# Data and Network Optimization Effect on Web Performance

Steven Rosenberg, Surbhi Dangi, Isuru Warnakulasooriya

February 10, 2012

CMU-SV-12-001

Carnegie Mellon University
Silicon Valley

Silicon Valley Campus
Carnegie Mellon University
Mountain View, CA 94035

# Data and Network Optimization Effect on Web Performance

**Steven Rosenberg, Surbhi Dangi & Isuru Warnakulasooriya**

**{steven.rosenberg, surbhi.dangi}@sv.cmu.edu, isuru.warn@alumni.cmu.edu**

Carnegie Mellon University Silicon Valley | NASA Research Park | Bldg 23 (MS 23-11) | Moffett Field, CA 94035

## 0. Summary

In this study, we measure the effects of two software approaches to improving data and network performance: 1. Content optimization and compression; and 2. Optimizing network protocols. We achieve content optimization and compression by means of BoostEdge and employ the SPDY network protocol to lower the round trip time for HTTP transactions. BoostEdge [1] by ActivNetworks is a leading Application Delivery Controller (ADC) product, and SPDY [2] by Google is a network optimization protocol built into the Chrome browser. Since the data and transport layers are separate, we conclude our investigation by studying the combined effect of these two techniques on web performance. Using document mean load time as the measure, we found that for our testing profile, with and without packet loss, both BoostEdge and SPDY provide a significant improvement in speed over HTTP. When tested in various combinations, we find that the effects are more or less additive and that the maximum improvement is gained by using BoostEdge and SPDY together. Interestingly, the two approaches are also complimentary; i.e., in situations where data predominates (i.e. "heavy" data, and fewer network requests), BoostEdge provides a larger boost via its data optimization capabilities and in cases where the data is relatively small, or "light", but there are many network transactions required, SPDY provides an increased proportion of the overall boost. The general effect is that relative level of improvement remains consistent over various types of websites.

## I. Introduction

It is expected that demand for affordable data bandwidth will outpace supply for the foreseeable future. In developed countries, we are entering an era of all data available any time, any place, including pervasive HD video streaming, augmented reality apps, mobile telepresence, connected cars, etc. In sparsely populated rural areas and less developed countries, there is a need to expand the available bandwidth for the increasing numbers of users who rely on their mobile devices as their primary service, and to build out sufficient fixed line capacity, all within tight financial constraints.

To address these issues, many technologies are being developed or optimized to remedy the problem of bandwidth congestion and degraded performance. In this situation, software techniques for increasing the capacity of current infrastructure and web servers are very appealing to network providers and website owners, as they are easy and quick to install, and do not require expensive capital and maintenance costs for equipment. Two of the most promising software approaches are (a) content optimization and compression, and (b) optimizing network protocols. Since network protocol optimization and data optimization operate at different levels, there is an opportunity for improvement beyond what can be achieved by either of the approaches individually. In this paper, we report on the performance benefits observed by following a unified approach, using both network protocol and data optimization techniques, and the inherent benefits in network performance by combining these approaches in to a single solution.

Specifically, to demonstrate our hypothesis, we have chosen SPDY [2] by Google, a network optimization protocol built into the Chrome browser and BoostEdge [1] by ActivNetworks, a leading Application Delivery Controller (ADC) product for data optimization and compression. SPDY is an application layer protocol that improves on HTTP response times by crunching the latency effects of a network through optimizing the “handshakes” and reducing the number of required connections for a session (very efficient on high Round Trip Time sessions). SPDY is widely and transparently available to users via the Chrome browser, although at the moment it is not widely supported on the server side.

BoostEdge, on the other hand, operates on the HTTP layer to compress data content, resulting in bandwidth savings and lower page-load times (very efficient on congested networks). It also facilitates data "optimization" on the client side in multiple ways such as:

1. Forcing a client to optimally cache data.
2. Introducing a time delay to send data to the client when the previously sent data requires time to load.
3. Enabling lossy image compression that is geared to the client screen size.

BoostEdge also facilitates "optimization" of transactions on the server side, as loads increase towards saturation, by various techniques that take advantage of its knowledge of the nature of the connection and client, such as speeding up very slow connections more than the already fast connections. While there are other ADCs that can also compress data, BoostEdge's approach is extremely flexible and transparent to the client as it requires no special software or hardware on the client.

## II. Testbed Description

To measure network performance, we built a testbed that would allow us to benchmark BoostEdge and SPDY by measuring HTTP and HTTPS performance individually and in relevant combinations.

The testbed (Figure 1) is a fairly simple one and consists of a 1 Gbps switch bridging two loaders and the BoostEdge device. The BoostEdge device then connects to two servers through another 1 Gbps switch that play host to downloaded websites as a simulation of web traffic for this setup. Web requests are generated using an external machine, connecting from the Carnegie Mellon University Silicon Valley (CMU SV) Ethernet and wireless network to the testbed through the first switch. The external machine is used to introduce web traffic and enable bandwidth throttling.

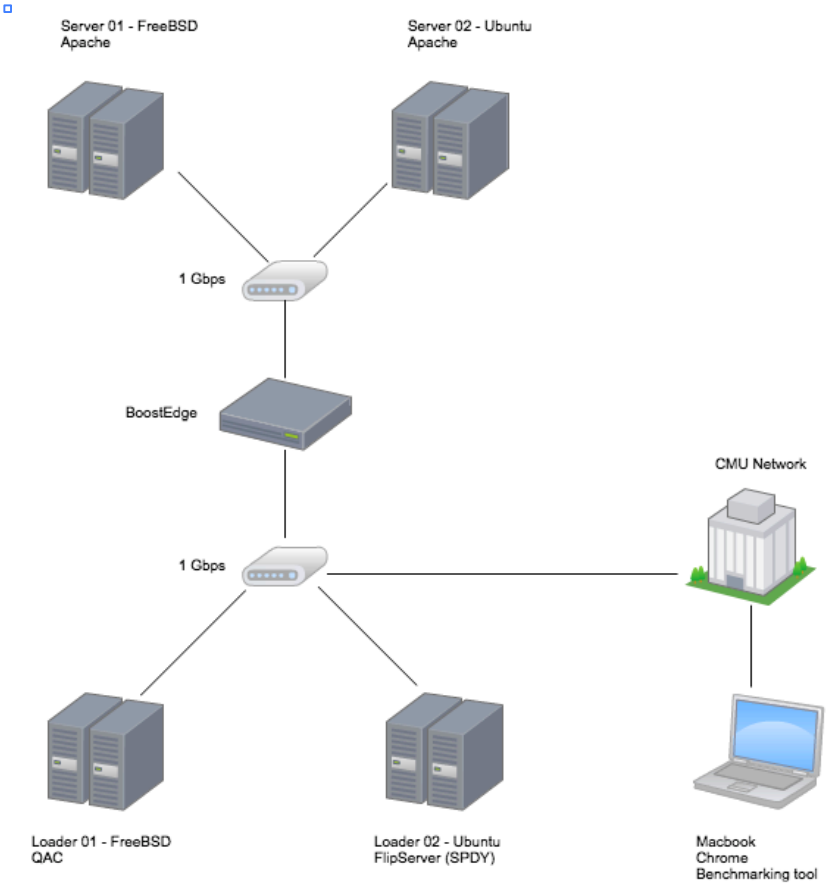


Figure 1 - Testbed Description

The two loaders can also act as clients to generate web traffic, if desired. In addition, Loader01 contains the QAC tool, an internal testing tool developed by ActivNetworks that simulates and measures web traffic. Loader02 hosts the Flip server, which is built-in with the SPDY protocol.

When requests are generated by a Chrome Browser on an external machine, and routed through the loader containing the Flip server, on their way to the BoostEdge device and the web servers, a SPDY connection can be established between the Flip server and the external machine's Chrome browser. The output of the Flip server to/from the BoostEdge device and to the web servers will be HTTP.

The two servers hold copies of the 45 most-visited websites in the US, downloaded to a depth of two. Selecting websites with homepage weights greater than 200 Kb, these 45 websites were derived from an initial list of the 100 top US websites selected using the alexa.com index [3]. YSlow[4], a Google Chrome tool, was used to determine homepage weights for all websites.

The BoostEdge appliance, which sits in the center of the testbed, when set to the "pass through mode" ensures that traffic passing through the appliance, goes untouched. When set in the active mode, all traffic to the servers first passes through BoostEdge, which compresses and optimizes the traffic from the servers.

