In practice, $4RTT$ has been shown to be a safe value for T_0 [16].

Using these two functions of p, the influence of timeout intervals can be introduced into the model in the following way. We calculate first the throughput of the connection when excluding these intervals; in other words, when assuming that the window resumes its linear increase directly after a congestion event whatever the method of detection (Fig. 6). Denote this throughput by $\overline{X}_d$. Denote by $\overline{X}_f$ the throughput of the connection in the presence of timeout intervals (Fig. 7). The throughput in both cases is equal to the ratio of the average number of packets that cross the network between two congestion events ($\simeq 1/p$) and the average time between congestion events (see [6, 8] for details). Let $\{S_n\}$ denote the process of time intervals between congestion events when timeout intervals are excluded (Fig. 6). Let $\{S'_n\}$ denote the process of time intervals between congestion events when timeout intervals are considered (Fig. 7). Given the average number of packets that cross the network between two congestion events is the same in both cases ($\simeq 1/p$), it follows that

$$\overline{X}_f = \overline{X}_d \frac{E[S_n]}{E[S'_n]},$$

and

$$\overline{X}_d = \frac{1/p}{E[S_n]}.$$

Using these two equations and the fact that

$$E[S'_n] = E[S_n] + Q(p)Z(p),$$

we can write

$$\overline{X}_f = \frac{\overline{X}_d}{1 + p\overline{X}_d Q(p)Z(p)}.$$

Note that our expressions for the throughput are in terms of packets per second. Note also that our throughput corresponds to the rate at which TCP packets leave the network, also called the *receiving rate*.

Modeling Window Limitation

Another problem with the previous model is that it does not consider the fact that a TCP source cannot inject into the network more packets than the window advertised by the receiver [8]. This phenomenon can be seen in Fig. 4 where we plot the number of bytes in the network as a function of time. The model becomes sublinear, and the calculation of a simple expression for the throughput seems to be impossible except for some particular models for the network [8, 18, 21]. For example, the calculation is straightforward when the time between congestion events is constant. Using techniques from queuing theory, the authors in [12, 21] succeed in calculating the throughput when congestion events appear according to a homogeneous Poisson process.

For more general models for the network (e.g., a generally distributed time between congestion events), one can always think about finding some approximations of the throughput. One possible approximation is to assume that the receiver window is always reached between congestion events. The problem is then automatically transformed into a simpler one with constant time intervals between congestion events equal to the average of these time intervals. This is the kind of approximation used in [8]. In [6] we find bounds for the throughput which are also a good approximation. The advantage of our bounds is that they are valid for any receiver window. The first approximation is only valid for small receiver windows, so one needs to apply a condition on the average window at moments of congestion in order to use either the approximation or the expression of the throughput obtained when there is no window limitation.

Note that the problem of window limitation exists because of

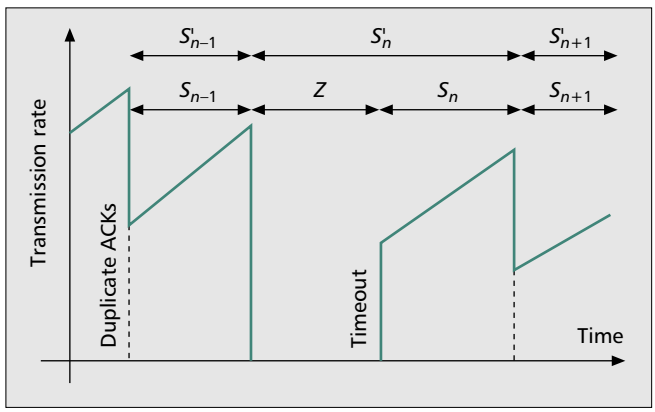


Figure 7. *The model with timeout intervals.*

the small windows advertised by current Internet receivers (less than 64 kbytes). This is supposed to disappear in the future with the trend of increasing the window field in the TCP header and dynamically changing the buffer size allocated to the TCP connection at the receiver [22]. It is even recommended in [23] that future studies of TCP congestion control should consider an infinite receiver window. Models that account for the limitation of TCP transmission rate will not be of much importance.

