WEB SERVER PERFORMANCE WITH CUBIC AND COMPOUND TCP

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

ABSTRACT
TCP congestion control mechanisms are used to prevent traffic volume from exceeding network resource limits. Consequently, they indirectly impact the performance of Web servers in a congested network. Systems running under Windows and Linux, including Web servers, use respectively the Compound and Cubic TCP congestion control variants that are designed for high-speed long-delay networks. We evaluate the performance under congestion of the popular Windows-based IIS and Linux-based Apache Web servers when serving a single request from a browser. Specifically, we conduct experiments in a real test network with several routers to compare delay, throughput and shared bandwidth percentage when the Web server is subjected to various workloads under different levels of background network traffic. We find that IIS with Compound TCP has performance advantages over Apache with Cubic TCP when the two servers compete for bandwidth, but Apache has smaller delays than IIS for large and medium-sized files.

KEY WORDS
TCP congestion control, Cubic TCP, Compound TCP Web server performance, high bandwidth-delay product networks.

1 Introduction

Most applications today use the transport protocol TCP for communication on LANs and on the Internet. TCP provides reliable data transmission with flow and congestion control. Congestion control algorithms are used to estimate the level of congestion in the network by using loss-based and delay-based techniques. Congestion management is the most complex aspect of TCP; it has been extensively researched and continues to be studied today. Recent research has focused on managing congestion in high-speed long-delay i.e., high bandwidth-delay product (BDP) networks.

The most used TCP congestion control variants are Compound TCP on Windows and Cubic TCP on Linux. Many simulation and real-world studies have confirmed that these two algorithms exhibit good performance and are especially suited for the high BDP networks they target. However, no studies have been done that examine the impact of these two algorithms on the performance of Web servers with browser requests under different workloads, and in the presence of background traffic, by conducting experiments using a real network.

We study the performance of the popular IIS and Apache Web servers that respectively implement Cubic Compound and Cubic TCP congestion control in a test network with several routers. We send HTTP requests to the Web servers using a Web browser (Internet Explorer or Mozilla Firefox) under varying levels of server load and background traffic to create network congestion. The server workload and background network traffic consists of a mix of HTTP/TCP and UDP traffic generated using the freely-available tools http_load and Mgen respectively. Performance is measured by determining delay, throughput, and shared bandwidth percentage using a Wireshark packet analyzer and http_load. We only compare performance of the default (preconfigured) installations of the two congestion control algorithms (i.e., without making any adjustments to preset parameters). The main finding is that IIS with Compound TCP performs better than Apache with Cubic TCP when the two servers compete for bandwidth, but Apache has lower delays than IIS for large and medium-sized files.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we describe the experimental setup. In Section 4, we present and discuss the results, and in Section 5, we give the conclusion.

2 Related Work

Numerous studies have been conducted on the congestion control aspects of TCP. For example in [1], the TCP congestion window update algorithm is modified to address the issue of RTT-fairness, and it is shown by a combination of analysis, simulation and testbed studies that protocol stability and efficiency are maintained. In [2], simulation studies are used to show that an existing TCP variant is suited for use on the local segment of a spliced connection to address the needs of wireless multimedia flows with home entertainment centers. While many variants of its basic “Reno” congestion control algorithm [3] have been proposed, relatively few have seen large-scale deployment in real-world networks.

Two notable exceptions are Compound TCP [4, 5] used by the Windows operating system and Cubic TCP [6] used by the Linux operating system that attempt to address the needs of today's high BDP networks. Compound TCP (also known as C-TCP) modifies the basic congestion control algorithm by adding a scalable term to the congestion control window based on the standard "Vegas" estimate of network buffering. In contrast, Cubic TCP models the congestion window as a cubic function of the time since the last detected loss that also depends on the congestion window size prior to that loss. While both variants are known to perform well under congestion in high BDP networks and have good fairness properties, Compound TCP may not treat all flows equally and Cubic TCP may have sub-optimal network utilization and increase packet loss [7].

Our study examines performance of the IIS and Apache Web servers with their respective Compound and Cubic TCP variants under different workloads for a single browser request in a network when there are competing flows and other background traffic. Most TCP studies of its congestion control variants use simulation. For example, [8] and [9] study the performance of the two variants using a simulation tool. However, [10] suggests that simulation studies may not reflect the complexity of TCP protocol behavior with traffic in a real network. Also, in [11] it is observed that real implementations of congestion control algorithms often make changes to parts of the TCP stack that are not directly related to the congestion algorithm to improve overall performance. We consider "out-of-the-box" (i.e., default) implementations of TCP in the two most popular Web servers and their operating systems running on real hardware. Finally, we use background traffic in accordance with the guidelines in [12] to study performance of the two TCP variants.

3 Experimental Setup

3.1 Test Network and Traffic Generation

For our experiments, we used a network of clients connected to a network of servers via four Linux routers R1-R4 and five Ethernet switches S0-S5 as shown in Fig. 1. Details of the hardware and software used for the experiments are given in Table 1. All NICs and switches are 1 Gbps except for R1's NIC on its internal (i.e., client network side) interface, and switch S0, that are 100 Mbps in order to cause congestion. In the absence of other traffic, the RTT between routers R1 and R4 is approximately 0.3 ms.

The Web browser on the client (referred to as "Client" in the figure) sent a single request for a 504, 320, or 160 KB file (to represent large, medium and small files respectively) to the Web server (referred to as "Server" in the figure) in the presence of background traffic on the network and stress traffic to the Web server "Server" as described below. The experiments were done using Mozilla Firefox and repeated using Internet Explorer as the browser on "Client". Since the results with both browsers were similar, only the results using Firefox are reported.

The congestion scenarios for the experiments (described below) were created using the test network in Fig. 1 (or minor variations of it) by generating traffic as follows. For background traffic, a mix of UDP and HTTP/TCP traffic was generated using the Mgen [13] and http_load [14] generators respectively: the "Mgen Source" sent UDP packets to the "Mgen Sink" at varying rates, and a background traffic client (referred to as "BT Client") sent multiple HTTP requests at varying rates via http_load for a 320 KB file to a background traffic Web server (referred to as "BT Server"). In addition, stress traffic in the form of competing requests to the Web server "Server" at varying rates for a 320 KB file was generated by a client (referred to as "Stress Client") using http_load. The combination of background and stress traffic creates a traffic mix for examining the behavior of the two congestion control variants.

3.2 Congestion Scenarios

We studied performance of the Apache/Linux and IIS/Windows Web servers with their respective TCP congestion control approaches (Cubic TCP and Compound TCP) by sending browser requests to the servers under three congestion scenarios C1, C2, and C3. The scenarios create different mixes of background and stress traffic on the network totaling 100 Mbps in each case as described below.

C1: Background traffic is 10 Mbps of UDP traffic from Mgen Source to Mgen Sink, and 40 Mbps of HTTP/TCP traffic generated by requests from BT Client to BT Server i.e., the background traffic is 20% UDP and 80% TCP. The Web server "Server" additionally has a workload of 50 Mbps of HTTP/TCP stress traffic due to requests from Stress Client.

C2: Background traffic is 15 Mbps of UDP traffic from Mgen Source to Mgen Sink, and 60 Mbps of HTTP/TCP traffic generated by requests from BT Client to BT Server i.e., the background traffic is 20% UDP and 80% TCP. The Web server "Server" additionally has a workload of 25 Mbps of HTTP/TCP stress traffic due to requests from Stress Client.

C3: Background traffic is 10 Mbps of UDP traffic from Mgen Source to Mgen Sink, and 30 Mbps of HTTP/TCP traffic generated by requests from BT Client to BT Server i.e., the background traffic is 25% UDP and 75% TCP. The Web server "Server" additionally has a workload of 60 Mbps of HTTP/TCP stress traffic due to requests from Stress Client.

The differences between the scenarios can be characterized as follows: for UDP background traffic, scenarios C3 and C1 have the same levels, while C2 has 5 Mbps more. For TCP background traffic, C2 has 20 Mbps more than C1, which has 10 Mbps more than C3. For TCP stress traffic, C3 has 10 Mbps more than C1, which has 25 Mbps more than C2.

Comparing TCP stress traffic and the total (TCP and UDP) background traffic, C1 has the same levels, while C3 has 60% stress and 40% background, and C2 has 75% background and 25% stress.

For all three scenarios, we start Mgen, run http_load on Stress Client and BT Client, and send a single request from the Web browser on “Client” to the Web server on “Server”. The Wireshark packet analyzer is used on the server side to measure delay and throughput. Information from http_load running on the Stress and BT Clients is used to compute throughput on the client side. Measuring the throughput on both the client and server side enables the impact of congestion control with and without network delays to be studied.

4 Experimental Results

In this section, we present the results from experiments to test the performance of the Apache and IIS Web servers, which run Cubic TCP and Compound TCP respectively. We conduct experiments with competing flows, where background traffic and stress traffic use the same or different server types (IIS or Apache), or where two browsers send requests to different server types. Traffic is generated based on the three congestion scenarios C1, C2 and C3 described above, and each experiment consists of requesting a large, medium or small file (of size 504, 320 and 160 KB respectively). All results reported are the average of three repeated stable runs of each experiment; the results averaged were not significantly different.

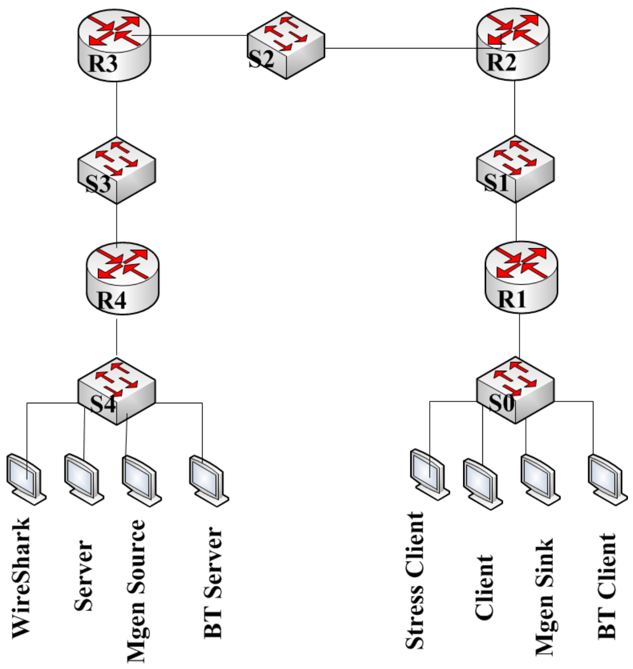

Figure 1. Test Network

Table 1
Hardware and Software Specification

Device/Function	**Details**
Switch S0	Netgear/FSM726S
Switch S1	Cisco/SG100d-08
Switch S2	Linksys/EG005W
Switch S3	Cisco/SG100d-08
Switch S4	Netgear/GS108T
Packet Analyzer	Dell OPTEPLEX GX520;XP/ Wireshark Version 1.2.7 (SVN Rev 32341)
Servers	Dell OPTEPLEX GX520; Windows server 2008 (IIS 7); Fedora 12 (Constantine) Kernel Linux 2.6.31.5-127.fc12.i686 Apache HTTP Server 2.2.16
Mgen source	Dell OPTEPLEX GX260; Windows XP Professional 2002 SP3
Mgen sink	Dell OPTEPLEX GX260; CentOS Version 2.16.0
Clients	Dell OPTEPLEX GX520; Windows XP Professional 2002 SP3; Linux Fedora 12 Kernel 2.6.31.5-127.fc12.i686; browser: Firefox v3.6.7, Internet Explorer 8.0.6001.18702
Routers	Dell OPTEPLEX GX520; Fedora 12 Kernel Linux 2.6.31.5-127.fc12.i686

4.1 Same Server Type

In the first set of experiments, Server and BT Server are the same type i.e., both are Apache or both are IIS. Traffic is generated according to the congestion scenarios C1, C2 and C3. Figs. 2-4 show the delay for a single browser request, and Figs. 5-7 show the server throughput using large, medium and small files respectively.

The server delay includes the processing time and the delays introduced by the congestion control algorithm. The former depends on the total workload (the file size, the total number of bytes processed, and the OS/server implementation), and the latter depends on the congestion level and the characteristics of the algorithm. Observed delays may be due to a combination of factors. For example, a more aggressive backoff policy to address increased congestion is compensated for by smaller delays due to the reduction in traffic and packet loss.

Let t_server(C) denote the total server delay in responding to the browser request for a scenario C. Fig. 2 shows that for large files, t_IIS(C3) > t_IIS(C1) ≈ t_IIS(C2) ≈ t_Apache(C2) > t_Apache(C3) > t_Apache(C1). For IIS, when serving a large file, an increase in the workload due to stress traffic increases the delay as expected in scenario C3 over scenario C1.

However, comparing scenario C1 with C2, the stress traffic is reduced by 50% in scenario C2, and the background traffic is increased by 50%, but there is no difference in the delay. This appears to suggest that Compound TCP has reduced the transmission rate due to congestion resulting in a delay that is the same as the delay for C1. For Apache, the relative performance for scenarios C1 and C3 is the same as for IIS, but the congestion due to the increased background traffic in C2 has resulted in larger delay than in C1. This means that Cubic TCP reduces its transmission rate more aggressively than IIS when congestion increases, resulting in a greater delay to serve a large file.

For medium files, Fig. 3 shows that t_IIS(C3) ≈ t_IIS(C1) > t_IIS(C2) > t_Apache(C3) ≈ t_Apache(C1) > t_Apache(C2). In this case, the additional stress traffic in C3 compared to C1 did not increase the delay because the requested file size is smaller and because both congestion control variants have reduced the transmission rate in C1 due to the increased background traffic. In the case of scenario C2, the reduced file size and the highest level of background traffic result in the least delay because each congestion variant has reduced its transmission rate sufficiently.

For small files, Fig. 4 shows that t_IIS(C1) < t_Apache(C1) ≈ t_IIS(C3) < t_IIS(C2) < t_Apache(C2) < t_Apache(C3). Here, the delays for scenarios C1 are less than for C3 as expected since it has less stress traffic, but the delay for C2 is larger than for C1 with both servers. Since the file size is small, the increased delays for C2 are due to reducing the transmission rate because of the congestion caused by the additional background traffic. In this case, Cubic TCP has a larger delay than Compound TCP because it is reducing its transmission rate more aggressively.

In general, the above results indicate that the Apache server is more efficient than IIS for large and medium files, but less efficient for small files. They also indicate that the Apache server has reduced performance compared to IIS when congestion increases as in scenario C2. This is because Cubic TCP adopts a more conservative approach than Compound TCP when losses are detected. More studies with a variety of congestion scenarios are needed to validate this claim.

In Figs. 5-7, NRT refers to the server throughput when retransmissions are ignored. It is seen that scenario C2 has the lowest and scenario C3 has the highest server throughput regardless of file size as would be expected. However, throughput for IIS is slightly higher than the corresponding throughput for Apache in all cases. This suggests that Compound TCP (IIS/Windows) may be transmitting more aggressively under congestion than Cubic TCP (Apache/Linux) since the throughput is higher even for large and moderate file sizes (when delays for IIS were higher). The throughput in each scenario for all file sizes is about the same because the stress traffic level in each scenario is the same, and the difference between the small, medium and large file sizes used for the single request made by the browser does not affect the throughput.

Fig. 8 shows the throughput measured at the Stress and BT clients with a request for a 320 KB file. As with server throughput, client throughput is also almost identical for the three file sizes, so only the results for one file size are shown here. While client throughput is less than server throughput due to network delay, relative relationships between throughput values are not preserved. For example in scenario C1, IIS and Apache throughput measured at the client is about the same, whereas IIS NRT throughput measured at the server side is slightly higher than the Apache NRT throughput (Fig. 6). This difference could be insignificant or due to a slightly lower packet loss with IIS (and hence fewer retransmissions by the IIS server). More detailed studies are needed to understand the relation between Compound and Cubic TCP, and the differences in observed throughput at the client and the server.

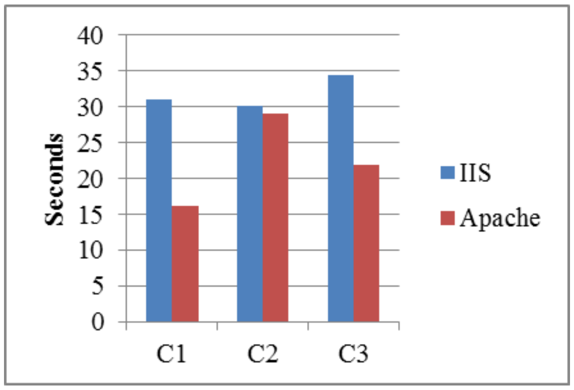

Figure 2. Server delay: 504 KB

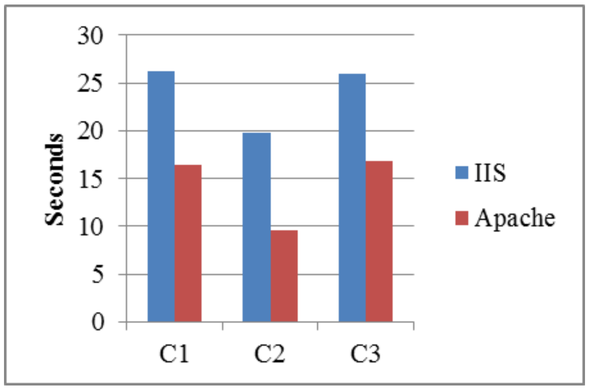

Figure 3. Server delay: 320 KB

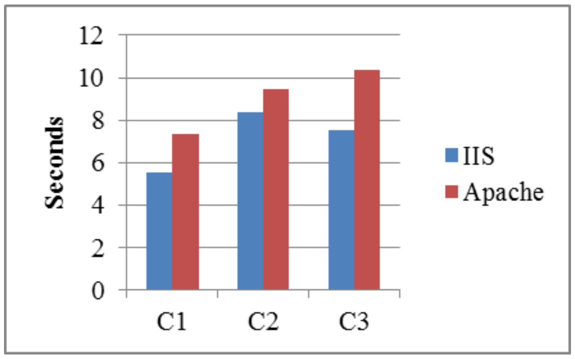

Figure 4. Server delay: 160 KB

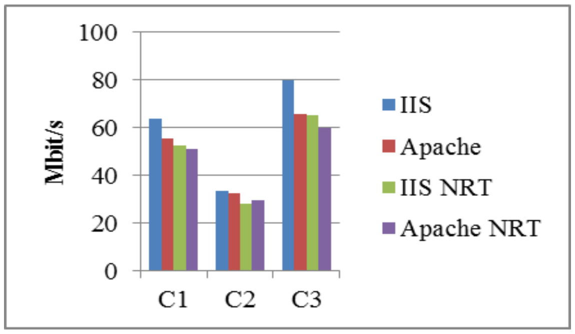

Figure 5. Server throughput: 504 KB

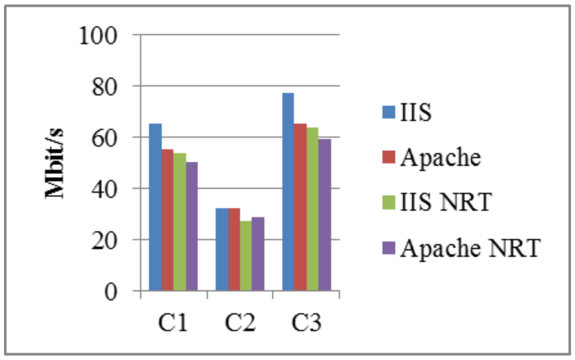

Figure 6. Server throughput: 504 KB

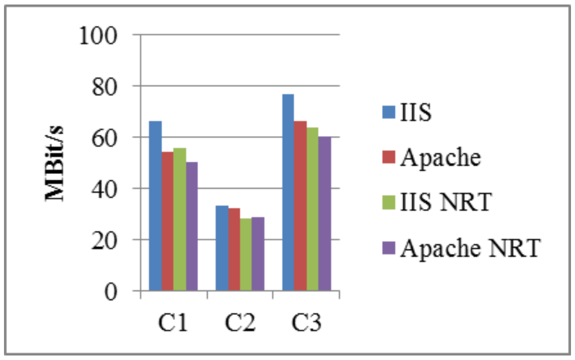

Figure 7. Server throughput: 160 KB

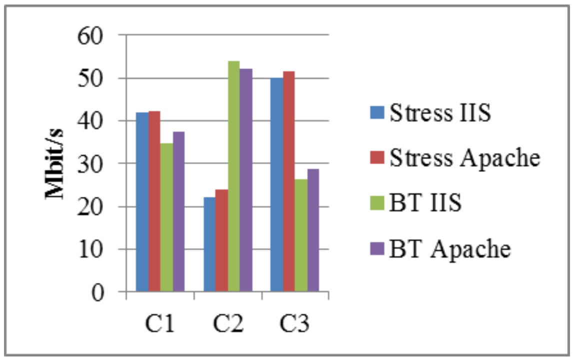

Figure 8. Client throughput: 320 KB

Table 2 shows the bandwidth percentage shared between the Stress Traffic (ST) and Background Traffic (BT) servers. For IIS, each server gets roughly an equal share of the bandwidth, whereas for Apache, one of the two servers is always getting a larger share. For instance in scenario C1, the maximum possible rates are 50 Mbps of TCP stress traffic and 40 Mbps of TCP background traffic. For IIS, 84.1% of the maximum stress traffic rate and 87% of the maximum background traffic rate are achieved, which is a 2.9% difference. For Apache, 84.4% of the maximum stress traffic rate and 93.9% of the maximum background traffic are achieved, which is a 9.5% difference. This result shows that Compound TCP enables better sharing of the available bandwidth with another like flow (i.e., Compound TCP) than Cubic TCP.

Table 2
Shared bandwidth percentage (same server type)

Scenario	ST IIS	BT IIS	ST Apache	BT Apache
C1	84.1	87.0	84.4	93.9
C2	88.5	89.9	95.7	87.0
C3	83.6	88.3	85.7	95.9

4.2 Different Server Types

In the second set of experiments, a file of the same size (504 KB) is requested by the browser under all three congestion scenarios. For these experiments, Server and BT Server are of different types (i.e., Server is IIS and BT Server is Apache, or Server is Apache and BT Server is IIS).

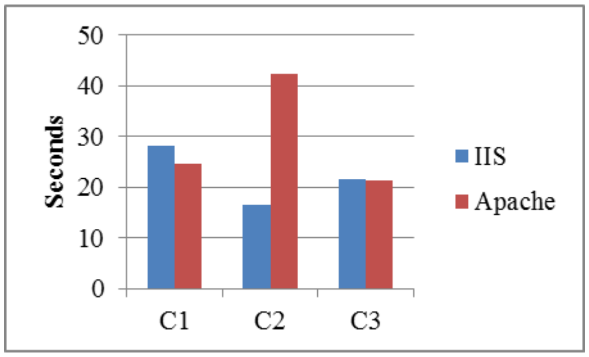

Figure 9. Delay (different servers): 504 KB

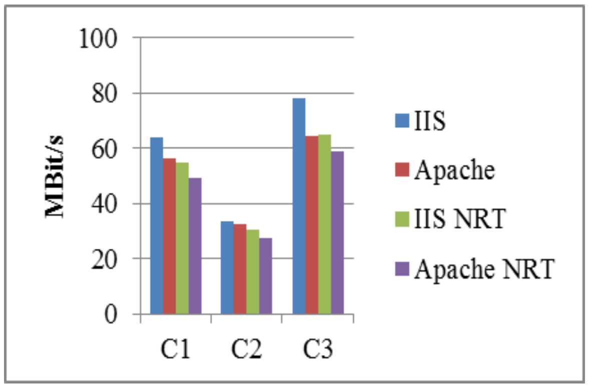

Figure 10. Server throughput (different servers): 504 KB

In Fig. 9, it is seen that for scenario C2 (in which background traffic is at its highest level), the delay for Apache when serving a single browser request is 3 times that for IIS when the stress and background servers are of different types. In contrast, for scenarios C1 and C3, the IIS delay is only slightly larger than the Apache delay. Comparing with the case when the servers were of the same type (Fig. 2), the IIS delays for C1 and C3 are larger than for Apache, but the difference between IIS and Apache for C2 is insignificant. It therefore appears that the reduction in the transmission rate under congestion is larger for Cubic TCP (Apache) than it is for Compound TCP (IIS).

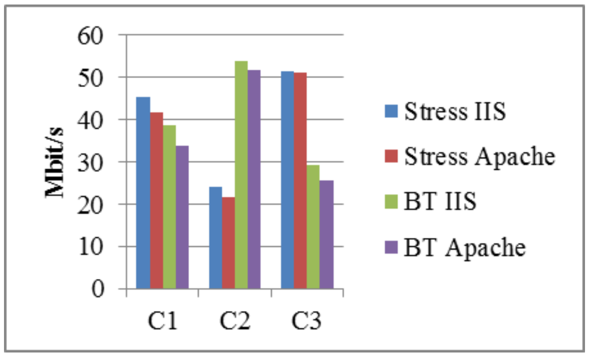

Figure 11. Client throughput (different servers): 504 KB

In Figs. 10 and 11, server and client NRT throughput with different server types is similar to the result with servers of the same type (Figs. 5 and 8). The IIS server now has slightly larger throughput than Apache for all scenarios.

Table 3
Shared bandwidth percentage (different servers)

Scenario	ST IIS	BT Apache	ST Apache	BT IIS
C1	90.7	84.3	83.5	96.8
C2	97.1	86.4	87.4	90.0
C3	85.7	85.8	85.2	98.0

Table 3 shows the bandwidth percentage shared between the Stress Traffic (ST) and Background Traffic (BT) servers when the servers are of different types. In scenario C1, IIS achieves 90.7% of the maximum stress traffic rate and Apache achieves 84.3% of the maximum background traffic rate, which is a 6.4% difference. When the servers are switched, IIS achieves 96.8% of the maximum background traffic rate and Apache achieves 83.5% of the maximum stress traffic rate, which is a 13.3% difference. Similarly in scenario C2, IIS achieves 97.1% of the maximum stress traffic rate and Apache achieves 86.4% of the maximum background traffic rate; and in scenario C3, IIS achieves 98.0% of the maximum background traffic rate and Apache achieves 85.2% of the maximum stress traffic rate. This shows that IIS can achieve most of its allowed maximum traffic rate when sharing the network with an unlike flow (i.e., Cubic TCP), which can only achieve about 84-86% of its allowed maximum traffic rate. Even when the shared percentages between Compound and Cubic TCP are more equitable, Cubic TCP did not achieve more than 87.4% of its allowed rate. Thus, Cubic TCP defers to Compound TCP in a congested network, which limits the performance of the Apache Web server.

4.3 Competing Servers

In the third set of experiments, we modify the test network to include two Web servers Server1 (S1) and Server2 (S2), two clients Client1 and Client2, and two Stress clients C1 Stress and C2 Stress. In addition to 10 Mbps of UDP traffic from Mgen, 30 Mbps of background HTTP/TCP traffic are generated by requests from BT Client (Cbt) to BT Server (BTS), and each server S1 and S2 has a workload of 30 Mbps of HTTP/TCP stress traffic due to requests from the clients C1 Stress and C2 Stress respectively.

As in the earlier experiments, the combined stress traffic and background traffic total 100 Mbps. The browser on client Client1 then sends a single request to the Web server S1, and simultaneously the browser on client Client2 sends a single request to the Web server S2. Each request is for a 504 KB file.

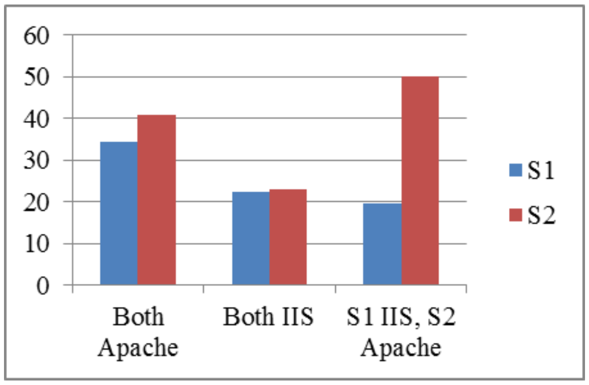

Figure 12. Server delay (competing servers): 504 KB

Fig. 12 compares the server delay when 1) both S1 and S2 run Apache, and BTS is IIS; or 2) both S1 and S2 run IIS, and BTS is Apache; or 3) S1 runs IIS and S2 runs Apache, and BTS is Apache. In 1) and 2), the delays are comparable. Although the two Apache servers do not share the bandwidth equally and experience delays that differ by about 5 seconds (i.e., 35 and 40 second delays), the IIS servers have equal delays of about 20 seconds. In 3), the performance of Apache degrades to an unacceptable level (a delay of 50 seconds), whereas IIS maintains its 20 second delay. It is clear that IIS (Compound TCP) performs better than Apache (Cubic TCP) in this case.

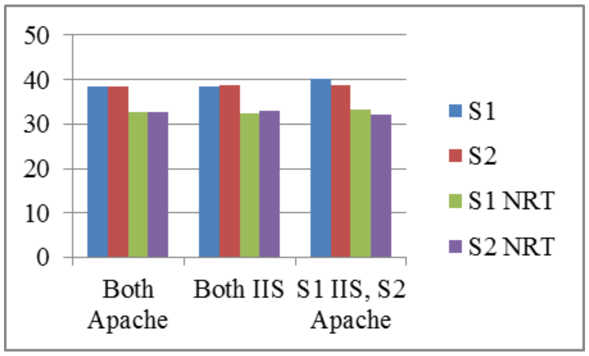

Figure 13. Server throughput (competing servers):504 KB

Fig. 13 shows the server throughput for this experiment. It can be seen that the throughput values for both servers are comparable in each of the three cases. Fig. 14 compares the throughput values measured at the client side. Again, the throughput values are comparable, although the throughput at the background client is marginally less than the throughput at each of the two stress clients.

Table 4 shows the bandwidth percentage shared between the three flows measured at the client side for the case of competing servers. It can be seen that each flow gets an approximately equal percentage of the maximum allowed bandwidth.

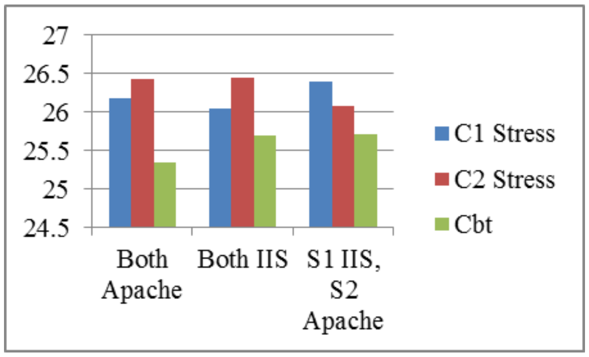

Figure 14. Client throughput (competing servers): 504 KB

Table 4
Client throughput (competing servers): 504 KB

Scenario	C1 Stress	C2 Stress	Cbt
S1, S2 Apache, BTS IIS	87.3	88.1	84.5
S1, S2 IIS, BTS Apache	86.8	88.1	85.6
S1 IIS, S2 Apache, BTS Apache	88	86.9	85.7

5 Conclusion

We conducted experiments in a test network to study the impact of two widely implemented TCP congestion control variants Compound TCP and Cubic TCP on Web server performance using the popular IIS and Apache Web servers. Congestion was created by generating a mix of background traffic and different workloads, and performance was measured using server delay for a single browser request, throughput measured at the server and client ends, and shared bandwidth percentage. The Apache server with Cubic TCP has less delay than the IIS server with Compound TCP for large and medium files, but not for small files. Throughput for IIS is slightly higher than throughput for Apache regardless of the congestion scenarios that were used in the experiments. For experiments with like servers, IIS shares bandwidth more equitably than Apache and for experiments with unlike servers, the latter adopts a conservative approach that limits its performance. When IIS competes with Apache for bandwidth under congestion, although both get a roughly equal share of the allowed bandwidth, the delay for Apache rises to an unacceptably high level.

References

[1] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, and M. Roccetti, “TCP Libra: Derivation, Analysis and Comparison with Other RTT-Fair TCPs”, Computer Networks, Elsevier, vol. 54, no. 14, October 2010, pp. 2327-2344.

[2] G. Marfia and M. Roccetti, “TCP At Last: Reconsidering TCP's Role for Wireless Entertainment Centers at Home”, IEEE Transactions on Consumer Electronics, IEEE Consumer Electronics Society, vol. 56, no. 4, November 2010, pp. 2233-2240.

[3] M. Allman, V. Paxon, and W. Stevens, “TCP Congestion Control”, RFC 2581, April 1999.

[4] K. Tan, J. Song, Q. Zhang, and M. Sridharan, “A Compound TCP Approach for High-Speed and long Distance Networks”, INFOCOM 2006, Barcelona, Spain, Apr. 2006.

[5] K. Tan, J. Song, M. Sridharan, and C. Ho, “CTCP: Improving TCP-Friendliness Over Low-Buffered Network Links”, Microsoft Technical Report.

[6] I. Rhee and L. Xu, “Cubic: A New TCP-Friendly High-Speed TCP Variant,” ACM SIGOPS Operating System Review, Volume 42, Issue 5, July 2008, pp. 64-74, 2008.

[7] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP”, IEEE Communications Surveys and Tutorials, vol. 12, no. 3, pp. 304-342, 2010.

[8] H. Jamal and K. Sultan, “Performance Analysis of TCP Congestion Control Algorithms”, International Journal of Computers and Communications, Issue 1, Volume 2, 2008.

[9] J. Chicco, D. Collange, and A. Blanc, “Simulation Study of New TCP Variants”, IEEE symposium on Computers and Communications, June 2010.

[10] M. Bateman and S. Bhatti, “TCP testing: How well does ns2 match reality?”, IEEE AINA, April 2010.

[11] Y. Li, D. Leith, and R. Shorten, “Experimental Evaluation of TCP Protocols for High-Speed Networks”, Technical Report, Hamilton Institute, 2005.

[12] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, **“**A Step toward Realistic Performance Evaluation of High-Speed TCP Variants”, PFLDnet, Nara, Japan, 2006.

[13] Mgen, available http://www.cs.itd.nrl.navy.mil/work/mgen accessed: Feb 13, 2012.

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