An external machine running Google Chrome generates web requests, measures web traffic, and gives us the ability to throttle the CMU SV network as needed. It connects to the CMU SV Ethernet and wireless networks. The wireless network is rated between 20 and 100 Mbit/sec. Our measurements of network speeds show that we achieve speeds of 20Mbps to 72Mbps while connecting to the testbed to run tests. This being greater than our desired speed of 1 Mbps, we throttle the network to achieve our desired speed.

The current benchmarking protocol only required us to use a subset of the testbed (Figure 2). A single server was loaded with a number of large, popular sites with different content and transaction characteristics. The external machine running Google Chrome was used to generate web requests for these websites. Since the QAC tool on Loader01 and one of the web servers were not used for these tests, a revised flow diagram of the testbed is used in the subsequent sections to make the test protocols easier to follow.

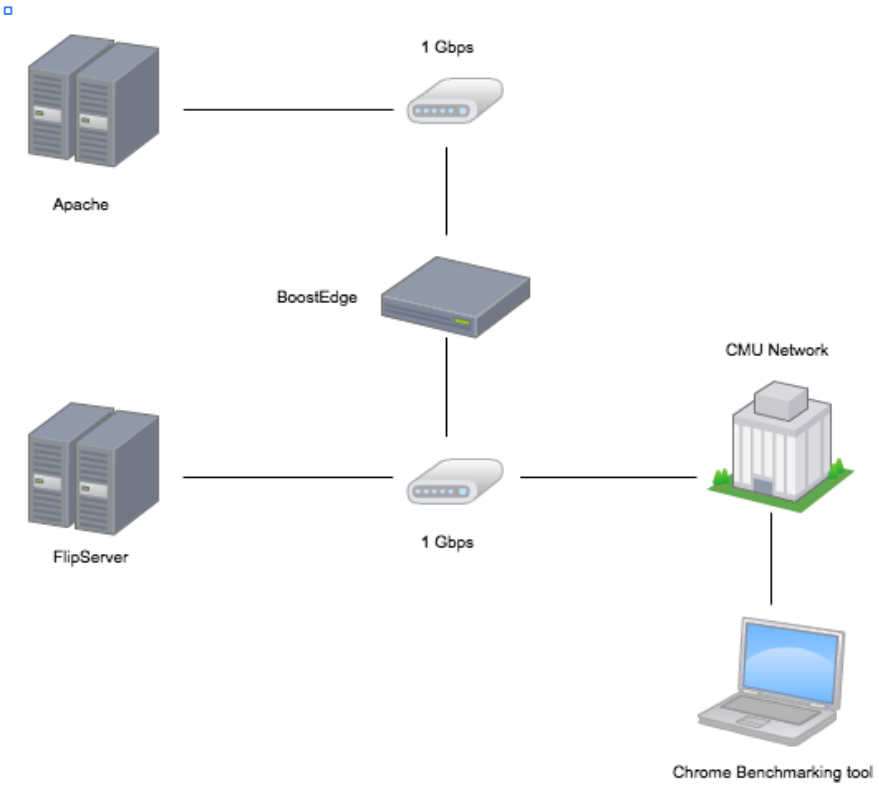


Figure 2 – Components of the testbed used for benchmarking

## III. Website Selection

Selecting websites to represent a real-world scenario was of essence to our study. The testbed was populated with working replicas of 25 websites using the following methodology:

1. After identifying the 100 most visited websites using the Alexa Traffic Rank, we downloaded copies (to a depth of 2) to our test bed servers using Wget[5]. In addition to creating local copies of the websites, Wget also cleaned up most external resource requests in each website.
2. For each downloaded website, we measured the home page weight and the number of HTTP requests on our local server. YSlow, a Chrome add-on, was used to determine the above two metrics.
3. Websites with a Home Page weight lower than 200 Kb were discarded. The remaining websites were then ranked by their homepage weights and 45 websites with the highest weights were shortlisted.
4. Since Wget only partially redirects external requests to the local machine, the websites were then manually scrutinized to redirect the remaining external requests to the local server. Applying our initial selection criterion of home page weight being greater than or equal to 200 Kb, 40 websites remained at this stage.
5. The 40 websites were then benchmarked using home page weights and HTTP requests using YSlow, as was done earlier. The 25 websites with the highest home page weight formed our final test-set.

HTTP requests were considered as a secondary metric to give a different dimension to our tests, after narrowing our test-set to 25 websites using the primary selection criteria of page weight being greater than 200 Kb. Table 1 documents both these metrics for the websites.

An external machine with the Chromium Page Benchmarking Tool[6] in a Chrome browser was used to generate the web requests. The Mean Doc Load Time, which is the time taken for the document object model (DOM) to load completely, was then recorded for each web request. A speed-up % was calculated to clearly distinguish the effect of using SPDY and/or BoostEdge, and the same was depicted graphically and in tables.

| Website | Home Page Size (KB) | HTTP Requests |
|---|---|---|
| LATimes | 2049.6 | 160 |
| **WashingtonPost** | 1487.6 | 109 |
| **CNN** | 1379.8 | 109 |
| **MapQuest** | 1231.7 | 78 |
| **Answers** | 1035.2 | 39 |
| **Ehow** | 1006 | 27 |
| **FoxNews** | 981.8 | 45 |
| **Yahoo** | 948.4 | 50 |
| **SalesForce** | 944.1 | 52 |
| **Adobe** | 913.6 | 77 |
| **GoDaddy** | 811.5 | 52 |
| **BBC** | 799.1 | 79 |
| **Yelp** | 773.8 | 45 |
| **CareerBuilder** | 638 | 36 |
| **AOL** | 628.1 | 23 |
| **ATT** | 601.9 | 57 |
| **BestBuy** | 599.1 | 75 |
| **Ebay** | 562.9 | 41 |
| **Comcast** | 440.3 | 51 |
| **Wordpress** | 394.8 | 46 |
| **USPS** | 356.1 | 86 |
| **FoxSport** | 346.9 | 71 |
| **TypePad** | 330.4 | 39 |
| **Cj** | 330 | 88 |
| **Huffingtonpost** | 329.3 | 32 |
| **Zynga** | 321.4 | 26 |
| **Clickbank** | 304 | 38 |
| **Pandora** | 274.4 | 32 |
| **NYTimes** | 256.6 | 114 |
| **NFL** | 233.5 | 45 |
| **Go** | 218.9 | 21 |
| **Weather** | 196.7 | 37 |
| **ESPN** | 132.5 | 33 |
| **CBSSports** | 130.3 | 30 |
| **IMDB** | 123 | 42 |
| **Youtube** | 113.9 | 16 |
| **Digg** | 102.1 | 25 |
| **Photobucket** | 60.8 | 38 |
| **Twitter** | 54.4 | 29 |
| **MySpace** | 46.9 | 25 |

Table 1: 40 Websites with most traffic – top 25 with the heaviest home page weights

## IV. Dataflow

Below are the data flow diagrams for the three test cases.

### 1. BoostEdge vs. HTTP

Figure 3 below shows the flow of data for the test case designed to compare the performance of BoostEdge against HTTP. In both conditions, there is two-way network traffic from the Chrome Browser through BoostEdge to the web server. In the "BoostEdge on" case, BoostEdge is active and acts on the data, compressing and optimizing the traffic back to the Chrome browser. In the HTTP case, BoostEdge operates in the "transparent mode", merely passing any traffic through the network, without touching it. Thus the right way to look at this is to think of the BoostEdge appliance being located just in front of or behind an internet gateway for the web server, while the connection from BoostEdge to the Chrome Browser can be any arbitrary internet distance (latency) and connection speed.

