

Modelling of SACK TCP and application to the HTTP File Transfer Environment

Damien Phillips and Jiankun Hu, {damphil, jiankun}@cs.rmit.edu.au
School of Computer Science and Information Technology, RMIT University
GPO Box 2476V, Melbourne 3001, Australia

Abstract

It is known that analytic modelling for TCP latency is a non trivial task. Recently, some significant progress has been made, such as the comprehensive result by Sikdar et al. However, models similar to these often rely on trial and error methods such as "data fitting". This can lead to a very limited scope for the resulting model and also large estimation error. In this paper, we propose improvements to Sikdar's SACK TCP model. A new delayed acknowledgement slow start model is developed that is analytically derived from the slow start algorithm which provides a novel mechanism to model the relationship between RTT and the delayed acknowledgement timer. We introduce a simple mechanism to include time taken to send an HTTP Get Request to broaden the scope of our SACK TCP model to website file transfer. Simulation and live Internet experimentation has validated our scheme.

1 Introduction

Recently, [5] has provided comprehensive analytic models for TCP behaviour. In [5], new analytic models are proposed to estimate the latency and steady-state throughput of TCP Tahoe, Reno and SACK. However, their model for slow start introduces errors for the sum of segments sent in slow start [4]. We also show that the slow start model underestimates the window increase pattern for larger windows. We introduce the ability to model the HTTP Get request to predict the transfer times of common webpage file downloads.

2 Development of SACK TCP Model

The proposed modular structure includes connection, request, transfer, and our SACK-recovery phase models. We use a similar structure as in [5] to give the final expected

number of rounds to transmit N segments, and apply a fixed RTT to find the expected time.

The TCP connection establishment model is given in [1]. The timeout model is given in [3]. The congestion avoidance model is equivalent to that in [5]. The Get request for a particular file from a webserver involves sending a string identifying the required file, before the server can begin to process the request and send the file. This request is also subject to packet loss, and as such, we model this request equivalent to connection establishment.

2.1 Slow Start

Due to the random nature of Internet traffic, it is impossible to provide a deterministic expression for the window size pattern of delayed ACK slow start.

[5] presents a delayed ACK slow start model that attempts to model the expected value of $cwnd$ for any round, by averaging the case for delayed ACK timer fired and not. However, we note that the model in [5] must start with a fixed iw of one and is based on a $\sqrt{2}$ increase ratio. At higher congestion windows, this can significantly underestimate the window increase in later rounds, such that a 30% error can be accumulated in 5 rounds. As shown in [4], this has implications for other aspects of the models used in [5], such as the statistical impossibility of time out in one section of their model.

To solve the problem of random delayed ACK timer firing, we introduce bounding to the delayed ACK window size increase dependent upon how often the delayed ACK timer is fired. The lower bound states that delayed ACK timer is never activated apart from $iw = 1$, while the upper bound states that delayed ACK timer is activated for the last odd segment in a window. The models for delayed ACK slow start are derived from two recursive definitions given by

$$\begin{array}{l} \text{Fired ACK} \\ seg_{n+1} = 1.5seg_n + 0.5 [seg_n \text{ is odd}] \end{array} \quad (1)$$

$$\begin{array}{l} \text{Non-Fired ACK} \\ w_{n+1} = w_n + \lfloor \frac{w_n - 1}{2} \rfloor \end{array} \quad (2)$$

with the notation $[condition]$ to be equal to 1 when $condition$ is met and 0 when not. By observing that $[condition]$ and $[x]$ can be imitated by values proportional to $(-1)^x$, the models for the slow start delayed ACK fired and non-fired modules were completed. The non-fired ACK model is given by (Note that the exception cases are omitted as they are simple to derive, and that the Fired ACK case is omitted as it is derived in a similar fashion to the Non-Fired ACK case.)

$$\begin{aligned} c_{iw} &= iw - 0.5 \mp_{iw} 0.1 + \frac{2}{3} [iw = 1] + fa_{iw} \\ fa_{iw} &= \{0.0148, 0.0223, 0.0334, 0.0501, \\ &\quad -0.1741, -0.1248, -0.0611\} \\ sqn_k &= \begin{aligned} &[iw + c_{iw} (2) (1.5^k - 1.5) \\ &\mp_{iw} (0.375 \mp_k 0.375) \\ &+ [iw = 1] + 0.5] \end{aligned} \quad n > 1 \end{aligned} \quad (3)$$

$$k_{wm} = \left[\log_{\frac{3}{2}} \left(\frac{2.25W_{max} - 3.375}{c_{iw}} \right) + \frac{0.75}{W_{max}} \right] \quad W_{max} > iw + 1 \quad (4)$$

$$k_{exp} = \left[\log_{\frac{3}{2}} \left(\frac{N - \frac{1}{2} - iw - [iw=1] \pm_{iw} \frac{3}{8}}{2c_{iw}} \right) + 1.5 \right] \quad N > iw + 1 \quad (5)$$

where \pm_x indicates $+$ for odd x and $-$ for even x , being a shorthand for $-(-1)^x$, while \mp_x indicates $-$ for odd x and $+$ for even x , being a shorthand for $+(-1)^x$.

The time required to send N segments becomes (and, if required, adding one duration of delayed ACK timeout, or limited by W_{max})

$$T_{ss} = \begin{cases} RTT \left[k_{wm} - 1 + \lceil \frac{N - N_{exp}}{W_{max}} \rceil \right] & N > N_{exp} \\ RTT [k_{exp}] & \text{otherwise} \end{cases} \quad (6)$$

2.2 SACK Recovery

On detection of a lost segment in TCP SACK [2], $flightsize = N_{loss} + N_{follow} + N_{newround}$, which is derived from the loss pattern and the whether the loss occurred in slow start or congestion avoidance. The number of rounds to recover N_{loss} (k) and the number of segments sent in this phase (k') is given by

$$k = \max \left(\left\lceil \log_2 \left(\frac{N_{loss}}{N_{extra} + 1} \right) \right\rceil, 0 \right) \quad (7)$$

$$k' = \min \left((N_{extra} + 1) (2^k - 1), N_{remain} - N_{extra} \right) \quad (8)$$

$$N_{remain} = N_{remain} - N_{extra} \quad (9)$$

The total time for SACK recovery after a successful fast retransmit or timeout is given by

$$\begin{aligned} &\text{Fast Recovery} \\ N_{remain} &= N_{remain} - k' + N_{loss} \end{aligned} \quad (10)$$

$$T_{recovery} = RTT [N_{newround} > 0 \text{ and } N_{follow} < 3] + RTT (k + 1)$$

$$\begin{aligned} \text{iff } n_{loss} - 2 < ssthresh \text{ and} \\ N_{follow} + N_{newround} >= 3 \\ \text{Timeout} \end{aligned} \quad (11)$$

$$k' = \min (sqn (k_{wm} (ssthresh) - 1), N_{remain} + N_{loss}) \quad (12)$$

$$N_{remain} = N_{remain} - k' + N_{loss} \quad (13)$$

$$T_{recovery} = RTT [N_{newround} > 0 \text{ and } N_{follow} < 3] + E [TO] + T_{ss} (k', ssthresh) \quad (14)$$

2.3 Total Transfer Time

All TCP flows, of N segments in length, commence data transfer in slow start and, dependent upon the number of loss indications M , progress through a number of recovery and congestion avoidance phases, equal to M . These loss cases and total time, given a loss indication probability p , are given by

$$T_{1loss} = E [T_{ss} (i) + T_{recovery} (i) + T_{ca} (N_{remain})] \quad (15)$$

$$T_{multloss} = T_{1loss} (2D_a) + (M - 1) E [T_{recovery} (D_a) + T_{ca} (N_{remain})] \quad (16)$$

$$T_{transfer} = T_{setup} + \sum_{M=0}^N C_M^N p^M (1-p)^{N-M} T_{Mloss} \quad (17)$$

where T_{Mloss} is the time taken to transfer N segments dependent upon the number of losses M , and D_a is the average duration between the number of losses M . This result differs from [5] where a total transfer time is presented such that the total probability applied to the possible M loss indication cases exceeds one.

3 Simulation and Experimentation

NS-2 simulations were conducted to verify the ability of our model to predict the transfer times of one way transfers. Traces of HTTP transfers over the Internet were also captured. Our model was able to predict the transfer times of both the simulation and experimental results.

References

- [1] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM (3)*, pages 1742–1751, 2000.
- [2] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.
- [3] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling tcp reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking (TON)*, 8(2):133–145, 2000.
- [4] D. Phillips, J. Hu, B. Lloyd-Smith, and R. Harris. A note on an analytic model for slow start in TCP. In *ICON 2003*, 2003.
- [5] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK. *IEEE/ACM Trans. Netw.*, 11(6):959–971, 2003.