

TCP Model for Short Lived Flows

Marco Mellia, *Member, IEEE*, Ion Stoica, *Member, IEEE*, and Hui Zhang, *Member, IEEE*

Abstract—We propose a recursive, analytical model to predict the TCP performance in terms of completion time for short-lived flows. Based on the knowledge of the average dropping probability, the average roundtrip time and the flow length, the model provides very good results when compared to simulation.

Index Terms—HTTP flows, TCP, traffic modeling.

I. INTRODUCTION

ACCORDING to recent studies [1], TCP [2], [3] accounts for 95% of the total traffic volume, and 80% of the total number of flows in the Internet. Among the TCP flows, a large majority are short lived flows with the average and the median lengths no larger than 10 KB. These figures highlight the importance of understanding the behavior of short lived TCP flows.

In recent years, the modeling of the TCP behavior has received considerable attention [4]–[6]. However, most of the proposed models focus on characterizing the TCP performance in steady-state. In contrast, modeling the TCP behavior in the Slow Start phase was largely ignored. This letter addresses this problem, by presenting a simple TCP model to estimate the completion time of short lived flows in the presence of segment losses. Our model computes the average completion time as a function of the roundtrip time (RTT), the loss probability, and the flow length. To evaluate the model, we use a variety of simulations which show that the measured average completion times matches the predicted values very well.

II. ASSUMPTION

We consider the reference Slow Start algorithm implemented in BSDUNIX [7]. Since all major TCP proposals, such as TCP Reno, Newreno, and SACK implement the same Slow Start and similar loss recovery algorithms, our model accurately predicts the average completion times for all these TCP variants. For simplicity, we do not model the delayed acknowledgment mechanism.

We assume that the sender is limited only by the congestion window size, and not by the receiver window size. We consider a unidirectional data transfer of maximum nine data segments. This translates into a maximum transfer of 13.5 KB [assuming a maximum transfer unit (MTU) of 1.5 KB], which should be enough to cover most of the short-lived TCP flows. The short size of the TCP transfers considered in this paper has two immediate implications. First, many of the TCP flows never exit

the Slow Start phase. Second, the probability that a packet loss will trigger the Fast Recovery algorithm is very low. This is because the window size will be small during the entire duration of the transfer, and thus the sender will not be able to send enough segments to generate three duplicate acknowledgments.¹ As a result, most losses trigger a retransmission timeout (RTO) which will cause the TCP flow to reenter the Slow Start Phase.

We do not model the Congestion Avoidance algorithm, and consider that the congestion window always grows exponentially, i.e., the congestion window always increases by one upon receiving an acknowledgment. As demonstrated by the simulation results, this assumption has minimal effects on the accuracy of our model. The main reason for this is twofold. First, when the window size is very small, the increase of the congestion window under Slow Start is relatively close to the increase of the congestion window under Congestion Avoidance. Second, TCP enters the Congestion Avoidance phase only when the congestion window size is greater or equal to two [2], [3], [7].

We assume that all segments of a flow traverse the same path and that the loss rate is uniformly distributed, i.e., each segment is dropped with the same probability. Note that the assumption that the loss rate is uniformly distributed is likely to hold in practice for short lived flows. Indeed, in the case of high speed links, the correlation between subsequent losses affecting the same flow is highly reduced by the high statistical multiplexing on the link. In addition, modern buffer management schemes such as RED reduce further the probability of dropping consecutive segments of the same flow. The average RTT is assumed to be known. We consider the transmission time to be negligible compared to the propagation time. This allows us to ignore the dependencies between RTT and the segment size, and the queueing delay due to waiting for the previous segment of the same flow to be transmitted.

The main goal of this work is to estimate the *completion time*, i.e., the time it takes a source to successfully transfer a given amount of data using a TCP congestion control algorithm. A transfer is considered successful when the source receives the ACK for the last segment (we ignore the closing procedure).

The completion time consists of two terms:

- connection establishing time, i.e., the time it takes to successfully complete the three-way handshake algorithm [2], [3];
- time to complete the transfer of the actual data segments.

III. MODEL DESCRIPTION

To describe our model, we use the following variables:

- \mathcal{R} average RTT;
 p_s SYN segment dropping probability;

Manuscript received June 14, 2001. The associate editor coordinating the review of this letter and approving it for publication was Prof. I. S. Venieris.

M. Mellia is with the Dipartimento di Elettronica, Politecnico di Torino, Turin 10129, Italy (e-mail: mellia@mail.tlc.polito.it).

I. Stoica is with the Computer Science Division, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: istoica@cs.berkeley.edu).

H. Zhang is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: hzhang@cs.cmu.edu).

Publisher Item Identifier S 1089-7798(02)01920-8.

¹Fast Recovery can be triggered only when the congestion window is larger or equal to four segments, which can happen only when the flow has at least seven segments.

\mathcal{T}_s RTO for SYN segments (which is set to 3 s [2]);
 p data segment dropping probability, $q = 1 - p$;
 \mathcal{T} estimated RTO for data segments.

A. Connection Setup

During the connection setup, the TCP sender transmits one SYN segment and waits for the SYN-ACK segment. Upon receiving this segment, the sender acknowledges the SYN-ACK segment and starts sending data segments. At each stage of this process, if either party does not receive the ACK within \mathcal{T}_s , it retransmits the SYN segment and doubles the SYN timeout value. Thus, similarly to [6], the average connection setup time C_s can be expressed as

$$C_s = \mathcal{R} + (1-p_s) \sum_{i=1}^{\infty} p_s^i \sum_{j=1}^i 2^{j-1} \mathcal{T}_s = \mathcal{R} + \mathcal{T}_s \frac{p_s}{1-2p_s}. \quad (1)$$

Note that this holds only if $p_s < 1/2$, otherwise the sum does not converge. In practice, TCP gives up setting up the connection after a limited number of retries.

B. Data Transfer

Immediately after receiving the SYN-ACK, the sender starts transmitting data segments and waits for ACK segments. Upon sending a data segment, the sender initializes a timer \mathcal{T} . If the timer expires, the sender assumes that the data segment was lost, and it retransmits the original data segment. At the same time the value of the timer is doubled. An important point to note here is that, in accordance to most TCP implementations, here we assume that the sender maintains only one timer for all in-flight segments [3], [7]. The timer is associated to the segment with the lowest sequence number that has not been acknowledged yet. When an ACK segment is received, the timer is reset and re-assigned to the subsequent segment.

Next, we make two important observations:

- \mathcal{T} is modified during the evolution of the connection. It is initially set to three seconds, and updated upon receiving each ACK by using Karn's algorithm.
- the granularity of estimating the average and variance of RTT is typically very large, i.e., 500 ms.

On terrestrial links, the RTT is usually much smaller than the timer granularity, thus the RTT estimation algorithm converges to the minimum value immediately after that the first estimation (i.e., the first ACK) is received. Thus, in our model we consider $\mathcal{T} = 3$ s for the first data segment, and $\mathcal{T} = 1.5$ s for all the remaining ones.

Let C_m^w denote the average time spent to successfully send m segments with an initial congestion window of size w . Thus, C_n^1 is the average completion time of a flow of size n , and C_m^m is the average time to successfully transfer a *burst* of m segments that belong to the same congestion window. Obviously, $C_m^w = C_m^m$ for $w \geq m$. This allows us to derive a *recursive* model, that explicitly considers C_m^1 as a function of p , \mathcal{T} , \mathcal{R} and $C_{m'}^w$, $m' < m$.

Note that $C_n^1 = C_n^1 + C_{n-1}^2$, since after the TCP source receives the ACK for the first segment, it transmits the remaining $n - 1$ segments using an initial congestion window of two.

Next we derive C_m^m . If $m = 1$, the completion time is similar to (1):

$$C_1^1 = \mathcal{R} + q \sum_{i=1}^{\infty} p^i \sum_{j=1}^i 2^{j-1} \mathcal{T} = \mathcal{R} + \mathcal{T} \frac{p}{1-2p}.$$

With a burst of two segments we have

$$C_2^2 = \mathcal{R}q^2 + qp(\mathcal{T} + \mathcal{R} + C_1^1) + pq(\mathcal{T} + C_1^1) + p^2(\mathcal{T} + C_2^1).$$

Upon receiving the ACK of the first segment, the sender reschedules the retransmission timer. As a result, the detection of a possible loss of the second segment is delayed by one RTT. Similarly to the previous cases, we can write

$$\begin{aligned} C_3^3 &= q^3\mathcal{R} + 2q^2p(\mathcal{T} + \mathcal{R} + C_1^1) + pq^2(\mathcal{T} + C_1^1) \\ &\quad + qp^2(\mathcal{T} + \mathcal{R} + C_2^1) + 2p^2q(\mathcal{T} + C_2^1) + p^3(\mathcal{T} + C_3^1) \\ C_4^4 &= \mathcal{R}q^4 + pq^3(\mathcal{R} + C_1^1) + 3q^3p(\mathcal{T} + \mathcal{R} + C_1^1) \\ &\quad + 3q^2p^2(\mathcal{T} + \mathcal{R} + C_2^1) + 3p^2q^2(\mathcal{T} + C_2^1) \\ &\quad + qp^3(\mathcal{T} + \mathcal{R} + C_3^1) + 3p^3q(\mathcal{T} + C_3^1) + p^4(\mathcal{T} + C_4^1). \end{aligned}$$

Note that when evaluating C_4^4 , if the first segment in the burst is dropped, then the Fast Recovery algorithm will be triggered, and the source will retransmit the segment immediately without waiting for an RTO.

We next derive $C_{m'}^2$. If $m' = 3$, three packets have to be sent, with an initial congestion window of two. Thus, after the burst of two segments is sent, one of the following cases occurs. 1) No loss. The third packet is sent as soon as the first ACK is received. 2) Or the first packet in the burst is lost. The RTO triggers the packet retransmission, and the congestion window is reset to one. At this point the sender has still two packets to send. 3) Or the second packet in the burst is lost. Upon receiving the ACK for the first packet, the sender increases its congestion window and sends the third packet. At the same time, the retransmission timer is rescheduled. Since TCP uses cumulative acknowledgments and employs only one retransmission timer, the possible loss of the third segment can be detected only when the ACK of the second segment is received (assuming that the second segment is not lost). This fact in conjunction with the assumption that the transmission time is negligible allow us to consider the equivalent scenario in which the retransmission of the second segment and of the original segment happen at the same time, that is, the two segments are sent in a burst, and therefore the completion time is C_2^2 . 4) Or both packets in the burst are lost. After the retransmission timer expires, the sender starts all over again. The only difference is that now the retransmission timer value is double.

Thus:

$$C_3^2 = q^2(\mathcal{R} + C_1^1) + qp(\mathcal{T} + \mathcal{R} + C_2^2) + pq(\mathcal{T} + C_2^1) + p^2(\mathcal{T} + C_3^1).$$

Using the same reasoning as before, we can write

$$\begin{aligned} C_4^2 &= q^2(\mathcal{R} + C_2^2) + qp(\mathcal{T} + \mathcal{R} + C_3^2) \\ &\quad + pq(\mathcal{T} + C_3^1) + p^2(\mathcal{T} + C_4^1) \\ C_5^2 &= q^2(\mathcal{R} + C_3^3) + qp(\mathcal{T} + \mathcal{R} + C_4^2) \\ &\quad + pq(\mathcal{T} + C_4^1) + p^2(\mathcal{T} + C_5^1) \\ C_6^2 &= q^2(\mathcal{R} + C_4^4) + qp(\mathcal{T} + \mathcal{R} + C_5^2) \\ &\quad + pq(\mathcal{T} + C_5^1) + p^2(\mathcal{T} + C_6^1). \end{aligned}$$

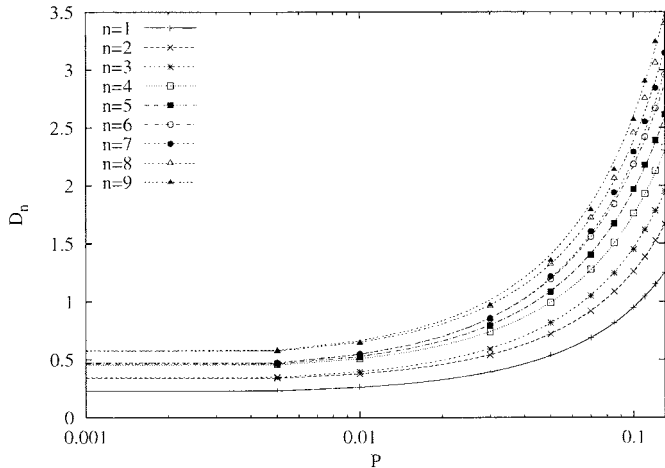


Fig. 1. Comparison between the proposed model and the simulation measurements: single source.

The Fast Recovery algorithm can be triggered when evaluating C_m^4 , $m > 4$, as the number of in-flight segments can exceed four. Thus

$$C_5^4 = q^4(\mathcal{R} + C_1^1) + 2pq^3(\mathcal{R} + C_2^1) + 2q^3p(T + \mathcal{R} + C_2^2) + 3q^2p^2(T + \mathcal{R} + C_3^1) + 3p^2q^2(T + C_3^1) + qp^3(T + \mathcal{R} + C_4^2) + 3p^3q(T + C_4^1) + p^4(T + C_5^1)$$

$$C_6^4 = q^4(\mathcal{R} + C_2^2) + 3pq^3(\mathcal{R} + C_3^1) + 2q^3p(T + \mathcal{R} + C_3^3) + q^2p^2(T + \mathcal{R} + C_4^3) + 2q^2p^2(T + \mathcal{R} + C_4^4) + 3p^2q^2(T + C_4^1) + qp^3(T + \mathcal{R} + C_5^2) + 3p^3q(T + C_5^1) + p^4(T + C_6^1)$$

and

$$C_7^2 = q^2(\mathcal{R} + C_5^4) + pq(T + C_6^1) + qp(T + \mathcal{R} + C_6^2) + p^2(T + C_7^1)$$

$$C_8^2 = q^2(\mathcal{R} + C_6^4) + pq(T + C_7^1) + qp(T + \mathcal{R} + C_7^2) + p^2(T + C_8^1)$$

This procedure can be repeated to evaluate C_m^w for $m > 9$, but the complexity grows exponentially due to all the possible combinations that must be taken into account. We note that this model can be completed with a steady state model to estimate the TCP performance of long lived flows, as proposed in [6].

Model Validation: We evaluate our model by simulation using the ns-2 simulator. In the first set of simulations, we consider a single TCP flow that transmits exactly n segments on a 50 ms bidirectional link that directly connects the source and the receiver. The link capacity is 10 Gbps, which makes the transmission time negligible. Data segments are randomly dropped with probability p . We repeat the experiment 100 000 times to get accurate results.

Using the model, we then evaluated the TCP performance in terms of completion time for different dropping probabilities, where $\mathcal{R} = 100$ ms and p is fixed. The measured and the estimated completion times are plotted in Fig. 1 as functions of the loss rate. As it can be seen, the model tracks very accurately the simulation results.

A second and more realistic scenario simulates WWW-like traffic pattern, where a number of TCP-RENO sources and receivers are connected via a bottleneck link of 10-Mbps capacity, and 50 ms propagation delay. All routers implement RED, with a maximum queue size of 50 segments. The flow arrival rates are

TABLE I
AVERAGE FLOW LENGTH PER GROUP (MTU = 576 BYTES)

Group	Length		Group	Length	
	[bytes]	[segments]		[bytes]	[segments]
0	61	1	5	4149	8
1	239	1	6	6358	12
2	539	1	7	10910	19
3	1349	3	8	18978	35
4	2739	5	9	90439	158

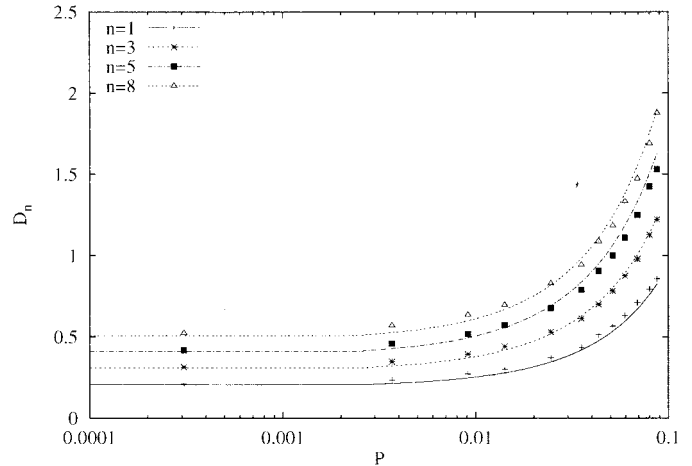


Fig. 2. Comparison between the proposed model and the simulation measurements: multiple sources.

modeled by Poisson processes. The flow lengths are drawn from the distribution shown in Table I, that was derived to best match the AT&T Labs' recent measurements of the Internet traffic [1]. Note that more than 60% of the flows are shorter than ten segments.

Finally, we run a variety of simulations under different offered loads, and measure the flow completion time and the dropping probability along the flow path. Fig. 2 presents a comparison between the measured and the predicted completion times. As it can be seen the predicted values are very close to the simulation results. Note that these results show that the TCP performance is mainly dominated by the RTO, observation that supports our assumptions.

IV. CONCLUSIONS

We have presented a TCP model for short-lived flows to estimate the completion time as a function of the average loss rate and the RTT along the flow path. The simulation results show that our model is highly accurate over a variety of scenarios.

REFERENCES

- [1] A. Feldmann, J. Rexford, and R. Caceres, "Efficient policies for carrying Web traffic over flow-switched networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 673–685, Dec. 1998.
- [2] J. Postel, "Transmission control protocol," RFC 793, Sept. 1981.
- [3] W. R. Stevens, *TCP/IP Illustrated*. Reading, MA: Addison-Wesley, 1994, vol. 1.
- [4] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM'98*, Sept. 1998, pp. 303–314.
- [5] C. Casetti and M. Meo, "A new approach to model the stationary behavior of TCP connections," in *IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, pp. 367–375.
- [6] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, pp. 1742–1751.
- [7] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.3 BSD UNIX Operating System*. Reading, MA: Addison-Wesley, 1989.