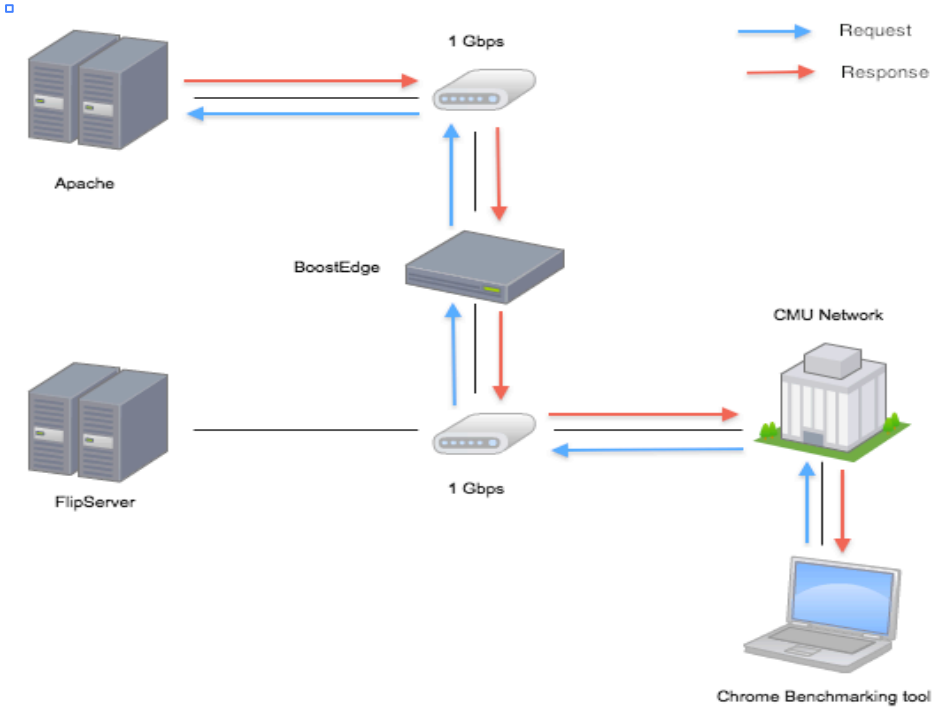


Figure 3 – Dataflow for BoostEdge vs. HTTP comparison

### 2. SPDY vs. HTTP

Figure 4 below shows the flow of data for the test case designed to compare the performance of SPDY against HTTP. In both cases, the data first goes through the Flip Server, then through BoostEdge, to the web server, and then back again. In both cases, BoostEdge operates in "transparent mode", merely passing any traffic through the network, without touching it. For the case of SPDY on, the Chrome browser establishes a SPDY connection with the Flip server, and traffic between these two points uses the SPDY protocol. The Flip Server emits HTTP for the network link to the Web server, and receives HTTP back.

Thus the right way to look at this is to think of the Flip Server being located just in front of or behind the internet gateway for the web server, while the connection from the Flip Server to the Chrome Browser can be any arbitrary internet distance (latency) and connection speed.

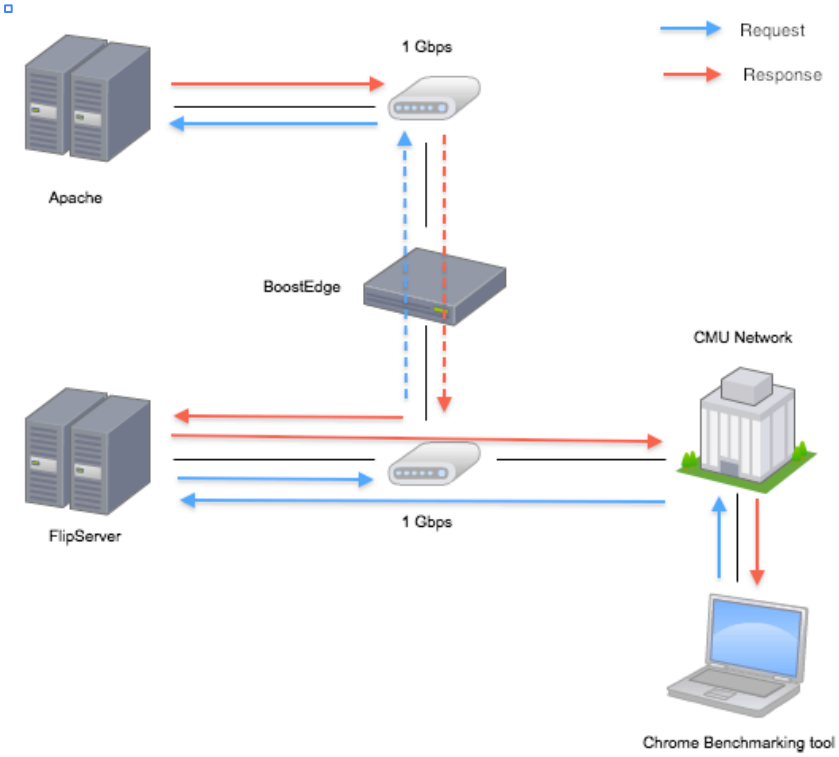


Figure 4 – Dataflow for SPDY vs. HTTP comparison

### 3. SPDY and BoostEdge vs. HTTP

Figure 5 below shows the data flow for the BE+SPDY vs. HTTP comparison. In both cases, the data first goes through the Flip Server, then through BoostEdge to the web server, and then back again. In the first case, SPDY and BoostEdge are "on", the Chrome browser establishes a SPDY connection with the Flip server, and traffic between these two points uses the SPDY protocol. The Flip Server emits HTTP, which goes through BoostEdge to the web server. On the return trip, BoostEdge processes the HTTP, compressing and optimizing it and sending it on to the Flip server. The Flip server in turn takes the optimized HTTP, and applies the SPDY protocol for the connection back to the Chrome browser. In the HTTP case, both the Flip server and BoostEdge operate in "transparent mode", merely passing any traffic through the network, without touching it. Thus the right way to look at this is to think of the Flip Server and BoostEdge being located just in front of or behind the internet gateway for the web server, while the connection from the Flip Server to the Chrome Browser can be any arbitrary internet distance (latency) and connection speed.

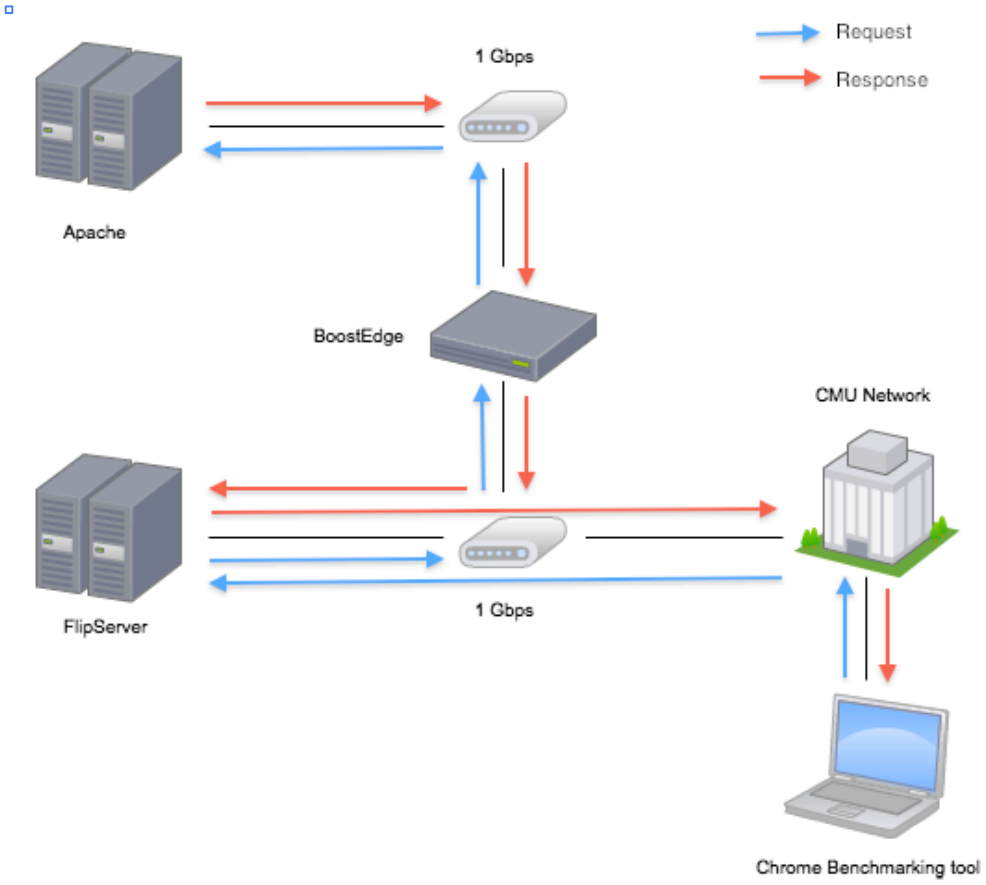


Figure 5 – Dataflow for BoostEdge and SPDY combined vs. HTTP comparison

## V. Test Profiles and Findings

The benefits of SPDY and BoostEdge are dependent on the nature of the network session. For instance, with a gigabyte network only extremely heavy loads would show apparent benefits in performance. Hence, in our testing profile, we constrained the bandwidth to 1 Mbps with a latency of 100 msec. Although BoostEdge has a wide range of settings that optimize overall performance in saturated and low speed networks, since our testbed did not allow for saturating the BoostEdge load sufficiently to require tuning the performance, we used the default or recommended settings for our tests.

