TCP's Retransmission Timer and the Minimum RTO

Alae Loukili, Alexander L. Wijesinha, Ramesh K. Karne, and Anthony K. Tsetse
Department of Computer & Information Sciences
Towson University
Towson, MD 21252, U.S.A.
{aloukili, awijesinha, rkarne, atsetse }@towson.edu

***Abstract*-We study the performance impact of recently recommended TCP retransmission timer settings using a bare PC Web server with no operating system or kernel running in the machine. We evaluate server performance in a test LAN with various settings of the alpha and beta constants used for computing SRTT and RTTVAR in the presence of varying levels of background traffic generated by conventional systems. We also study performance with different minimum RTO settings, and compare the performance of the bare PC Web server using the recommended timer settings with the performance of the Apache and IIS Web servers running on Linux and Windows respectively. We find that 1) no combinations of alpha and beta, or sampling strategies, perform consistently better than others under the different levels of background traffic; 2) lower minimum RTO settings than the recommended 1-second minimum will work when there is moderate background traffic, but the 1-second minimum is best when there are higher levels of congestion; and 3) using the standard timer settings but not using the TCP SACK option and congestion control mechanisms degrades bare server performance for some levels of background traffic.**

Keywords: Retransmission timer, TCP, networking, congestion control

I. INTRODUCTION

The retransmission timer is a key element of TCP. It is used to set the retransmission timeout (RTO) value that determines when TCP should retransmit. The RTO needs to be set before initiating a connection and subsequently updated based on the smoothed round-trip time (SRTT) and the round-trip time variation (RTTVAR). The latter values are computed using round-trip time (RTT) measurements. The latest IETF recommendation for setting the retransmission timer given in [1] reduces the initial RTO setting from 3 seconds to 1 second based on the prevalence of today's high-speed networks and data from recent studies. These studies showed that RTTs of most connections are less than 1 second, retransmission rates during the handshake are about 2%, and about 2.5% of connections have an RTT exceeding 1 second. The IETF recommendations are designed to protect networks from spurious retransmission and congestion collapse.

However, TCP is used in a variety of environments, and by applications that have special needs. Moreover, different algorithms are used for computing SRTT, RTTVAR and RTO (or minimum RTO), and for congestion control by popular operating systems such as Linux and Windows. This has led to alternative approaches to retransmission that have been determined to be safe, but that do not conform to the IETF standard. Clock granularity and the accuracy of RTT measurements (and their frequency) are also of importance when implementing the TCP retransmission algorithm.

In this study, we examine the impact of using the recommended retransmission timer settings in [1]. Our studies use a bare PC Web server [2] in which the server application is intertwined with lean implementations of the necessary network protocols and there is no operating system or kernel running in the machine. The experiments are conducted in a small test LAN with several routers. We first measure bare PC Web server performance with various settings of the alpha and beta constants used for computing SRTT and RTTVAR in the IETF algorithm [1], and with different RTT sampling strategies (frequencies). These measurements are done in the presence of different levels of background traffic generated by conventional systems to create congestion. We then examine the effect on retransmission of using alternative minimum RTO settings. Finally, we compare the performance of the bare PC Web server using the recommended settings in [1] with the performance of the Apache and IIS Web servers running on Linux and Windows respectively.

We find that no particular combinations of alpha and beta, or sampling strategies, perform consistently better than others under the different levels of background traffic. Also, much lower minimum RTO settings than the recommended 1-second minimum will work when there is only moderate background traffic, but the 1-second RTO minimum is best when there are higher levels of congestion. However, not using congestion control even with the standard timer settings degrades performance depending on the level of background traffic. The remainder of the paper is organized as follows. In Section II, we present related work. In Section III, we describe the test network used for experiments. In Section IV, we give details of the experiments and present the results. In Section V, we give the conclusion.