Fluid Models vs. Discrete Models

Some of the models for TCP assume that the window increases continuously between congestion events [7, 12, 18, 24]. The use of a continuous model for the window facilitates analysis since it permits the use of tools from the theory of continuous functions such as integration and differentiation. The continuous increase may hold for the congestion window at the source, but not for the volume of data in the network. Indeed, the former quantity increases in small values (in bytes) upon ACK arrivals, whereas the latter increases in steps of one packet every b RTTs [8]. This is due to the Nagle algorithm [25], which prohibits the TCP source from injecting small packets into the network. At the moment of congestion, it is the window at the source which is divided by two rather than the number of packets in the network [2]. So models assuming continuous increase in the window, also called *fluid* models, are appropriate for the prediction of the window variation at the TCP source. The throughput obtained with these models is the throughput the TCP connection would realize if it were not limited by the Nagle algorithm.

The correct throughput can be obtained by one of two ways: either using a discrete model for the window evolution such as the one in [8], or using a fluid model and introducing some corrections into the calculated throughput to account for the difference between the size of the congestion window at the source and the volume of data in the network. In our work on TCP modeling [6, 21], we chose the second alternative for the calculation of TCP throughput due to the simplicity of analysis the use of fluid models permits. We present here the technique we used in order to account for the packet nature of TCP. To illustrate this technique, we plot in Fig. 8 the number of packets that cross the network between two congestion events for both a fluid model and a discrete (or packet) model. The curve for the discrete model is taken from [8]. On average, half of the window is assumed to be dropped in the network upon congestion (the last RTT in the figure). Suppose that all the transmission rates are expressed in terms of packets/s. Denote by E $[W_n]$ the average window size upon congestion and by E $[S_n]$ the average time interval between congestion events. Let $\overline{X}$ denote the throughput obtained with the fluid model and $\overline{X}_d$ the real throughput of TCP. A good approximation for $\overline{X}_d$ can be obtained by:

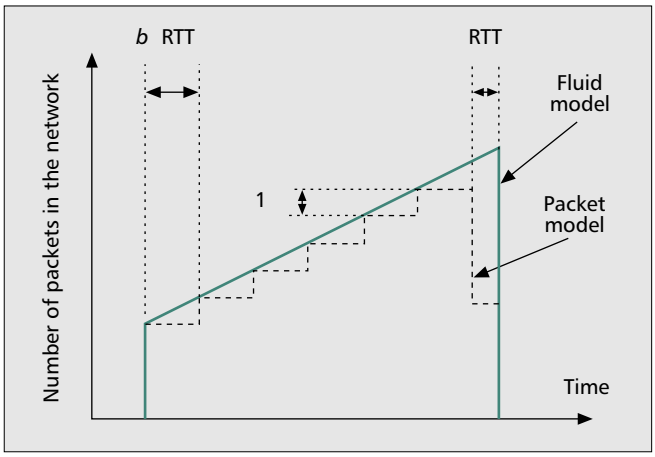
- Shifting down the fluid window by 0.5 packet, which results in a decrease in the throughput $\overline{X}$ by 0.5/RTT.
- Subtracting the rate of dropped packets, which is approximately equal to

$$\text{Rate of dropped packets} = \frac{1}{2}\frac{E[W_n]}{E[S_n]}.$$

It follows that

$$\overline{X}_d = \overline{X} - \frac{0.5}{RTT} - \frac{0.5E[W_n]}{E[S_n]}.$$

Using one of our results [3, Eq. 3],



■ Figure 8. *The fluid model vs. the discrete model.*

$$E[W_n] = \frac{E[S_n]}{bRTT(1-v)},$$

we can write

$$\overline{X}_d = \overline{X} - \frac{0.5}{RTT}\left(1 + \frac{1}{b(1-v)}\right).$$

With this correction, and as we will see in later figures when validating our model, a fluid model for TCP is able to give the same throughput as a detailed discrete model.

Modeling the Network

This is the part of the model where the heterogeneity of the Internet has the greatest impact. The objective of this part is to find a good characterization of congestion or loss moments. A loss moment in our terminology is the moment at which the TCP source decides that the network is congested and it must reduce its window. These moments can be directly characterized by making some assumptions on the way they appear. For example, one can assume that they appear according to a deterministic process or a Poisson process. This direct characterization is the approach widely used in the literature [6–8, 18]. Another possible but indirect characterization consists of finding a model for the reaction of the network to a particular TCP packet or during a small time interval. Examples of this approach are papers which suppose that TCP packets are dropped within the network with a constant or variable probability [11], or those assuming that loss moments form a Poisson process with a variable intensity function of the window size of the connection [24]. The advantage of the indirect approach is that it decouples the model for the network from the control policy at the TCP source. It happens that on some Internet paths, the process of losses seen by a TCP connection is a function of the way it increases and decreases its window, and this process changes if another control policy is used (e.g., if packets are transmitted at a constant rate). The indirect approach is very useful on such paths since it permits one to deduce the loss process a TCP connection will see from that seen by another connection with a different congestion control policy. We cite different applications of the indirect approach on such paths. As an application one can probe the network with a given flow of packets (e.g., a constant rate flow) and calculate the parameters of an appropriate model for the network (e.g., packet drop probability, variation of loss intensity as a function of transmission rate). With these parameters, it is possible to predict the performance of a TCP

Hour (traces of 20 min)	Covariance coefficient $Cov(S_n, S_{n-1})/Var(S_n)$
11:00	+0.034
12:00	+0.041
12:30	+0.113
13:00	–0.001
13:30	–0.191
14:00	–0.078

Table 1. *SD: covariance coefficient.*

Hour (traces of 40 min)	Covariance coefficient $Cov(S_n, S_{n-1}) = Var(S_n)$
15:00	+0.106
19:00	+0.101
20:00	+0.015
21:00	–0.01
22:00	–0.048
23:00	–0.005

Table 2. *MD: covariance coefficient.*

Hour (traces of 60 min)	Covariance coefficient $Cov(S_n, S_{n-1}) = Var(S_n)$
11:00	–0.197
12:00	–0.001
14:00	–0.102
16:00	–0.107
20:00	+0.023
22:00	–0.09

Table 3. *LD: covariance coefficient.*

connection on the same path. Another application of the indirect approach is that from the trace of a TCP connection, one can build a model for the network and predict the performance of another TCP connection with another congestion control policy. This will be useful for a study of the influence of a change of TCP congestion control parameters. A third application of the indirect approach is that a TCP-friendly application (e.g., TFRC [16]) can deduce, from the loss process it sees, the loss process a TCP flow would see, and hence get a better estimate of the rate to use. All these applications are not possible with the direct approach since, without a model for network reaction, we cannot deduce the performance of a TCP connection from the loss process seen by another connection with another control policy. Recall that we are talking about paths where the loss process changes with the congestion control policy.

The difficulty with the indirect approach is in the definition of a correct model for the Internet and the calculation of its parameters. This may be possible for a simple network of one router (e.g., a RED buffer is known to drop packets with a probability that increases linearly with the average queue length [5]), but it seems to be difficult for a wide network such as the Internet. It is not clear if a model exists for the Internet that works on all paths. Certainly, the Internet reaction to TCP packets changes from one path to another and with time on the same path. For example, on some paths the network may drop packets with a constant probability, on other paths with a probability that increases linearly with the congestion window, on other paths with a probability that increases logarithmically with the congestion window, and so on. One can imagine different models for the network. The question is, on how many paths are such models useful?

Given the difficulty in coming up with a model for network reaction that works on all Internet paths, we shall only focus on the direct approach. Recall that the direct approach consists of using the parameters of the loss process seen by the connection for throughput calculation. Our TCP connections use the same control policy; thus, we can decide on the heterogeneity of the Internet from the loss processes they see. In the next section we present some measurement results to show to what extent the Internet is heterogenous. We then use the expression of TCP throughput we found in [6] to evaluate the influence of the assumption on the network we make. Note that in [6], we found a simple expression of TCP throughput for a general process of congestion events.

Diversity of Loss Processes in the Internet

We present in this section some of the loss processes we found on our three connections. First, we plot the distribution of time intervals between congestion events. Figures 9–14 show some samples of the distributions we got. The figures contain for comparison some theoretical distributions (exponential for the SD and LD connections, normal for the MD connection). On the SD connection the process is highly bursty, which results in a spike close to the origin. This burstiness can also be seen in Fig. 4, where the window is repeatedly divided by two in short time intervals. We noticed that the congestion on our SD connection persists for multiple consecutive RTTs during which the network keeps dropping packets and the source keeps reducing its window. On the MD connection, the time intervals between losses closely follow a normal distribution. On the LD connection, as one would expect, the loss process is close to Poisson. Indeed, on the LD connection the source has a small window and does not contribute to the congestion of the network. The loss process it sees is the superposition of a large number of processes in all the routers it crosses. We also found some correlation of losses on the three connections. This correlation varies during the day between negative and positive values with an absolute value of the covariance coefficient sometimes reaching 0.2. Tables 1, 2, and 3 show some of the covariance coefficients we saw on the three connections at different hours during the day. Recall that the covariance coefficient (of order 1) of a process $\{S_n\}$is given by

$$Cov = \frac{E[S_n S_{n-1}] - E[S_n]^2}{E[S_n^2] - E[S_n]^2}.$$

This coefficient varies between –1 when interloss times are highly negatively correlated and +1 when they are highly positively correlated.

Our measurements show some examples of losses a TCP connection may see over the Internet. We expect to see other processes on other paths. Other distributions of inter-loss times could be found. Also, some paths such as those including satellite and wireless links, may exhibit more memory than our paths which will result in a more important correlation of losses. Using one of our results in [6], we discuss in the next section the impact of the different parameters of a loss process on TCP throughput. This will give us an idea of how much a certain assumption on the occurrence of losses or congestion events influences the result of the modeling.

The Influence of the Loss Process Choice on TCP Modeling

Most of existing models for TCP make simplistic assumptions on interloss times (e.g., deterministic [7, 8], Poisson [19]). In [6] we propose a general model for the network where we only make the restricted assumption that the loss process is stationary and ergodic. We allow the interloss times to follow any distribution and the losses to be correlated. We derive a general expression for the throughput which uses, in addition

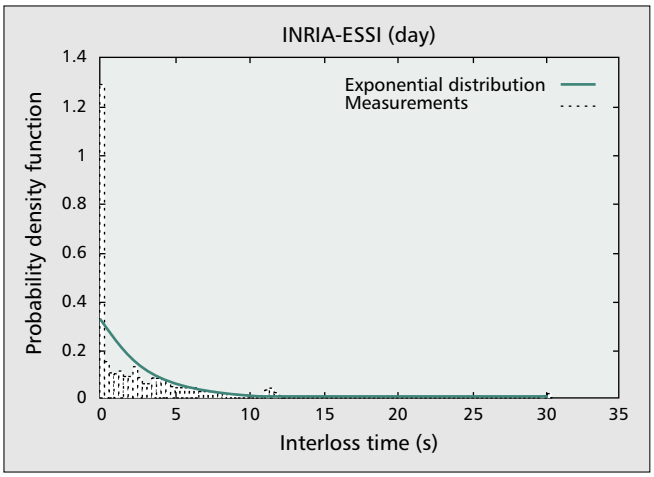


Figure 9. *SD connection: interloss time distribution.*

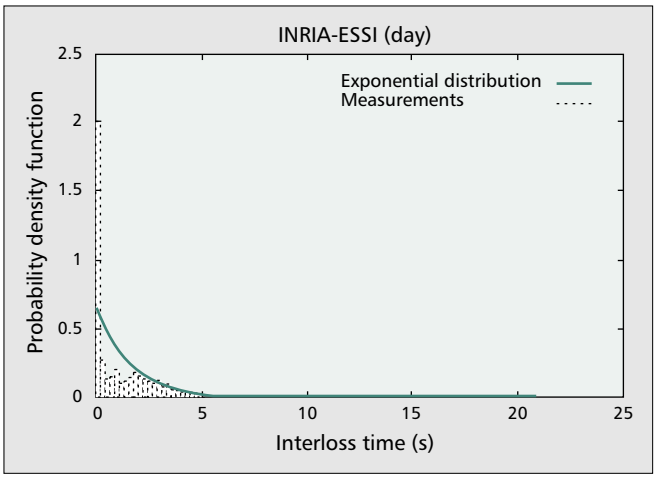


Figure 10. *SD connection: interloss time distribution.*

to the packet loss probability (p) and the average RTT (RTT), the variance of interloss times as well as the covariance function of the loss process. For a window reduction factor equal to one half we find

$$\text{TCP throughput} = \frac{1}{RTT\sqrt{bp}}\sqrt{\frac{3}{2}+\frac{1}{2}\hat{V}+\sum_{k=1}^{\infty}\left(\frac{1}{2}\right)^{k}\hat{C}(k)}.$$

$\hat{V}$ and $\hat{C}(k)$ are the variance ($\mathrm{Var}S_n$) and covariance functions ($\mathrm{Cov}(S_n,S_{\{n-1\}})$), respectively, of the loss process normalized to the square of the average time between losses.

Three terms appear under the square root in our formula. The first term corresponds to the intensity of losses or average interloss time. The second term represents the variation of interloss times. The third term represents the correlation of losses. This third term is equal to zero when the loss process exhibits no correlation. If we take only the first term, we get the well-known square root formula [7] which was indeed established for a deterministic loss process.

This formula gives the error we introduce when making a certain assumption on the loss process. For example, it tells us that the throughput is an increasing function of the variance of interloss times. Assuming that losses are deterministic when they are actually Poisson should lead to an underestimation of the real throughput. In contrast, assuming that losses are Poisson when they are deterministic should lead to an overestimation of the real throughput. The next section contains some results of measurements that show how the use of a wrong variance yields a wrong throughput estimate. The formula also tells us that due to the geometrical decrease in the weights of the $\hat{C}(k)$, a small number of covariance functions is sufficient to calculate the throughput even if the loss process is highly correlated. The window evolution becomes independent of the past after a certain number of loss events.

Separate Model Validation

We introduce in this section the notion of separate validation of each part of a model for TCP. Researchers compare directly the real throughput achieved by a TCP connection to the final result of their modeling. But as we saw, a model for TCP is composed of two parts: a model for the window evolution and a model for losses. Proceeding for the validation in one step hides the errors introduced by these two parts. It gives us the sum of the two errors instead of each of them separately. First, this prevents us from distinguishing from which part of the model the error is mainly due. Second, and most important, the errors introduced by the two parts of the model may be of opposite signs which may make the total error small and acceptable. The result will be a wrong estimation of the capacity of the model since, as we will see later, this phenomenon of error cancellation does not always exist.

To avoid the problem of error superposition and possibly error cancellation, we propose to validate separately the two

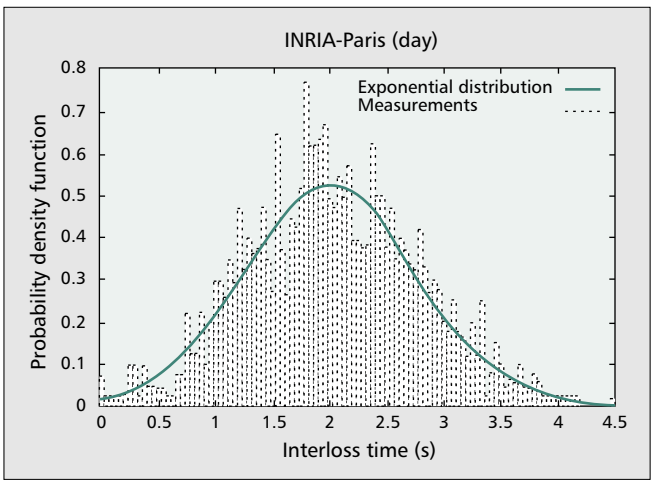


Figure 11. *MD connection: interloss time distribution.*

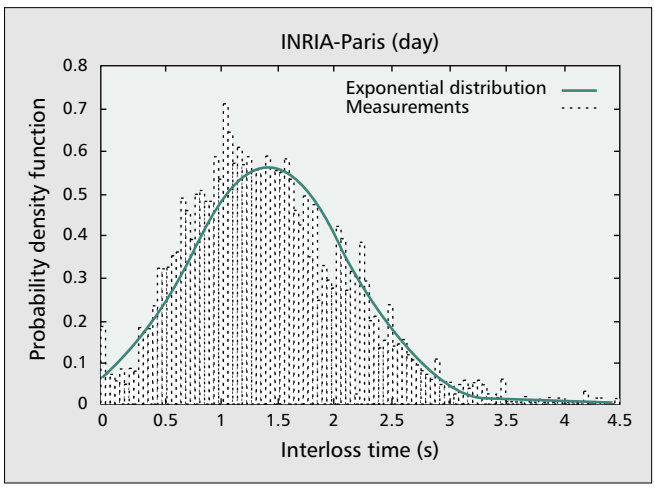


Figure 12. *MD connection: interloss time distribution.*

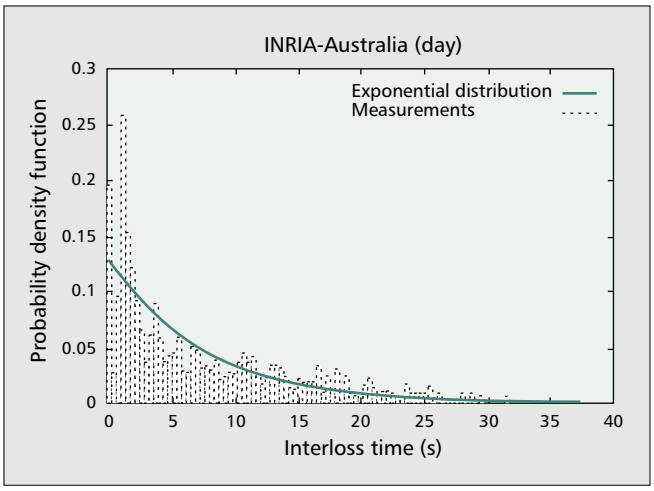


Figure 13. *LD connection: interloss time distribution.*

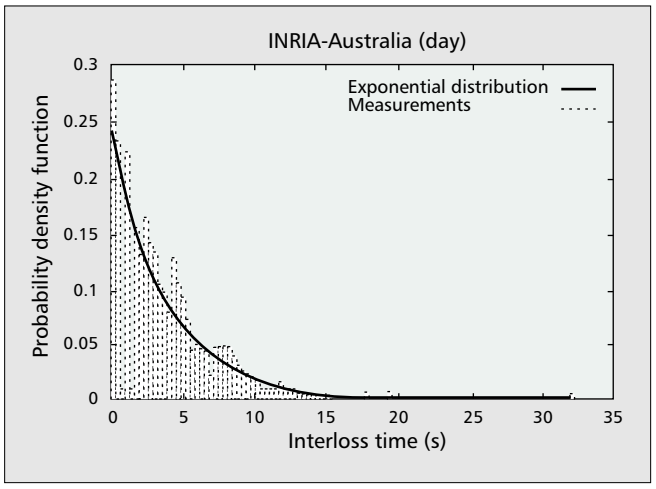


Figure 14. *LD connection: interloss time distribution.*

parts of the model. We start first with the model for losses. Consider that a fluid linear model is used for window evolution. To get only the error introduced by the assumption on the distribution of losses, the window of TCP should increase continuously and linearly between losses and decrease multiplicatively by a factor 0.5 upon losses. However, a TCP window does not have this ideal behavior in reality. What we do here is to construct this ideal behavior of the window using the moments of losses seen by the TCP connection. The average of RTT measurements is used for the calculation of the linear window increase rate. The ideal window is shown by the straight line in Fig. 4. We call the version of TCP that has the window behaving as the ideal window *ideal TCP* or the *exact fluid model*. Then we calculate numerically the throughput obtained by ideal TCP and compare it with the result of our modeling under a certain assumption on the loss process. The comparison gives us the error introduced by the model for losses.

The throughput of ideal TCP is calculated as follows. First, we sum over all the areas between the ideal window, the loss moments, and the time axis; then we divide this sum by the total time of the measurement. This gives the time average of the ideal window size. The ideal throughput is obtained by dividing the average of the ideal window by the measured average RTT. Now, to get the error introduced by the model for the window evolution, all what we need to do is to compare the ideal throughput to the real throughput. Before this comparison, the ideal throughput has to be corrected for timeouts and for the packet nature of TCP using the heuristics we outlined earlier.

We present some results to confirm the utility of such a method for validation. We take first the SD connection. We plot in Fig. 15 the ideal throughput we obtained during the different hours of the day. We also plot in the same figure the real throughput, the result of a linear fluid model assuming deterministic interloss times, and the result of the packet model in [8], which also considers that losses are deterministic. A model assuming deterministic losses underestimates the ideal throughput given the high variance of interloss times observed on the SD connection (Figs. 9 and 10). The ideal throughput in turn overestimates the real throughput due to the sublinear growth of the window discussed earlier. We notice that a direct comparison of the real throughput with the result of the modeling in case of deterministic losses hides the two errors and gives us an impression that the model works correctly.

We now take the LD connection (Fig. 16). The ideal throughput approximates well the real throughput given that the window increase on the LD connection is quite linear (discussed earlier). However, a model with deterministic losses does not give as good performance in this case as on the SD connection since the loss process is close to Poisson (Figs. 13 and 14). The assumption that losses form a Poisson process should give better performance. This difference in the perfor-

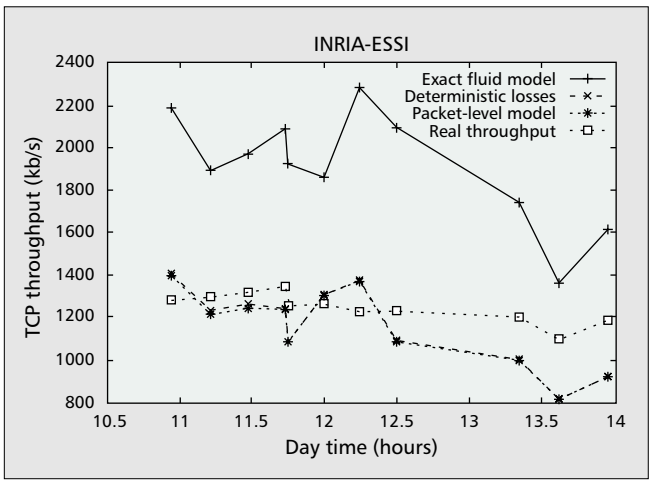


Figure 15. *SD connection: separate model validation.*

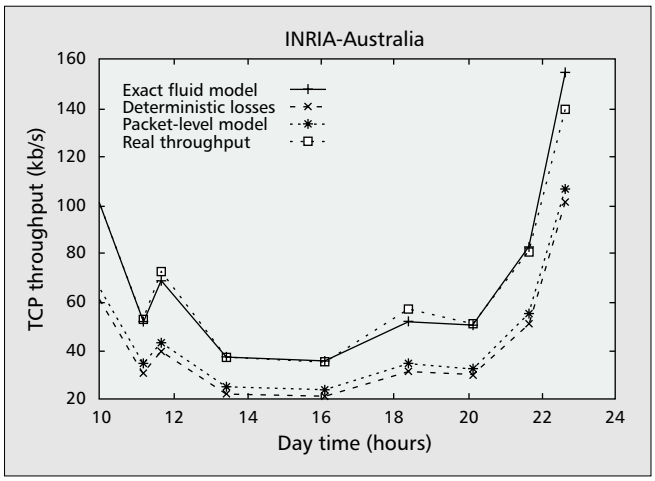


Figure 16. *LD connection: separate model validation.*

mance of a model with deterministic losses between the SD and LD connections cannot be explained without the method of validation we introduced. Using our method, we conclude that it is better to take the loss process as Poisson on the LD connection.

Conclusions

We present in this article an overview of the different issues to be considered when modeling TCP. Our main results can be summarized as follows:

- A linear window increase model does not hold on paths where the round-trip time is dependent on the window size. A sublinear window increase model needs to be considered in this case. The modeling of the sublinearity requires a model for the round-trip time which can be inferred from end-to-end measurements.
- The process of congestion events needs to be well characterized. A simplistic assumption may lead to considerable error on some Internet paths. In particular, underestimating the variance of time intervals between congestion events leads to underestimation of the throughput.
- The validation of a model for TCP needs to be done in two steps. The model for window evolution and the model for the network need to be validated separately. A one-step validation hides the error introduced by each part of the model, which may make the results inexplicable in some situations.

Acknowledgments

The author would like to thank Eitan Altman and Kostya Avrachenkov for their encouragement and their valuable remarks on an earlier version of this article. Grateful thanks are expressed to Prof. Colman Altman for his help in improving the presentation of the article. The author would also like to thank his colleagues at ESSI, ENST, and the University of South Australia for providing the required material to conduct the experiments.

References

[1] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, Nov. 1997.
[2] V. Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM, Aug. 1988.
[3] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss," *IEEE/ACM Trans. Net.*, vol. 5, no. 3, Jun. 1997, pp. 336–50.
[4] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *ACM Comp. Commun. Rev.*, Oct 1991.
[5] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. Net.*, Aug. 1993.
[6] E. Altman, K. Avratchenkov, and C. Barakat, "A Stochastic Model for TCP/IP with Stationary Random Losses," ACM SIGCOMM, Sept. 2000.
[7] M. Mathis *et al.*, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *Comp. Commun. Rev.*, July 1997.
[8] J. Padhye *et al.*, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," *ACM SIGCOMM*, Sept. 1998.
[9] T. Bu and D. Towsley, "Fixed Point Approximation for TCP behavior in an AQM Network," UMass CMPSCI tech. rep. no. 00-43, July 2000.
[10] V. Firoiu and M. Borden, "Queue Management for Congestion Control," IEEE INFOCOM, Mar 2000.
[11] A. Misra and T. Ott, "The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss," IEEE INFOCOM, Mar. 1999.
[12] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with Application to RED," ACM SIGCOMM, Aug. 2000.
[13] R. Morris, "Scalable TCP Congestion Control," IEEE INFOCOM, Mar. 2000.
[14] H. Chaskar, T. V. Lakshman, and U. Madhow, "On the Design of Interfaces for TCP/IP Over Wireless," IEEE MILCOM, 1996.
[15] S. Floyd and K. Fall, "Promoting the Use of End-To-End Congestion Control in the Internet," *IEEE/ACM Trans. Net.*, vol. 7, no. 4, Aug. 1999, pp. 458–72.
[16] S. Floyd, M. Handley and J. Padhye, "Equation-based Congestion Control for Unicast Applications," *ACM SIGCOMM*, Aug. 2000.
[17] C. Barakat, "Performance Evaluation of TCP Congestion Control," Ph.D. thesis, available at http://www.inria.fr/mistral/personnel/Chadi.Barakat/
[18] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *ACM Comp. Commun. Rev.*, Jul 1996.
[19] V. Misra, W.-B. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior," *Performance*, Oct 1999.
[20] M. Allman, H. Balakrishnan, and S. Floyd, "Enhancing TCP's Loss Recovery Using Early Duplicate Acknowledgment Response," Internet draft, June 2000, work in progress.
[21] E. Altman *et al.*, "State-dependent M/G/1 Type Queueing Analysis for Congestion Control in Data Networks," IEEE INFOCOM, Apr. 2001.
[22] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP Buffer Tuning," *ACM SIGCOMM*, Sept. 1998.
[23] M. Allman and A. Falk, "On the Effective Evaluation of TCP," *ACM Comp. Commun. Rev.*, vol. 29, no. 5, Oct. 1999.
[24] S. Savari and E. Telatar, "The Behavior of Certain Stochastic Processes Arising in Window Protocols," IEEE GLOBECOM, Dec. 1999.
[25] J. Nagle, "Congestion Control in IP/TCP Internetworks," RFC 896, Jan. 1984.

Biography

CHADI BARAKAT (cbarakat@sophia.inria.fr) received his degree in electrical and electronics engineering from the Lebanese University in Beirut in 1997. In 1998, he obtained a D.E.A. degree in networks and distributed systems from the University of Nice-Sophia Antipolis, France. Since 1998 he has been working toward a Ph.D. within the MISTRAL team at INRIA Sophia Antipolis, France. He obtained his Ph.D. in April 2001 and after that he joined the ICA department in EPFL, Lausanne. His main research interests are congestion and error control in computer networks, performance evaluation of communication protocols, and integration of new transmission media such as satellite networks into the Internet.