In table 2 below, we set out the 6 test conditions used in all test cases. This testing sequence was constant for all test cases. For each test condition, the activity status of the two protocols (SPDY & HTTP) along with the BoostEdge status is noted in the table.

| Test Condition | Iterations | Protocol | SPDY Status | Through FlipServer | BoostEdge Status | BoostEdge Compression |
|---|---|---|---|---|---|---|
| 1 | 10 | SPDY | ON | Yes | OFF | n/a |
| 2 | 10 | SPDY | OFF | Yes | OFF | n/a |
| 3 | 10 | SPDY | ON | Yes | ON | MAX |
| 4 | 10 | SPDY | OFF | Yes | ON | MAX |
| 5 | 10 | HTTP | n/a | No | OFF | n/a |
| 6 | 10 | HTTP | n/a | No | ON | MAX |

Table 2: Test conditions

Table 2 depicts the six test scenarios listed below (in the same order) for which the Mean Doc Load Times were recorded for the websites in our test-set, relative to HTTP:

1. HTTP
2. HTTP through the Flip server
3. SPDY through the Flip server
4. SPDY through the Flip server via BoostEdge
5. HTTP through the Flip server via BoostEdge
6. HTTP via BoostEdge

Our study analyzes three different behaviors of the testbed (described in detail below):

**Test Case 1:** Average Performance

**Test Case 2:** Speed-up introduced primarily by BoostEdge

**Test Case 3:** Speed-up introduced primarily by SPDY

## Test Case 1: Average Performance

To represent the typical behavior of the effects of SPDY and BoostEdge – individually and in combinations, we used the following test protocol:

1. **Average Performance without Packet Loss:** Using the Page Benchmarking Tool in Google Chrome on the external machine to generate requests, the 25 websites in our test bed were tested individually over 10 iterations and their Mean Doc Load Time was recorded. The bandwidth was throttled to 1 Mbps with no set packet loss. Testing was done using a direct Ethernet connection.

| Throttling Settings | |
|---|---|
| Throttling - Bandwidth | 1 Mbps |
| Throttling - Delay | 100 ms |
| Packet loss set | 0.00% |
| Packet loss Measured* | 0.0% - 0.1% |
| *Packet loss measured over a 1000 ping requests | |

Table 3: Throttling Settings – Average performance without packet loss

The results for average web performance without packet loss are tabulated below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 8924.36 | N/A | 4.47 |
| 2. HTTP through FlipServer | 9892.932 | -10.8531256 | 5.29 |
| 3. SPDY through FlipServer | 8676.04 | 2.782496448 | 2.75 |
| 4. SPDY through FlipServer w BoostEdge | 3965.236 | 55.5683993 | 2.01 |
| 5. HTTP through FlipServer w BoostEdge | 5266.584 | 40.98642368 | 3.22 |
| 6. HTTP w BoostEdge | 4425.6 | 50.40988934 | 4.66 |

Table 4: Result for Average performance without packet loss

## Results and Analysis:

1. **HTTP:** Test condition 1 represents our default condition - "plain old" HTTP.
2. **HTTP through Flip Server:** This condition shows the effect when we route the network through the Flip Server, when still running HTTP. Looking at the impact of using the Flip Server, we can see that we are paying about an 11% cost. Some of this is due to the added time of running through the Flip Server, but is primarily due to the fact that the Flip Server uses SSL.
3. **SPDY through Flip Server:** Turning SPDY on and passing it through the Flip Server results in a net gain of 2.8% over plain HTTP, in condition 1. When comparing against condition 2, and taking into account that this includes an 11% cost of running SSL, SPDY offers a significant improvement in performance over HTTP, of ~13.8%.
4. **SPDY through Flip Server with BoostEdge:** The 4$^{th}$ condition shows the change in performance when we now turn on BoostEdge. We see a further improvement of ~55.6% in performance over HTTP.
5. **HTTP through Flip Server with BoostEdge:** Looking at the 5$^{th}$ condition, we can compare HTTP through the Flip Server performance, to HTTP through the Flip Server with BoostEdge performance on average page load times (conditions 2 & 5) as well as HTTP versus SPDY performance with BoostEdge turned on (conditions 4 & 5). In the first case, we can see that when the HTTP traffic is going through the Flip server, HTTP with BoostEdge improves average page load times over plain HTTP by an average of ~52%. Since the traffic uses SSL when the Flip server is used, this is really looking at the effects of using BoostEdge with SSL. Looking at the effect of also turning on SPDY, we can see that SPDY provides an additional 15% boost in performance over HTTP when BoostEdge is also used. This is consistent with the 13% improvement in test condition #2.
6. **HTTP with BoostEdge:** Looking at the 6$^{th}$ condition, HTTP with BoostEdge without using the Flip Server, we can see an improvement due to BoostEdge of 50.4% in average page load times. This is consistent with the ~52% improvement seen in test condition 5 as compared to condition 2. The benefits of SPDY and BoostEdge over plain HTTP in average page load

times appear to be consistent and additive, to within a few percent, across all the test conditions.

We can see these results clearly when viewed as a chart:

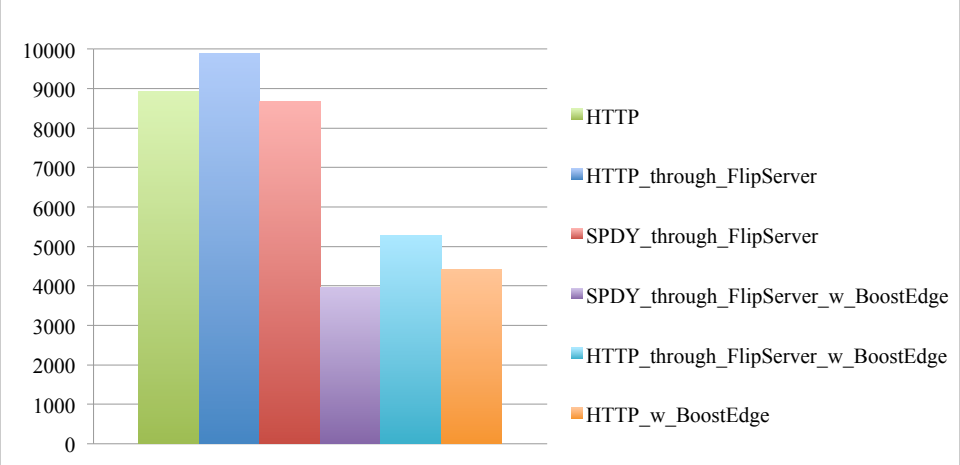


Chart 1: Graphical Representation of Result – Average performance without packet loss

**Result Summary:** These results show that for our testing profile of 1 Mbps and 100 ms delay, with no packet loss, for 25 websites, we gain the largest improvement in average page load time using BoostEdge together with SPDY. We can conclude that the data optimization contribution of BoostEdge and the network optimization contribution of SPDY appear to be complementary and additive in effect. Both approaches show significant improvements in performance individually as well. These results are averaged over 25 websites and, of course, the results in any particular case will vary somewhat from the average. We include the specific measurements for all 25 websites in Appendix 1.

2. **Average Performance with Packet Loss:** Using the Page Benchmarking Tool in Google Chrome on the external machine, 25 websites in our testbed were tested individually over 10 iterations and their Mean Doc Load Time was recorded. In the real world, particularly when using WiFi or mobile networks, there can often be packet loss. Thus, we introduced a typical ~1.0% packet loss (Jim Roskind, JAR@Google.com, personal communication, August 22, 2011) on the network throttled to a bandwidth of 1 Mbps. Testing was done using a direct Ethernet connection.

| **Throttling Settings** | |
|---|---|
| **Throttling - Bandwidth** | 1 Mbps |
| **Throttling - Delay** | 100 ms |
| **Packet loss set** | 0.50% |
| **Packet loss Measured*** | 0.9% - 1.0% |
| *Packet loss measured over a 1000 ping requests | |

Table 5: Throttling Settings – Average performance with packet loss

The results for average web performance with packet loss are tabulated below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 9179.8 | N/A | 5.71 |
| 2. HTTP_through_FlipServer | 10535.6 | -14.76938495 | 7.9 |
| 3. SPDY_through_FlipServer | 9404.3 | -2.44558705 | 8.02 |
| 4. SPDY_through_FlipServer_w_BoostEdge | 4275.3 | 53.42708991 | 9.72 |
| 5. HTTP_through_FlipServer_w_BoostEdge | 5443.1 | 40.70567986 | 5.27 |
| 6. HTTP_w_BoostEdge | 4677.1 | 49.05008824 | 6.02 |

Table 6: Result for Average performance with packet loss

## Results and Analysis:

**General Observation:** The first thing to note is that the introduction of packet loss, which is a random occurrence, increases the error bars and also the average page load times. These effects are expected - random packet loss will of course increase the error rate, while lost packets require additional time to re-establish.

1. **HTTP:** Test condition 1 again represents the default condition, "plain old" HTTP.
2. **HTTP through Flip Server:** Test condition 2 shows the effect when we route the network traffic through the Flip Server running SSL. Looking at the effect of using the Flip Server, we can see that we are paying about a 15% cost, an increase of 4% over the condition of no packet loss. Since SPDY uses fewer connections than HTTP, to enhance efficiency, the effect of dropped packets can be more disruptive, which is the result we are seeing.
3. **SPDY through Flip Server:** Next, in the 3rd condition, we turn on SPDY, and we can compare the effect of SPDY vs. HTTP, when using the Flip Server. Here we see that there is about ~2.5% loss over HTTP, but compared to running HTTP through the Flip server using SSL, SPDY offers a 12.5% improvement in performance, consistent with the case of no packet loss. It appears, under our condition of packet loss that there is degradation in performance due to packet loss, but the relative advantage of using SPDY over HTTP when using the Flip Server is comparable to the same test condition with no packet loss.
4. **SPDY through Flip Server with BoostEdge:** The 4th condition shows the change in performance when we now turn on BoostEdge in addition to SPDY. We see an improvement of ~53% in performance over HTTP (condition 1), and ~51% compared to just SPDY. These results are consistent with the results found when there is no packet loss (55.5% and 53% respectively).
5. **HTTP through Flip Server with BoostEdge:** Looking at the 5th condition, we can compare HTTP through the Flip Server performance, to HTTP through the Flip Server w. BoostEdge performance on average page load times (conditions 2 & 5) as well as HTTP versus SPDY performance with BoostEdge turned on (conditions 4 & 5). In the first case, we can see that HTTP with BoostEdge improves average page load times over plain HTTP by an average of ~55%, when using the Flip Server. This shows that the improvement due to BoostEdge is

consistent whether the traffic is routed through the Flip Server or not, and consistent whether there is packet loss or not. Looking at the effect of also turning on SPDY, we can see that SPDY provides an additional ~13% boost in performance over HTTP when BoostEdge is also used. This is consistent with what we see when there is no packet loss (~15% improvement).

6. **HTTP with BoostEdge:** Looking at the 6th condition, HTTP with BoostEdge without using the Flip Server, we can see an improvement due to BoostEdge of 49% in average page load times, slightly less than the case of no packet loss (~50%). The "cost" of using the Flip server and SSL is about ~10% in this case, (conditions 5 & 6), which is consistent with the cost (~11%) when there is no packet loss.

We can see these results more clearly when viewed as a chart:

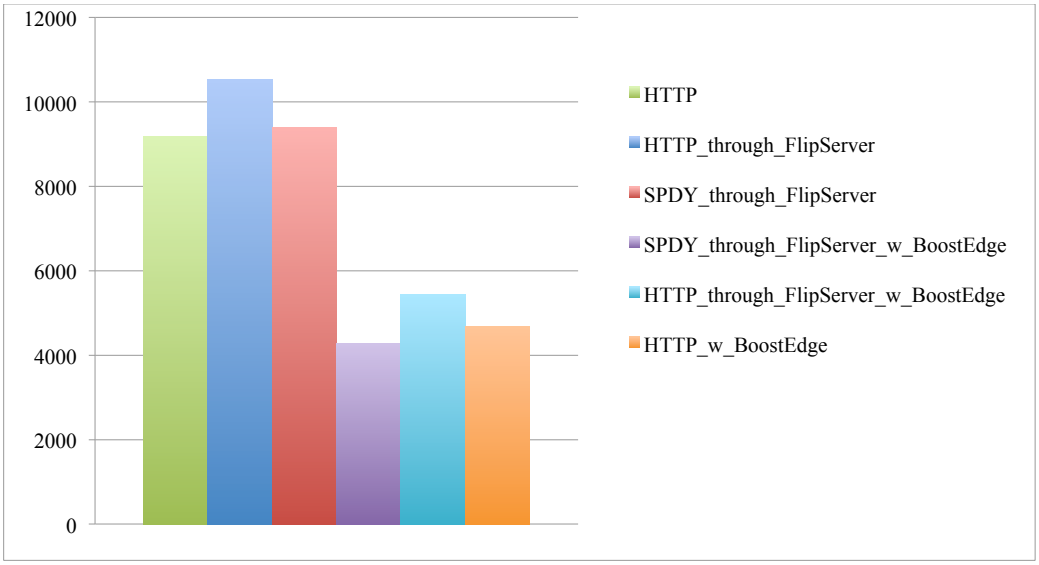


Chart 2: Graphical Representation of Result – Average performance with packet loss

**Result Summary:** These results show that for our testing profile of 1 Mbps and 100 ms delay, when packet loss of ~1% is added to our 25 websites, we still gain the largest improvement in average page load time using BoostEdge together with SPDY, while gaining the benefits of SSL. That is, the data optimization contribution of BoostEdge and the network optimization contribution of SPDY still appear to be complementary and additive in effect, at about the same levels as with no packet loss. The introduction of packet loss does increase the average page load times and slightly reduce some performance benefits of SPDY and BoostEdge. Both approaches show significant improvements in performance individually as well. These results are average results over all 25 websites, and of course the results in any particular case will vary somewhat from the average. We include the specific measurements for all 25 websites in Appendix 2.

**Test Case 2: Speed-up introduced primarily by BoostEdge**

The above results describe an average performance over 25 websites. This has generated robust results - the result paradigm persisted over successive refinements of test profiles in the process of attaining the final testbed of 25 websites (i.e. reducing error bars, replacing bad websites and removing external resource calls did not alter the relative nature of speed-up from plain HTTP even though the absolute magnitude of measurements varied). However, averaging obscures the variation among individual websites.

To provide more insight, the speed-up introduced by BoostEdge was studied in more detail by comparing the performance on targeted websites with high data content, where BoostEdge should show an inherent advantage, against websites with relatively low data content, by using the following testing protocol:

1. 5 websites with the highest home page weights (amongst the 25) were tested over 10 iterations each, using a 1 Mbps network without packet loss, using the Page Benchmarking Tool in Google Chrome on the external machine to record their Mean Doc Load Times. Testing was done using a direct Ethernet connection.
2. Similarly, 5 websites with the lowest home page weights (amongst the 25) were tested over 10 iterations using a 1 Mbps network without packet loss, using the Page Benchmarking Tool in Google Chrome on the external machine to record their Mean Doc Load Times. Testing was done using a direct Ethernet connection.

The test profile for the above test scenarios is as below:

| **Throttling Settings** | |
|---|---|
| **Throttling - Bandwidth** | 1 Mbps |
| **Throttling - Delay** | 100 ms |
| **Packet loss set** | 0.00% |
| **Packet loss Measured*** | 0.0% - 0.1% |
| *Packet loss measured over a 1000 ping requests | |

Table 7: Throttling settings to measure speed-up introduced primarily by BoostEdge

The five websites with the **highest home page weight** in our testbed are tabulated below:

| Website | Homepage (Kb) | HTTP Requests |
|---|---|---|
| LATimes | 2049.6 | 160 |
| WashingtonPost | 1487.6 | 109 |
| CNN | 1379.8 | 109 |
| MapQuest | 1231.7 | 78 |
| Answers | 1035.2 | 39 |

Table 8: Test websites with the highest Home Page weight - to measure speed-up introduced primarily by BoostEdge

The results for the 5 heaviest websites are shown in Table 9 below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 15778 | N/A | 3.5 |
| 2. HTTP_through_FlipServer | 16962.5 | -7.50728863 | 2.7 |
| 3. SPDY_through_FlipServer | 15180.5 | 3.786918494 | 1.99 |
| 4. SPDY_through_FlipServer_w_BoostEdge | 6200.1 | 60.70414501 | 1.86 |
| 5. HTTP_through_FlipServer_w_BoostEdge | 8220.8 | 47.89707187 | 1.7 |
| 6. HTTP_w_BoostEdge | 7249.1 | 54.0556471 | 3.3 |

Table 9: Results to study the speed-up introduced primarily by BoostEdge

## Results and Analysis:

1. **HTTP:** Test condition 1 again represents the default condition, "plain old" HTTP. Not surprisingly, the heaviest pages show page load times higher than the average for all 25 websites.
2. **HTTP through Flip Server:** Test condition 2 shows the effect when we route the network through the Flip Server running SSL. Looking at the effect of using the Flip Server, we can see that we are paying about a 7.5% cost.
3. **SPDY through Flip Server:** Next, in the 3rd condition, we turn on SPDY, and we can compare the effect of SPDY vs. HTTP, when using the Flip Server. Here we see that there is a net gain of ~4% over HTTP. However, if we take into account the cost shown in test condition 2 of running SSL, SPDY offers an improvement in performance of ~11% over plain HTTP using SSL for these heaviest websites.
4. **SPDY through Flip Server with BoostEdge:** The 4th condition shows the change in performance when we now turn on BoostEdge. We see an improvement of ~61% in performance over HTTP (condition 1).
5. **HTTP through Flip Server with BoostEdge:** Looking at the 5th condition, we can compare HTTP through the Flip Server performance, to HTTP through the Flip Server with BoostEdge performance on average page load times (conditions 2 & 5) as well as HTTP versus SPDY performance with BoostEdge turned on (conditions 4 & 5). In the first case, we can see that HTTP with BoostEdge improves average page load times over plain HTTP by an average of 55.5%, when using the Flip Server, consistent with the previous measurements. Looking at the effect of also turning on SPDY, we can see that SPDY provides an additional 13% boost in performance over HTTP when BoostEdge is also used, also consistent with previous measurements.
6. **HTTP with BoostEdge:** Looking at the 6th condition, HTTP with BoostEdge without using the Flip Server, we can again see an improvement due to BoostEdge of 54% in average page load times, a slight improvement over previous measurements, perhaps because BoostEdge's data compression capabilities have more effect on performance with heavier pages.

Below is a graphical representation of the result:

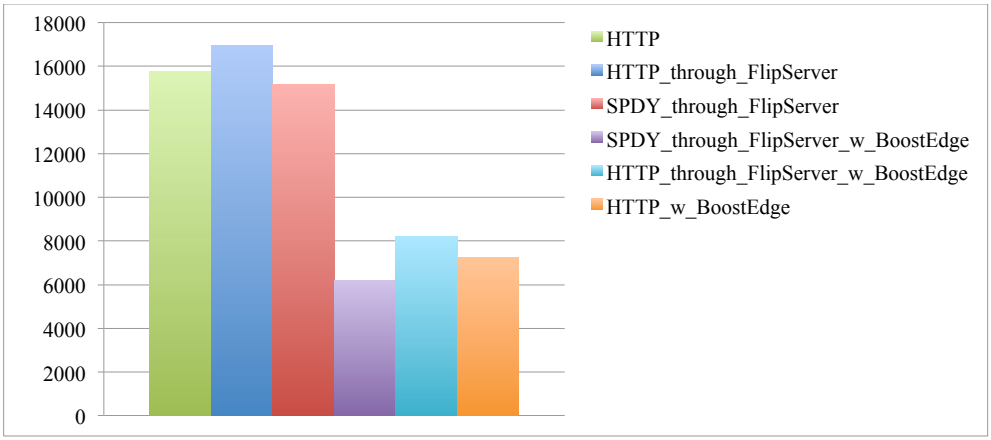


Chart 3: Graphical representation of results to study the speed-up introduced primarily by BoostEdge

**Result Summary:** These results show that for the heaviest websites, we still gain the largest improvement in average page load time using BoostEdge together with SPDY. That is, the data optimization contribution of BoostEdge and the network optimization contribution of SPDY still appear to be complementary and additive in effect. Both approaches show significant improvements in performance individually as well, although for the heaviest websites, BoostEdge offers an increased advantage, while the SPDY advantage is a bit reduced, compared to the averages over all websites.

The five websites with the **lowest home page weight** in our testbed are tabulated below; these are roughly 10% - 30% of the weights of the 5 heaviest:

| Website | Homepage (Kb) | HTTP Requests |
|---|---|---|
| USPS | 356.1 | 86 |
| FoxSport | 346.9 | 71 |
| TypePad | 330.4 | 39 |
| Cj | 330 | 88 |
| Huffingtonpost | 329.3 | 32 |

Table 10: Test websites with the lowest Home Page weight - to measure speed-up introduced primarily by BoostEdge

The results for 5 lightest websites are shown in Table 11 below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 5536.1 | N/A | 1.65 |
| 2. HTTP_through_FlipServer | 6151.9 | -11.12335399 | 4.71 |
| 3. SPDY_through_FlipServer | 5218.4 | 5.738696917 | 2.86 |
| 4. SPDY_through_FlipServer_w_BoostEdge | 3201.3 | 42.17409368 | 3.61 |
| 5. HTTP_through_FlipServer_w_BoostEdge | 4237.1 | 23.46417153 | 2.62 |
| 6. HTTP_w_BoostEdge | 3645.6 | 34.14858836 | 1.75 |

Table 11: Results to study the speed-up introduced primarily by BoostEdge

## Results and Analysis:

1. **HTTP:** Test condition 1 again represents the default condition, "plain old" HTTP. Not surprisingly, the lightest pages show page load times shorter than the average for all 25 websites.
2. **HTTP through Flip Server:** Test condition 2 shows the effect when we route the network through the Flip Server running SSL. Looking at the effect of using the Flip Server, we can see that we are paying about an 11% cost.
3. **SPDY through Flip Server:** Next, in the 3rd condition, we turn on SPDY, and we can compare the effect of SPDY vs. HTTP, when using the Flip Server. Here we see that there is a net gain of ~6% over HTTP. However, if we take into account the cost shown in test condition 2, an 11% cost of running SSL, SPDY offers a significant improvement in performance of 17% over plain HTTP for these lightest websites.
4. **SPDY through Flip Server with BoostEdge:** The 4th condition shows the change in performance when we now turn on BoostEdge. We see an improvement of ~42% in performance over HTTP (condition 1). Comparing this result to the third condition shows the added impact of turning on BoostEdge, when SPDY is operating. This results in a 36.5% gain, somewhat less than the performance improvement with the heaviest websites.
5. **HTTP through Flip Server with BoostEdge:** Looking at the 5th condition, we can compare HTTP through the Flip Server performance, to HTTP through the Flip Server with BoostEdge performance on average page load times (conditions 2 & 5) as well as HTTP versus SPDY performance with BoostEdge turned on (conditions 4 & 5). In the first case, we can see that HTTP with BoostEdge improves average page load times over plain HTTP by an average of 35%, when using the Flip server, consistent with our measurement in the fourth condition. Looking at the effect of also turning on SPDY, we can see that SPDY provides an additional 19% boost in performance over HTTP when BoostEdge is also used. This is again consistent with the measurement we saw in condition 3.
6. **HTTP with BoostEdge:** Looking at the 6th condition, HTTP with BoostEdge without using the Flip Server, we can again see an improvement due to BoostEdge of 34% in average page load times, consistent with the improvement we saw when traffic is running through the Flip server.

Below is a graphical representation of the result:

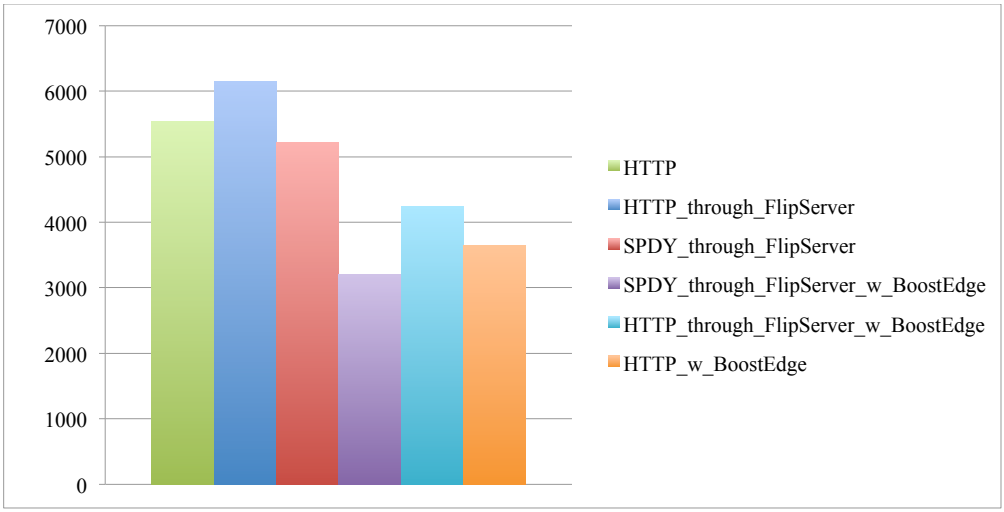


Chart 4: Graphical representation of results to study the speed-up introduced primarily by BoostEdge

**Result Summary:** These results show that for the lightest websites, we still gain the largest improvement in average page load time using BoostEdge together with SPDY. That is, the data optimization contribution of BoostEdge and the network optimization contribution of SPDY still appear to be complementary and additive in effect. Both approaches show significant improvements in performance individually as well, although for the lightest websites, SPDY offers an increased advantage, while the BoostEdge advantage is reduced, compared to the averages over all websites. This is the opposite result to the case of the heaviest websites. It appears that as websites get heavier, BoostEdge has a larger effect and SPDY less of an effect, but combined, performance differences due to changes to website heaviness balance out when using both BoostEdge and SPDY together.

**Test Case 3: Speed-up introduced primarily by SPDY**

In Test Case 2, the relative frequency of HTTP requests required for a website almost always mirrored the relative weight of the home page (Tables 8 and 10). Thus, the results in Test Case 2 for the performance of the top and bottom 5 websites among our top 25, as measured by home page weight would also hold true if we had ranked those websites by # of HTTP requests instead. However, if we go back to our top 25 websites used in Test Case 1, we would find that the top and bottom 5 websites, when ranked by # of HTTP requests are not identical to the top and bottom 5 websites, when ranked by home page weight. Therefore, we ran tests over the 5 top and bottom websites that would have been chosen from our final set of 25, if the only criteria was # of HTTP requests. These tests were run with the same settings as in Test Case 2.

Below are the Websites used for highest # and lowest # of HTTP requests, in this test. It can be seen that the relative ordering of home page weights frequently do not correspond to the ordering by HTTP requests,

although once again, the top and bottom groups are distinct.

The speed-up introduced by SPDY on websites with a large number of requests to the server where SPDY shows an inherent advantage, versus websites with few server requests, was studied in more detail by using the following testing protocol:

1. 5 websites with the highest # of HTTP requests (amongst the 25) were tested over 10 iterations using a 1 Mbps network, both without packet loss and with ~1.00% packet loss, using the Page Benchmarking Tool in Google Chrome on the external machine to record Mean Doc Load Times. Testing was done using a direct Ethernet connection.
2. Similarly, 5 websites with the lowest # of HTTP requests (amongst the 25) were tested over 10 iterations using a 1 Mbps network both without packet loss and with ~1.00% packet loss, using the Page Benchmarking Tool in Google Chrome on the external machine to record Mean Doc Load Times. Testing was done using a direct Ethernet connection.

The 5 websites with the highest number of HTTP requests are tabulated below:

| Website | Homepage (Kb) | HTTP Requests |
|---|---|---|
| LATimes | 2049.6 | 160 |
| WashingtonPost | 1487.6 | 109 |
| CNN | 1379.8 | 109 |
| Cj | 330 | 88 |
| USPS | 356.1 | 86 |

Table 12: Test websites with the highest number of HTTP Requests - to measure speed-up introduced primarily by SPDY

The results for 5 websites with the highest number of HTTP requests are shown in Table 13 below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 13685.1 | N/A | 3.31 |
| 2. HTTP_through_FlipServer | 14839.8 | -8.437643861 | 1.3 |
| 3. SPDY_through_FlipServer | 12757.2 | 6.780366968 | 1.54 |
| 4. SPDY_through_FlipServer_w_BoostEdge | 5846.2 | 57.280546 | 2.3 |
| 5. HTTP_through_FlipServer_w_BoostEdge | 8112.1 | 40.72312223 | 2.13 |
| 6. HTTP_w_BoostEdge | 7116.6 | 47.99745709 | 1.69 |

Table 13: Results with the highest number of HTTP Requests - to measure speed-up introduced primarily by SPDY

The 5 websites with the lowest number of HTTP requests are tabulated below:

| Website | Homepage (Kb) | HTTP Requests |
|---|---|---|
| TypePad | 330.4 | 39 |
| CareerBuilder | 638 | 36 |
| Huffingtonpost | 329.3 | 32 |
| Ehow | 1006 | 27 |
| AOL | 628.1 | 23 |

Table 14: Test websites with the lowest number of HTTP Requests - to measure speed-up introduced primarily by SPDY

The results for 5 websites with the lowest number of HTTP requests are shown in Table 15 below:

| Testcase | Average Page Load time (ms) | Speed-up over HTTP % | Error % |
|---|---|---|---|
| 1. HTTP | 7211.4 | N/A | 3.23 |
| 2. HTTP_through_FlipServer | 8021.9 | -11.23914912 | 3.48 |
| 3. SPDY_through_FlipServer | 7384.5 | -2.400366087 | 0.89 |
| 4. SPDY_through_FlipServer_w_BoostEdge | 3011.7 | 58.23695815 | 1.33 |
| 5. HTTP_through_FlipServer_w_BoostEdge | 3797.8 | 47.33616219 | 3.43 |
| 6. HTTP_w_BoostEdge | 3174.1 | 55.98496824 | 3.16 |

Table 15: Results with the lowest number of HTTP Requests - to measure speed-up introduced primarily by SPDY

## Results and Analysis:

1. **HTTP:** Looking at test condition 1, which again represents the default condition, "plain old" HTTP, we see that websites with more HTTP requests take significantly longer than those that have few HTTP requests. No surprises here.
2. **HTTP through Flip Server:** Looking at test condition 2, we see that the cost to running HTTP through the Flip server using SSL is ~8.5% for the highest 5 websites, and 11% for the lowest 5 websites. Thus the effect when we route the HTTP network traffic through the Flip Server is consistent with the earlier test cases, with a proportionally higher cost for those websites with fewest HTTP requests.
3. **SPDY through Flip Server:** Next, in the 3rd condition, we turn on SPDY, and we can compare the effect of SPDY vs. HTTP, when using the Flip Server. Here we see that there is a net gain of ~15% for the highest 5 websites, and only ~9% for the lowest websites with the least HTTP requests. These results are consistent with the earlier test cases, and show that SPDY provides more benefit when more HTTP requests are necessary. Since SPDY is designed to optimize network performance, this result is consistent with SPDY's design.
4. **SPDY through Flip Server with BoostEdge:** The 4th condition shows the change in performance when we now turn on BoostEdge. We see an improvement of ~44% in performance over SPDY alone in the case of the highest 5 sites, and an improvement of 60% in the case of the 5 lowest sites. However, the improvement over HTTP (test condition 1) is almost the same for both sets of websites: 57% vs 58%. This is because, as we saw when looking at the heaviest vs. lightest websites, in Test Case 2, when there is less opportunity for network optimization, the advantage of data optimization via BoostEdge increases, and vice versa.
5. **HTTP through Flip Server with BoostEdge:** Looking at the 5th condition, we can see this clearly, as we measure the effects of BoostEdge by comparing HTTP through the Flip Server performance, to HTTP through the Flip Server with BoostEdge performance (conditions 2 & 5). We also measure the effects of SPDY by comparing HTTP versus SPDY performance with BoostEdge turned on (conditions 4 & 5). In the first case, we can see BoostEdge improves performance over HTTP by 32%, for the highest 5 websites, and 58% for the lowest 5. Looking at

the effect of also turning on SPDY, we can see that SPDY provides an additional 16.5% boost in performance over HTTP when BoostEdge is also used, for the highest 5 websites, and 11% for the lowest 5 websites.

6. **HTTP with BoostEdge:** Looking at the 6th condition, HTTP with BoostEdge without using the Flip Server, we can see an improvement due to BoostEdge of 48% for the highest 5 websites and 56% for the lowest 5 websites.

A graphical representation of the results for the 5 websites with the highest number of HTTP requests is as shown below:

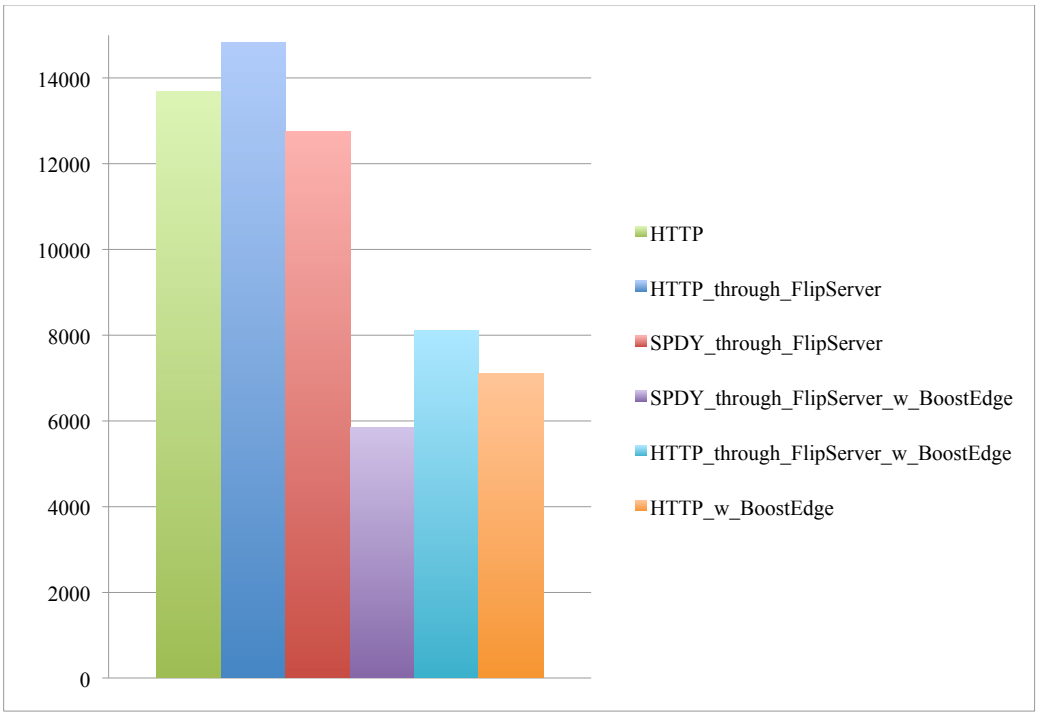


Chart 5: Results for five websites with the highest number of HTTP Requests

A graphical representation of the results for the 5 websites with the lowest number of HTTP requests is as shown below:

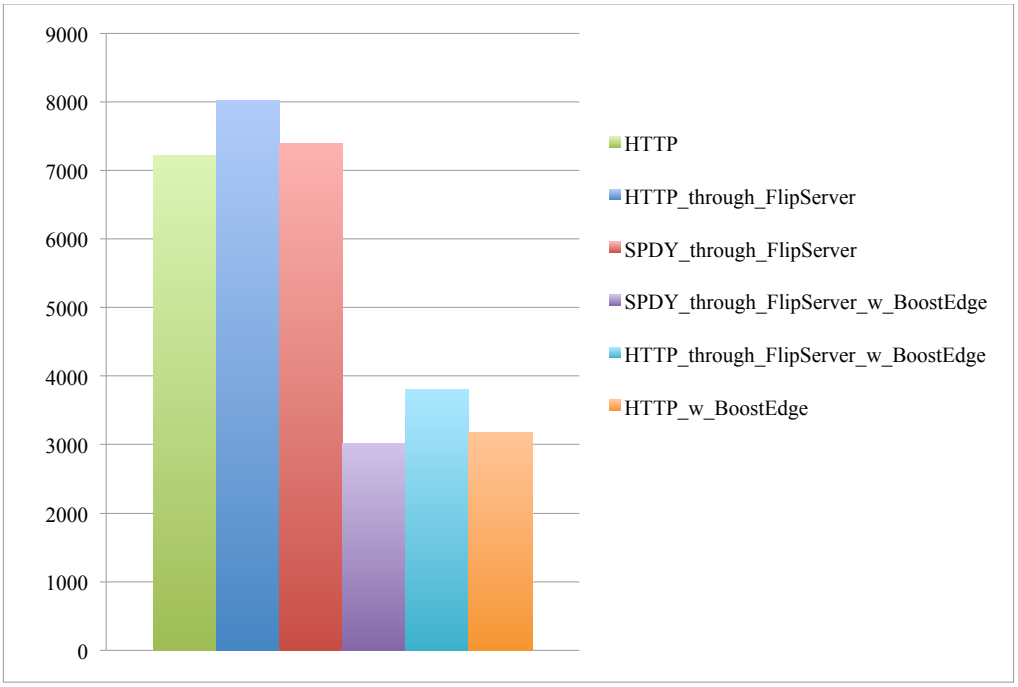


Chart 6: Results for five websites with the lowest number of HTTP Requests

**Result Summary:** The results show that for websites with the highest # of HTTP requests, as well as for the websites with the lowest # of HTTP requests, we continue to gain the largest improvement in performance by using BoostEdge together with SPDY. The data optimization contribution of BoostEdge and the network optimization contribution of SPDY appear to be complementary and additive in effect. Both approaches show significant improvements in performance individually over HTTP, and when combined each seems to provide the most benefit under those conditions that reduce the benefit of the other. In other words, under conditions where network performance is dominating, SPDY tends to provide increased benefits, and under conditions where data optimization is dominating, BoostEdge provides increased benefits.

## VI. Discussion

To address the burgeoning demand for affordable data bandwidth, in this study we investigated two approaches: 1. Content optimization and compression; 2. Optimizing network protocols. In this paper, we utilized content optimization and compression by means of BoostEdge and employed the SPDY network protocol to lower the round trip time for HTTP transactions.

Further, since the data and transport layers are separate, we concluded our investigation by studying the combined effect of these two techniques on web performance. Using document mean load time as the measure, we found that for our testing profile, both with and without packet loss, both BoostEdge and SPDY provided significant improvement in speed over HTTP, while in the case of SPDY also providing SSL connections. When tested in various combinations, we found that the effects are more or less additive, and that the maximum improvement was gained by using BoostEdge and SPDY together. Interestingly, the two approaches are also complimentary. That is, in situations where data predominates (i.e. "heavy" data, and fewer network requests), BoostEdge provides a larger boost via its data optimization capabilities. In cases where the data is relatively small, or "light", but there are many network transactions required, SPDY provides an increased proportion of the overall boost. The general effect is that relative level of improvement remains fairly consistent over various types of websites.

Our absolute measurements are of course dependent on our testing profile, and also the low level of server saturation, as we were generating client requests one at a time. Thus these measurements will change as the parameters change. However, we would expect that the relative ordering of the results in the various test cases should hold up, unless of course the network and servers are so fast and under-loaded that computation costs for optimization consume more time than the resulting performance gain.

**Future work:** In this initial benchmarking, we did not test the case of heavily loaded servers and networks. This case is very interesting, as this represents the situation from the operator side. BoostEdge provides various capabilities for such situations, allowing for features such as tradeoffs between handling larger numbers of simultaneous connections and the level of data compression, providing greater optimization for connections that have the poorest latencies, etc. SPDY is also intended to shine under situations of heavy web traffic. It will be interesting to see how the combination of BoostEdge and SPDY perform under circumstances of heavy web traffic and highly loaded servers. We would expect that the additive effect we have seen in our tests should result in the ability to handle larger numbers of users before saturation, and improve the response times for these users, compared to using either approach on its own. In particular, the complimentary aspect of these two approaches should improve overall effectiveness as both data heavy and transaction heavy connections should benefit.

## VII. Conclusion

In this study, we measured the effects of two software approaches to improving network performance: 1. Content optimization and compression; and 2. Optimizing network protocols. We utilized content optimization and compression by means of BoostEdge and employed the SPDY network protocol to lower the round trip time for HTTP transactions. Since the data and transport layers are separate, we concluded

our investigation by studying the combined effect of these two techniques on web performance. Using document mean load time as the measure, we found that for our testing profile, both with and without packet loss, both BoostEdge and SPDY provided significant improvement in speed over HTTP. When tested in various combinations, we found that the effects are more or less additive, and that the maximum improvement was gained by using BoostEdge and SPDY together. