Performance evaluation of TCP connections in ideal and non-ideal network environments

Hala ElAarag, Mostafa Bassiouni*

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA

Received 7 July 2000; revised 17 January 2001; accepted 16 February 2001

Abstract

In this paper, we study the performance of TCP in both ideal and non-ideal network environments. For the ideal environments, we develop a simple analytical model for the throughput and transfer time of TCP as a function of the file size and TCP parameters. Our simulation measurements demonstrate that this model can accurately predict the throughput for ideal TCP connections characterized by no packet loss due to congestion or bit errors. If these ideal conditions are not met, the model gives an upper bound for throughput and lower bound for transfer time. For the non-ideal environments, we concentrate on wireless links. While our ideal model provides an easy to use tool to calculate bounds on the performance of all TCP implementations in such environments, we also show through simulation the relative performance of four well-known TCP implementations. We also present simulation results that demonstrate the dominant factors affecting the performance of wireless TCP. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: TCP; Network; Wireless networks; Performance evaluation; Modeling

1. Introduction

TCP is the most widely used transport protocol on today’s Internet. Nearly all applications that require reliable transmission (such as ftp, telnet, http, email, etc.) use TCP as their transport protocol.

There have been several implementations aiming at improving the performance of TCP since its evolution in 1981 [15]. The most important modification was the introduction of congestion control and avoidance techniques using the principle of self-clocking in Jacobson [10]. This version is usually referred to as Tahoe TCP. Tahoe TCP regulates the number of packets it sends by inflating and deflating a window according to the network requirements.

In order to do this, the TCP sender uses the cumulative acknowledgements (ACKs) sent by the receiver. If no packets are lost, TCP keeps inflating the window in three main phases: slow-start, congestion avoidance and maximum window [2,16]. In the ‘Slow-start’ phase, the TCP sender starts with a congestion window (cwnd) equals to 1. For each received ACK, TCP exponentially increases the window until it reaches a threshold (ssthresh), then it enters the ‘congestion avoidance’ phase where it continues to increase its cwnd linearly until it reaches the receiver’s maximum advertised window size. If this receiver’s advertised window does not change during transmission, the TCP window at the sender remains of constant size equal to the maximum. TCP continually measures how long acknowledgements take to return to the sender in order to determine the appropriate value of timeout and provide reliability by retransmitting lost packets.

Reno [11,17], New-Reno [9] and SACK [14] were designed to improve the performance of Tahoe TCP when packets are lost. However, when no packet losses occur they behave like Tahoe TCP. Thus in the ideal case of no packet losses, all these different implementations should have the same performance.

In Section 2, we develop a model to calculate the ideal TCP throughput. The model can be used to estimate upper bounds of throughput and lower bounds of transfer time in most standard TCP implementations. For networks having ideal conditions (no congestion and no bit errors), the bounds derived by the model are accurate predictions of the actual performance.

The increasing interest of mobile computers and the popularity of TCP on wired networks caused many researchers to investigate the usage of TCP on mobile networks. While in Section 2 we show that our model can
2. Analysis of throughput for ideal TCP

In this section, we calculate the throughput and transfer time of TCP in the ideal case where no losses occur. Fig. 1 shows the window size versus time for an ideal TCP connection. The window size is the size of the congestion window (cwnd) in packets. The three time periods shown in this graph correspond to the three phases: slow start, congestion avoidance and maximum window as explained earlier.

Let \( \theta \) be the throughput of ideal TCP (packets/s), \( N \) the total number of packets transmitted, \( T \) the total transfer time (s), \( W_s \) the ssthresh (in packets), \( W_{\text{max}} \) the maximum window size (in packets), RTT the round trip time (s), \( T_1 \) the length of time of the slow-start phase (s), \( T_2 \) the length of time of the congestion avoidance phase (s), \( T_3 \) the length of time when \( W \) is equal to \( W_{\text{max}} \), and \( N_i \) is the number of packets transmitted in \( T_i \), where \( i = 1, 2, 3 \).

Note that the window size \( W \) during the slow-start phase is \( 1 \leq W \leq W_s \), during the congestion avoidance phase is \( W_s < W \leq W_{\text{max}} \), and during the maximum size phase is \( W = W_{\text{max}} \).

Assume that:

- RTT is relatively larger than the transmission time of any window. This is a valid assumption in many WAN environments.
- \( W_s \) is a power of 2, which is not unusual in many TCP publications (e.g. Hoe [9]).

The throughput \( \theta \) is given by

\[
\theta = \frac{N}{T} \tag{1}
\]

where

\[
T = T_1 + T_2 + T_3
\]

It is easy to calculate \( T_1, T_2 \) and \( T_3 \). They are equal to the number of round trips in each phase multiplied by RTT. In the slow-start phase, the window starts with a value of 1 then it is doubled every RTT until it reaches \( W_s \). Thus the number of RTT in \( T_1 \) is equal to \( \log W_s + 1 \). Note that \( \log \) is log to the base 2. Then \( T_1 \) is equal to

\[
T_1 = (\log W_s + 1) \text{RTT}
\]

In the congestion avoidance phase, the window is increased by one every RTT until it reaches \( W_{\text{max}} \), thus \( T_2 \) is equal to

\[
T_2 = (W_{\text{max}} - W_s) \text{RTT}
\]

In \( T_1 \) windows of size \( W_{\text{max}} \) are transferred until the end of the file. Thus \( T_3 \) is equal to

\[
T_3 = \frac{(N - N_1 - N_2)}{W_{\text{max}}} \times \text{RTT}
\]

Note that the above analysis assumes that the slow-start phase includes windows of sizes 1 through \( W_s \), and that the congestion avoidance phase includes windows of sizes \( W_s + 1 \) through \( W_{\text{max}} \).

The number of packets transmitted in each phase can be easily calculated as follows:

\[
N_1 = 1 + 2 + 4 + 8 + \cdots W_s
\]

Thus

\[
N_1 = 2^{\log W_s + 1} - 1 = 2 \times W_s - 1
\]

\[
N_2 = (W_s + 1) + (W_s + 2) + \cdots (W_s + (W_{\text{max}} - W_s - 1))
\]

Thus

\[
N_2 = (W_{\text{max}} - W_s - 1) \left( \frac{W_{\text{max}} + W_s}{2} \right)
\]

Short files containing \( N \leq N_1 \) packets are completely transmitted during the slow-start phase. Files with \( N \leq N_1 + N_2 \) packets are completely transmitted before the end of the congestion avoidance phase. We call the
TCP performance for such cases a ‘transient’ performance. The throughput analysis is given below.

2.1. Transient behavior

Case 1. $N \leq N_1$

This case corresponds to the slow-start phase. The sender starts with a window of size 1 packet and then doubles its window after each round-trip. The exponential growth of the window in this phase leads to the following value of throughput (note that the logarithm uses base 2)

$$\theta(N) = \frac{N}{[\log N + 1]RTT}$$  \hspace{1cm} (2)

where $[x]$ is the smallest integer greater than $x$

Case 2. $N_1 < N \leq N_1 + N_2$

This case corresponds to the congestion avoidance phase, which has a linear growth of window size. After reaching the threshold window size $W_s$ at the end of the slow-start phase, the sender increases the window size by one packet in each round trip. Thus the successive window sizes in the slow-start phase are $1, 2, 4, \ldots, W_s$ while those in the congestion avoidance phase are $W_s + 1, W_s + 2, \ldots$. This leads to the following equation for throughput where the first term $\log(W_s + 1)$ of the denominator corresponds to the number of round trips of the completed slow-start phase and the second term (solution of a quadratic equation) is the number of round-trips in the ongoing congestion avoidance phase

$$\theta(N) = \frac{N}{([\log W_s + 1] + \sqrt{(2W_s + 1)\sqrt{8(N - 2W_s + 1) - (2W_s + 1)/2}]}RTT$$  \hspace{1cm} (3)

2.2. Steady state

Case 3. $N > N_1 + N_2$

This case corresponds to the steady-state phase, which starts when the congestion avoidance phase increases the window to the maximum value $W_{\text{max}}$. Notice that the number of round trips in the completed congestion avoidance phase is simply $W_{\text{max}} - W_s$. The equation for throughput has a similar logic to that of Eq. (3), but the denominator has three terms corresponding to the completed slow-start and congestion avoidance phases and the ongoing steady-state phase

$$\theta(N) = \frac{N}{[\log W_s + 1 + W_{\text{max}} - W_s + (1/W_{\text{max}})[N - 2\log W_s + 1 + 1 - (W_{\text{max}} - W_s - 1) \times (W_{\text{max}} + W_s)/2]}RTT$$  \hspace{1cm} (4)

if $N \gg W_{\text{max}}$, i.e. $N \to \infty$.

From Eq. (4), applying L’Hospital’s rule:

$$\lim_{N \to \infty} \theta = \frac{W_{\text{max}}}{RTT}$$  \hspace{1cm} (5)

Note that the steady state throughput given in Eq. (5) is only realizable if it is less than the maximum link bandwidth.

From the above analysis, we can also obtain the ideal transfer time to transmit $N$ packets. This transfer time would simply be the denominator of Eqs. (2)–(4) for cases 1–3, respectively.

The transient behavior is applicable to many TCP connections, which tend to be relatively short. This agrees with the observed profile of most web traffic. Internet connections transferring small amounts of data (especially at light traffic conditions) have better chance of encountering ideal TCP performance than connections transferring large amounts of data.

2.3. Experimental validation of the ideal TCP model

In this section, we apply our model to the network topology shown in Fig. 2. This topology is motivated by the many recent experiments of TCP performance over wireless and wired networks. Many researchers [3,4,12] used it for their simulations of wireless networks, where they had n0 as a fixed host, n1 as a base station and n2 as a mobile host; the link between n0 and n1 is wired, while that between n1 and n2 is wireless. Fall and Floyd [FAL96] used this configuration on wired networks to compare Reno, New-Reno and Sack TCP where n1 was a finite-buffer drop-tail gateway and n2 was a wired data receiver.

For our simulation, we chose the link between n0 and n1 to be of bandwidth 1.5 Mbps and delay (D1) of 10 ms, while the link between n1 and n2 to be of bandwidth 0.8 Mbps and delay (D2) of 100 ms. There is a DropTail queue at n1 and n2. We have a TCP connection between n0 and n2 where n0 is the source and n2 is the sink. The connection is used to transfer bulk data via ftp from n0 to n1. The intermediate node n1 merely transfers the data received between n0 and n2. We used Tahoe TCP for this experiment but the ideal performance applies for the other TCP implementations mentioned in Section 1. For the TCP parameters, we set the maximum window size to 50 and ssthresh to 32. The packet size is chosen to be 1024 bytes, while ACKs are set to 40 bytes. The queue sizes are set to 50 (which is equal to the maximum window size) to avoid
any drop of packets due to lack of buffer space. No error models are applied to the packets to avoid retransmission of corrupted packets. The resulting connection reflects the ideal TCP environment analyzed earlier in Section 2.

Fig. 3 shows the estimated and measured results for the ideal TCP throughput $\theta$ versus the total number of packets transmitted $N$. We show two estimated results. In the first, we use a fixed value for RTT, computed from the following equation:

$$\text{RTT} = \sum_{i=1}^{4} \mu_i + 2 \times (D_1 + D_2)$$  \hspace{1cm} (6)

where $\mu_i$ is equal to the service time computed as the packet or ACK size divided by the link bandwidth. Specifically, $\mu_1$ is the service time to transmit a packet at $n_0$, $\mu_2$ the service time to transmit a packet at $n_1$, $\mu_3$ the service time to transmit an ACK at $n_2$, $\mu_4$ the service time to transmit an ACK at $n_1$ and $D_1$ and $D_2$ are the delays on the links as shown in Fig. 2.

In the second computation, we used the average RTT as sampled by the connection itself. Mathis et al. [13] adopted those two estimates for RTT in their model of TCP congestion avoidance algorithm. The $x$-axis in Fig. 3 represents the values for $N$ (the number of packets in the file). The range $N = 8–32$ corresponds to the slow-start phase, $N = 64–512$ corresponds to the congestion avoidance phase, while $N = 1024–5000$ corresponds to the maximum window phase. We notice that, when $N$ is small (less than 64), the measured value is exactly equal to the estimated value with fixed RTT. As $N$ increases, the queuing delay during the transmission of large windows increases and the estimated throughput with fixed RTT becomes larger than the measured one. The estimated throughput using sampled RTT provides closer values to the measured one. When $N = 5000$ the estimated throughput with fixed RTT is way off, while the estimated throughput with sampled RTT is still close to the measured throughput. Fig. 4 shows the estimated and measured throughput as represented by the number of packets per RTT (i.e. throughput × RTT). The estimated throughput in Fig. 4 uses sampled RTT values.

In addition to calculating the ideal throughput, we can
also get an estimate on the upper bound of the average window size in a connection given that the packet size is constant. This can be easily calculated from Fig. 4.

$$\text{average window size} = \frac{\theta \times \text{RTT}}{\text{packet size}}$$

2.4. Using the ideal TCP model to compute performance bounds

2.4.1. In wired networks

The throughput model presented in Section 2 gives an upper bound for TCP in congested wired networks. For the topology in Fig. 2, we varied the buffer size at the intermediate node n1. We simulated Tahoe and Reno TCP. We transferred a file of size $N = 512$. Fig. 5 shows the throughput of TCP versus buffer size at n1. For buffer sizes 50, 40 and 30, no packets are dropped and hence the model gives an accurate estimate of throughput for this buffer size range. At buffer size = 20, we noticed that 17 packets are dropped in TCP Tahoe while 11 are dropped in TCP Reno. The model provides an upper bound on the throughput for the two cases of congestion, i.e. buffer sizes of 20 and 10.

Fig. 6 shows the throughput for different file sizes when the buffer at n1 has a size of 20 packets. For short file transfers, i.e. file sizes $N = 8$, 16 and 32 there are no packets lost. Tahoe and Reno TCP had similar performance and both achieved the ideal throughput. At $N = 64$, packets got dropped and one lost packet caused Reno to suffer a timeout. That is why throughput of Reno is less at this particular file size. Otherwise throughput of Reno is usually better than Tahoe.

2.4.2. In wireless networks

We modeled the network topology of Fig. 2 where n0 is the fixed host (FH), n1 the base station (BS) and n2 is the mobile host (MH). In this paper, we assume the base station is a conventional stationary base station [7], i.e. is not a mobile base station used in conjunction with totally mobile wireless
networks [5]. The link between FH and BS is wired, while that between BS and MH is wireless. The wireless link is lossy and suffers from relatively high bit error rate. Bit errors on wireless links are usually bursty and frequent. Studies indicate that BER may be in the order of $10^{-5}$ [8]. Forward error correction codes (FEC) and retransmission schemes at the link layer may reduce the BER by one or two orders of magnitude [1]. In the tests for this experiment, we assumed packet retransmissions are due to bit errors in the wireless link and not due to buffer congestion in the intermediate node. Fig. 7 shows that the ideal TCP model gives a close upper bound for throughput of TCP for short file transfers ($N = 100-500$). Fig. 8 shows that if the BER is reduced by about two orders of magnitude, the actual performance closely follows the ideal upper bound even for somewhat longer file transfers ($N = 1000-5000$).

Similarly, we can show that the ideal model provides lower bounds of the transfer time. The lower bound of the transfer time can be obtained from Figs. 5–8 by dividing the number of packets by the throughput.

$$T = \frac{N}{\theta}$$

### 3. Performance of TCP in non-ideal environments having wireless link

All indications show that mobile computers and their wireless links will be an integral part of the future Internet. We showed in Section 2.4.2 that our model could be easily used to provide performance bounds of any end-to-end TCP implementation in wireless environments. In this section, we study the relative performance of four TCP implementations (Tahoe, Reno, New-Reno, and SACK) in such environments. Unlike wired environments, wireless environments suffer from high bit error rates and handoff disconnections. Our performance tests also show that the performance of TCP is more sensitive to the length of the link-up period than the error rate of the wireless link, or the mobile disconnection probability. All results in this section are based on simulation tests and not on the analytical model for ideal TCP.

While SACK TCP is well known to perform better than Reno TCP in wired networks, the tests show that this is not always the case in mobile wireless networks. Figs. 9 and 10 show the throughput and average packet delay, respectively, for the four considered TCP implementations over a period.
of 600 s. Fig. 11 shows the transfer time versus the number of packets. Figs. 9–11 show that Reno had the highest throughput, the least average packet delay and the least transfer time. In terms of throughput and the least transfer time, the implementations were ordered as Reno followed by New-Reno, then SACK, and finally Tahoe.

In addition to bit errors, wireless links can suffer occasional disconnection periods (also called link-down periods) during which the mobile host is effectively disconnected from the base station, e.g. due to handoff [7]. Let $m_1$ denote the mean length of the connection (linkup) period and $m_2$ denote the mean length of the disconnection (link-down period). The disconnection probability of the wireless link is defined as $m_2/(m_1 + m_2)$.

We ran experiments on the four TCP implementations we mentioned earlier. For each implementation we had two scenarios: scenario1 had a disconnection probability $P = 9\%$ ($m_1 = 1.0$ s, $m_2 = 0.1$ s), and scenario2 had a disconnection probability $P = 28.6\%$ ($m_1 = 5.0$ s, $m_2 = 2.0$ s). Table 1 shows the results for Tahoe TCP. Simulations are run for 1000 s.

The disconnection probability in Scenario2 is more than three times larger than that of Scenario 1. Nevertheless, we find that the throughput of Scenario2 is about five times better than that of Scenario1. The transfer time of Scenario2 is about 8.4 times smaller than that of Scenario1. Thus the performance of the network does not only depend on the disconnection probability but more on the length of the mean of the connection period. Scenario2 had a larger connection period mean, and that is why it resulted in a better performance, even though it had an unrealistically high disconnection probability. Figs. 12 and 13 illustrate the performance of Reno TCP when $P = 9$ and 28.6\%, respectively.

The graphs are generated by tracing the packets between FH and MH. The $x$-axis shows the time of arrival or departure of the packets. The $y$-axis shows the packet number mod 100. The square on the graph represents the enqueueing of packet at FH. The circle represents lost ACKs. All other

Fig. 9. Throughput of TCP implementations.

Fig. 10. Packet delay of TCP implementations.
ACKs are assumed to be received by FH. The × on the graph represents the dropped packets. Dropped packets and ACKs are due to the disconnection of MH or BER of wireless link.

For the sake of illustration, Figs. 12 and 13 show a trace of 10 s using a simple cyclic pattern for connect/disconnect times as follows: the simulation starts when the mobile is connected. It stays connected for 1 s in case ofFig. 12 and 5.0 s in case of Fig. 13, then disconnects for 0.1 s in case of Fig. 12 and 2.0 s in case of Fig. 13. In Fig. 12, we notice that the reason for the bad performance of TCP in Scenario 1 is that the sender does not have enough time to send packets to MH before it disconnects. Although it disconnects for a very short time, by the time the sender starts sending again, the mobile disconnects again, and so on. The short connection time (1 s) and the frequent disconnections made FH unable to open up a large window and send enough packets to MH. On the other hand, in Scenario 2, we notice from Fig. 13, that there is a long connection time for FH to send packets before the mobile disconnects. Even though, the mobile disconnects for a long period (2 s), but the frequency of the disconnection in this scenario was much less than the first.

Fig. 11. Transfer time (in s) versus # of packets of TCP implementations.

Fig. 12. Reno TCP with $P = 9\%$. 
Table 1
Throughput, Goodput, and Transfer time for Tahoe TCP with \( P = 9\% \) and \( P = 28.6\% \)

<table>
<thead>
<tr>
<th></th>
<th>Scenario 1 ( P = 9% )</th>
<th>Scenario 2 ( P = 28.6% )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (packets/s)</td>
<td>5.275</td>
<td>26.671</td>
</tr>
<tr>
<td>Goodput</td>
<td>0.8768</td>
<td>0.96</td>
</tr>
<tr>
<td>Transfer time for 5000</td>
<td>138.734</td>
<td>16.4604</td>
</tr>
<tr>
<td>packets (s)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For a disconnection probability of 9\%, we varied the value of \( m_1 \) and found that the larger the mean of the connection period the better the throughput, the goodput and the transfer time. The performance of TCP with a disconnection probability of 9\% can be better than that with a disconnection probability of 28.6\% only with a suitable mean of connection period. This is shown for Reno TCP in Figs. 14–16 whose x-axis represents the mean value of the connection (linkup) period.

Fig. 13. Reno TCP with \( P = 28.6\% \).

Fig. 14. Throughput of Reno TCP versus \( m_1 \) for \( P = 9\% \).
experiments for Tahoe, New-Reno and SACK TCP and they provided similar results. Fig. 14 shows that for the same disconnection probability, the higher the mean of the connection period, the larger the throughput of TCP. The figure also shows the throughput of TCP with disconnection probability $P = 28.6\%$ (shown as a separate dot at mean connection of $m_1 = 5$). The value of this throughput is 26.67, which is better than the throughput with $P = 9\%$ and $m_1$ less than 3. Fig. 15 shows that for $P = 9\%$, the higher $m_1$ is, the higher the goodput of the network. The test with $P = 28.6\%$ ($m_1 = 5$, $m_2 = 2$) results in goodput of 0.96, which is better than the goodput with $P = 9\%$ and $m_1$ less than 5. At fixed disconnection probability, the utilization of the network is always better with higher values of $m_1$. In Fig. 16, we show the effect of $m_1$ on the transfer time for sending 5000 packets. The higher $m_1$, the less the transfer time. From Fig. 16 one can notice that the value of the transfer time becomes constant (52.29 s) for $m_1$ greater than or equal to 5. In this case, the TCP connection does not suffer any disconnection while sending the 5000 packets. For $P = 28.6\%$ ($m_1 = 5$, $m_2 = 2$) the transfer time was also 52.29 s. This means that the transfer time is highly sensitive and dependent on the value of $m_1$.

We have noticed from our simulation tests, that the disconnection of the mobile has a greater impact on the performance of TCP than the effect of BER. Disconnections, generally can be of the order of several seconds. They can even last for 1 min [6]. This results in the loss of a large number of contiguous data packets and ACKs, causing frequent timeouts of the TCP connection and resulting in a degradation of its performance. Combining this observation with the results of the effect of the connection period, we can conclude that the connection period has a dominant effect on the performance of wireless TCP.

4. Conclusions

In this paper, we presented a simple approach to calculate important performance metrics for ideal TCP. Our model provides an upper bound for the throughput and a lower bound for the transfer time for any standard TCP implementation. Our experimental validation proves the accuracy of the model. For the non-ideal environments, we concentrated on wireless links and presented simulation results that show the relative performance of four standard TCP implementations. While SACK TCP is well known to perform better than Reno TCP in wired networks, our tests show that Reno TCP can outperform SACK TCP. Our simulations demonstrate that the performance of wireless TCP does not only depend on the disconnection probability of the mobile but more on the mean length of the connection period.

Acknowledgements

The authors are grateful to the anonymous reviewers for
their comments and suggestions that helped improve the quality of this paper. This work has been supported by ARO under grant no. DAAH04-95-1-0250 and by an I-4 corridor grant from the State of Florida and Harris Corporation. The views and conclusions herein are those of the authors and do not represent the official policies of the funding agencies or the University of Central Florida.

References