II. RELATED WORK

Many studies have examined various aspects of TCP's retransmission timer algorithm. In [3], the authors determine how initial RTO settings (used by various operating system versions at the time) impact packet loss rate and suggest changing the initial RTO from 3 seconds to between 500 ms and 1 second. In [4], the RTO algorithm is enhanced by making it window-based rather than round-trip-time-based so that unnecessary retransmissions and unfair resource allocation could be minimized. In [5], the same authors argue that a 1-second minimum RTO is only needed

because of spurious retransmissions that are due to clock granularity and delayed acks. They note that the first problem is not an issue in modern operating systems, and then provide a mechanism to extend the minimum RTO to deal with the second problem. The mechanism is shown by simulation studies to improve TCP performance in the case of a wireless link connected to a high-speed backbone.

The Eifel response algorithm allows a TCP sender to prevent the negative effects of a detected spurious timeout by adapting the retransmission timer [6]. In [7], high-resolution timers are used to solve the problem of throughput collapse in datacenter Ethernets due to TCP timeouts resulting from synchronized request workloads. In [8], a large number of real TCP traces are analyzed, and it is found that RTT values within each connection vary widely. In [9], a new algorithm F-RTO for recovery after retransmission timeout is proposed, and it is shown that unnecessary retransmissions can be avoided without using TCP options. In contrast to these previous studies, we implement the RTO algorithm on a bare PC Web server using the recently recommended TCP timer settings from [1], and study its performance for various values of alpha and beta.

III. TEST NETWORK

The test network used for the experiments consisted of four subnets with clients and servers connected by routers as shown in Fig. 1. The routers used Fedora 12 Kernel Linux 2.6.31.5-127.fc12.i686 and all network interface cards were 1 Gbps except for a 100 Mbps card on the client side of router R1 in order to create congestion). The Ethernet switches were 1 Gbps except for switch S0, which was 100 Mbps. The clients, servers and routers ran on identical Dell OPTEPLEX GX 520 PCs.

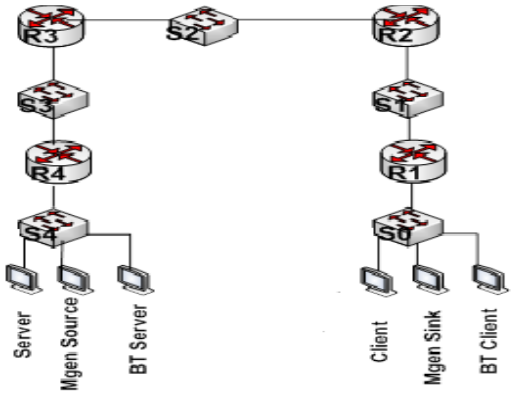

Figure 1. Test network

IV. EXPERIMENTS AND RESULTS

The experiments used background traffic consisting of 10% UDP and 90% TCP at three different levels: 75 Mbps, 100 Mbps and 125 Mbps. The background traffic was generated as follows: BT client used http-load [10] to generate HTTP/1.1 traffic by requesting an uncompressed 320-KB page from BT Server (Apache HTTP Server 2.2.16 running Fedora 12 (Constantine) Kernel Linux 2.6.31.5-127.fc12.i686), while Mgen source used the mgen traffic generator [11] to generate UDP traffic to the Mgen sink. Packets were captured using the Wireshark analyzer [12] at the server side. The traces were used to deduce the values of throughput and delay when the Web browser (Internet Explorer 8.0.6001.18702 with Windows XP Professional 2002 SP3) on the client "Client" requested a 320-KB file from the Web server "Server" (by computing the time between the "GET" request and the last valid ack sent by the client). Results using the Firefox v3.6.7 browser with Linux Fedora 12 Kernel 2.6.31.5-127.fc12.i686 were similar. To determine the effect of changing the values of alpha and beta, the bare PC Web server TCP code was modified accordingly. The measured sample values of RTT were then used to compute the smoothed round trip time (SRTT), the variance in round trip time (RTTVAR), and retransmission timeout (RTO) as follows using the standard algorithm [1]:

RTTVAR = (1− beta)*RTTVAR + beta*|SRTT – RTT|
SRTT = (1- alpha)*SRTT + alpha*RTT
RTO = SRTT + max (G, K*RTTVAR)

Here K=4, and G is the clock granularity, which is 250 microseconds for the bare PC. We examined the effect of changing the values of alpha and beta, and also the effect of computing the value of RTO using 3 different sampling strategies: 1) All: where the RTT measurement is based on the RTTs of all ack packets per window of data; 2) One: where the RTT measurement is based on the RTT of one ack packet per window of data; 3) Some: where the RTT measurement is based on the RTT of some ack packets per window of data (typically, alternate acks were used). The alpha and beta pairs whose combinations were used as the nine weights W1-W9 are shown in Table I. The weight W1 uses the values of alpha and beta from [1]. This weight together with the sampling strategy "One" (one sample per RTT) is commonly used in TCP implementations.

A. Throughput and Delay

For each weight W1-W9 and each of the three sampling strategies "All", "One", and "Some" defined above, the throughput and delay with respect to the 320-KB file were measured for background traffic levels of 75, 100, and 125 Mbps. The results are shown in Figs. 2-7.

TABLE I. ALPHA AND BETA VALUES USED AS WEIGHTS

beta / alpha	1/4	3/8	1/8
1/8	W1	W2	W3
3/16	W4	W5	W6
1/16	W7	W8	W9

In Figs. 2 and 3 respectively, the throughput for 75 Mbps background traffic varies between 8-16 Mbps) and the delay varies between 40-100 ms. In Figs. 4-7, due to congestion caused by the background traffic, the highest throughput for 100 and 125 Mbps background traffic was 450 and 375 Kbps respectively, while the lowest throughput was 50 Kbps. Delays for these higher levels of background traffic vary between a few to over 40 ms. The results in general show that while the often used W1/One combination gives the maximum throughput only for 100 Mbps background traffic,

it has the minimum delay with both 100 and 125 Mbps background traffic. However, no particular combination of weight and sampling strategy is consistently better than the others. Also, sampling more often does not always give higher throughput or lower delays.

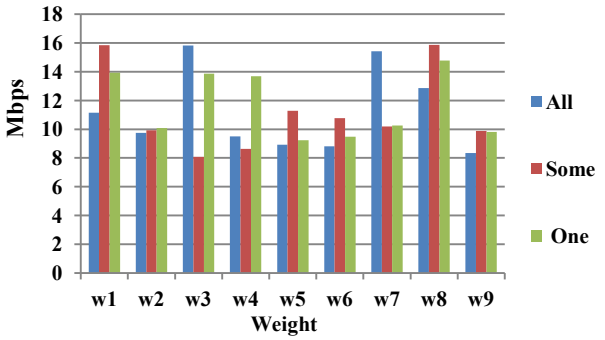

Figure 2. Throughput at 75 Mbps background traffic

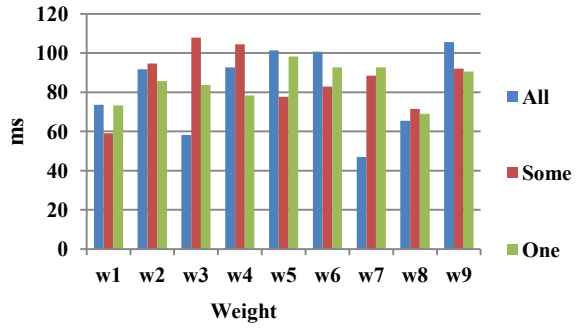

Figure 3. Delay at 75 Mbps background traffic

B. *RTO and RTT*

To determine the accuracy of the computed RTO value for the different weights W1-W9, we compare it with the actual value of RTT in each case. As previously, we use the three sampling strategies "some" (S), "one" (O), and "all" (A), and background traffic levels of 75, 100, and 125 Mbps. The results for these sampling strategies and the three weights W1, W5, and W9 are shown in Table II (results for the other combinations were similar). The values in the table are the percentages of packets for which the RTT is within the computed RTO for a given weight/sampling strategy considering the two cases RTO ≤ 1 second and RTO > 1 second respectively. The table also shows the percentage of packets whose RTT is less than 1 second but exceeds the computed RTO. Such packets would not be retransmitted if the minimum RTO setting is 1 second as required by [1].

For 75 Mbps background traffic, all packets had RTO < 1 second and the RTT did not exceed the RTO for a majority of the packets (83-100%). For this level of background traffic, the percentage of packets that would need to be retransmitted if the minimum RTO is not set to 1 second is highest (17%) with the W1/One combination. For 100 and 125 Mbps background traffic, although between 1-68% of packets have RTO > 1 second, none of these packets needed to be retransmitted. For these levels of background traffic, for packets that have RTO < 1 second, almost all do not need retransmission even if a minimum RTO setting of 1 second is not used (regardless of the weight/sampling strategy combination). For example, with the W1/One combination, 50% of packets have RTO < 1 second and none of the packets need to be retransmitted.

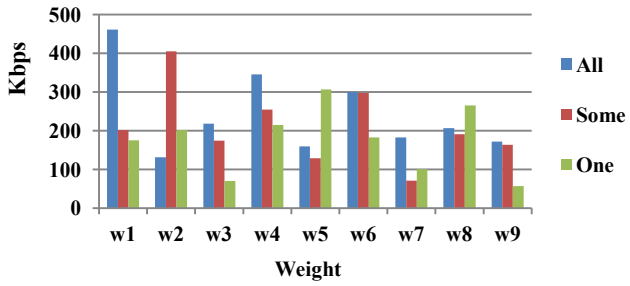

Figure 4. Throughput at 100 Mbps background traffic

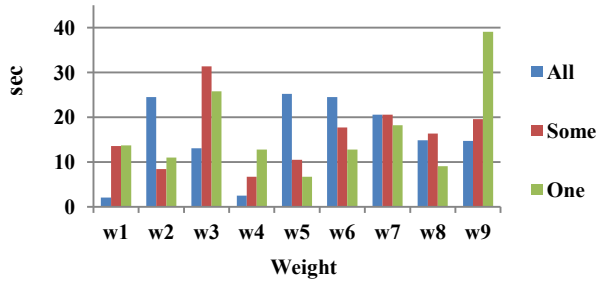

Figure 5. Delay at 100 Mbps background traffic

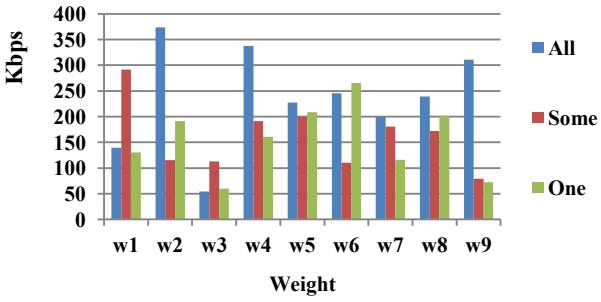

Figure 6. Throughput at 125 Mbps background traffic

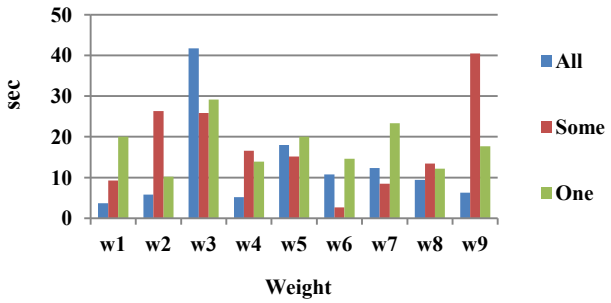

Figure 7. Delay at 125 Mbps background traffic

TABLE II. PERCENTAGE OF PACKETS WITHIN MINIMUM RTO

		RTO ≤ 1 s						RTO > 1 s		
		RTT ≤ RTO			RTT > RTO, RTT < 1 s			RTT < RTO		
Mbps	Wt.	S	O	A	S	O	A	S	O	A
75	W1	96	83	96	4	17	4	0	0	0
	W5	93	93	96	7	7	4	0	0	0
	W9	94	100	96	6	0	4	0	0	0
100	W1	72	50	98	0	0	1	28	50	1
	W5	86	59	88	0	0	0	14	41	12
	W9	75	32	81	0	0	0	25	68	19
125	W1	73	50	62	0	0	0	27	50	38
	W5	80	50	82	0	0	0	20	50	18
	W9	73	50	90	0	0	0	27	46	10

The results of investigating the minimum RTO setting in more detail are shown in Figs. 8 and 9. The graphs show the percentage of packets that would avoid retransmission if the minimum RTO thresholds were set to 0.25, 0.5, 0.75 and 1 second with 100 and 125 Mbps background traffic (the case of 75 Mbps is not shown since the RTO was then always less than 0.25 seconds and the percentage was 100% at all thresholds). For 100 Mbps background traffic (Fig. 8), using the current minimum RTO setting of 1 second, close to 100% of the packets avoid retransmission with the W1/All combination. If a lower minimum RTO setting of 0.5 or 0.75 seconds is used, with the W1/All combination, the percentage is essentially the same. However, the percentage drops with the other combinations, and it can be as low as 50% even using a minimum RTO of 1 second with the W1/One combination. At a higher background traffic level of 125 Mbps (Fig. 9), with the W1/All combination, only 60% of packets avoid retransmission with a minimum RTO of 1 second, but there is only a small drop in this percentage using a minimum RTO of 0.75 or 0.5 seconds. With the W1/One combination, the percentage of packets avoiding retransmission is 50% of packets using a minimum RTO of 1 second, but only 35% using a minimum RTO of 0.75 or 0.5 seconds.

C. *Web Server Performance*

To compare the performance of a bare PC Web server with the standard timer settings in [1] and the Apache and IIS Web servers, the goodput (i.e., the application throughput discounting retransmissions) and delay were measured under background traffic of 75, 100, and 125 Mbps when a pair of servers are run together i.e., (bare PC, Apache), (Apache, IIS), and (IIS, bare PC). Windows server 2008 (IIS 7) and Apache HTTP Server 2.2.16 running Fedora 12 (Constantine) Kernel Linux 2.6.31.5-127.fc12.i686 were used for the performance comparisons. The bare server used only one sample measurement per RTT (i.e., the sampling strategy "One") in the RTO algorithm. Each experiment was run 6 times and the average values are shown in Figs. 10-15 (each bar is the average for a given server-there were no significant differences in the values averaged). In each figure, the first two bars are for the bare and Apache servers respectively when they run together, the third and fourth bars are for the Apache and IIS servers respectively when they run together, and the fifth and sixth bars are for the IIS and bare servers respectively when they run together.

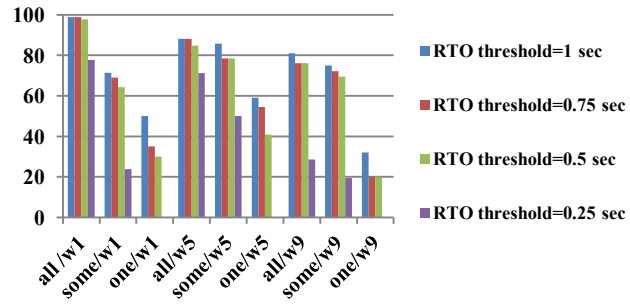

Figure 8. Percentage of packets avoiding retransmission at 100 Mbps Background Traffic

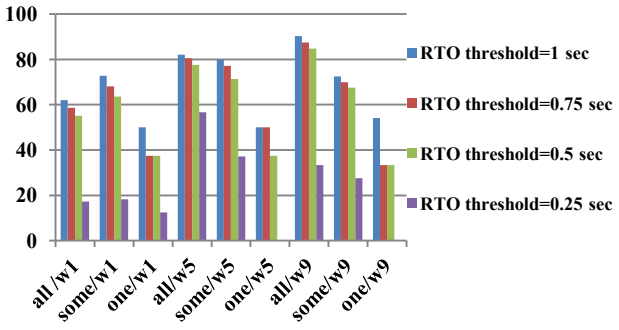

Figure 9. Percentage of packets avoiding retransmission at 125 Mbps background Traffic

For 75 Mbps of background traffic, both Apache and IIS perform better than the bare server. For 100 and 125 Mbps of background traffic, retransmissions occur and the delays of the bare server are higher than for Apache and IIS, and its goodput is lower than for IIS. Compared to Apache, the goodput of the bare server is slightly higher with 125 Mbps of background traffic, but with 100 Mbps of background traffic, the goodput is lower. When run together, IIS performs slightly better than Apache except that its delay for 75 Mbps background traffic is 20 ms more than for Apache. The performance differences in Apache and IIS possibly reflect the differences in their congestion control algorithms and TCP implementations. The lack of congestion control in the bare server (and its "go-back-n"/no SACK rule) results in increased delays at 75 Mbps of background traffic and more retransmissions at higher background traffic levels.

V. CONCLUSION

We studied the impact of changing the values of alpha and beta in the TCP retransmission algorithm using recently recommended initial timer settings. Our studies used a bare PC Web server with no congestion control mechanisms and no TCP optimizations. We also studied the performance of the Web server for various values of the minimum retransmission timeout, and compared the performance of the bare server with the Apache and IIS Web servers. Our results show that no values of alpha and beta are better than others,

and standard timer settings with no SACK and no congestion control degrade performance for some traffic levels.

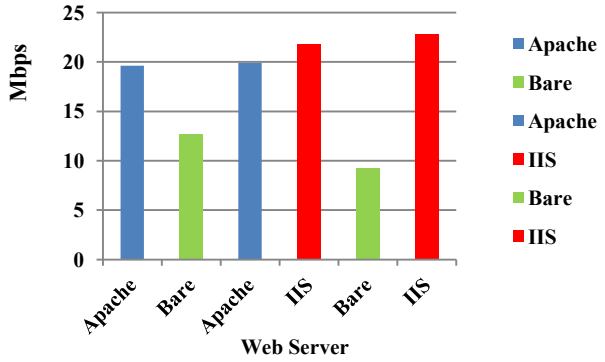

Figure 10. Web server goodput at 75 Mbps background traffic

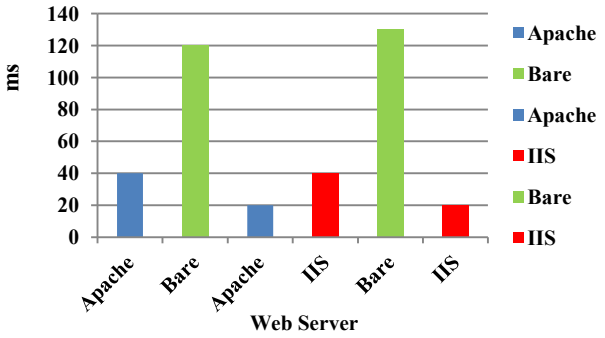

Figure 11. Web server delay at 75 Mbps background traffic

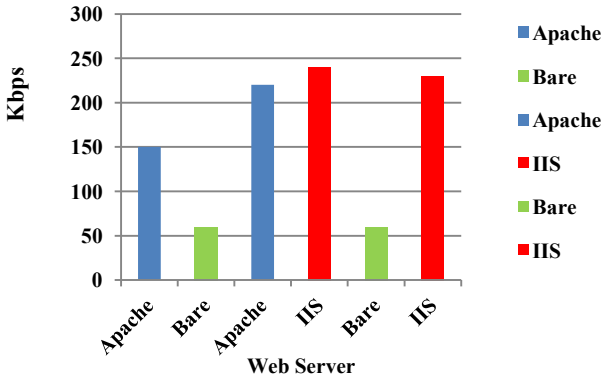

Figure 12. Web server goodput at 100 Mbps background traffic

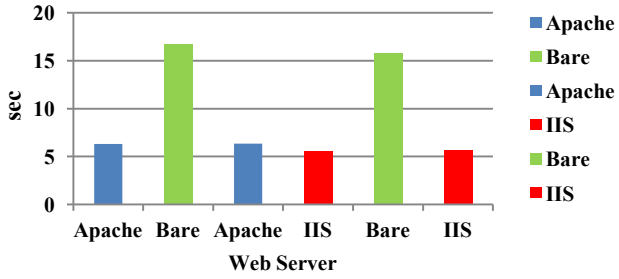

Figure 13. Web server delay at 100 Mbps background traffic

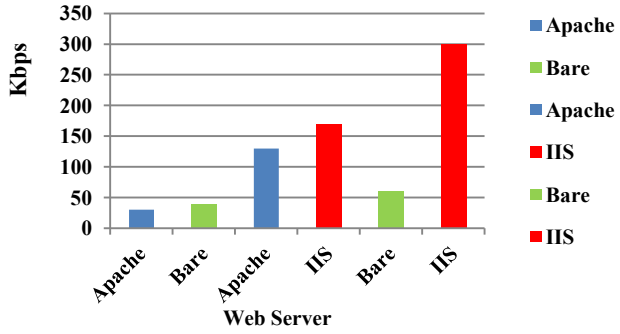

Figure 14. Web server goodput at 125 Mbps background traffic

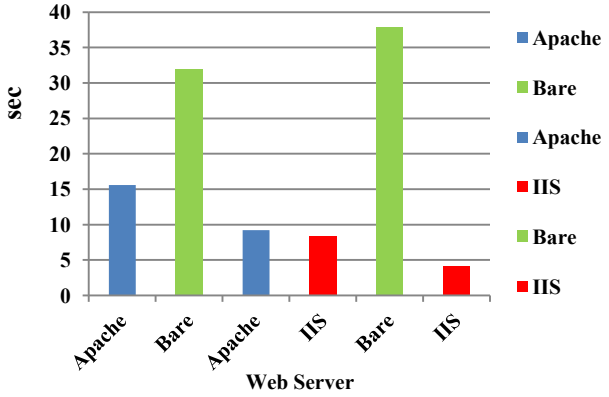

Figure 15. Web server delay at 125 Mbps background traffic

REFERENCES

[1] V. Paxon, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer," RFC 6298, Jun. 2011.

[2] L. He , R. K. Karne ,A. Wijesinha , and A. Emdadi, "Design and Performance of a Bare PC Web Server," IJCTA, vol. 15, no. 2, pp. 110-112, Jun. 2008.

[3] N. Seddigh and M. Devetsikiotis, "Studies of TCP's Retransmission Timeout Mechanism," IEEE ICC, vol. 6, pp 1834-1840, Jun. 2001.

[4] I. Psaras and V. Tsaoussidis, "Why TCP timers (still) don't work well," Elsevier Computer Networks Journal, COMNET 2007.

[5] I. Psaras and V. Tsaoussidis, "The TCP minimum RTO revisited," *in Proc. IFIP Networking*, Atlanta, GA, May 2007.

[6] A. Gurtov and R. Ludwig, "The Eifel response algorithm for TCP," RFC 4015, Feb. 2005.

[7] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datcenter communication," *in Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[8] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP round-trip times," *in Proc. ACM SIGCOMM*, Miami, FA, Oct. 2003.

[9] P. Sarolahti, M. Kojo, and K. Raatikainen, "F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts," *in Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[10] http_load, available at http://www.acme.com, accessed: March 13, 2012.

[11] Mgen, available at http://cs.itd.nrl.navy.mil/work/mgen, accessed: March 12, 2012.

[12] Wireshark, available at http://www.wireshark.org, accessed: March 13, 2012.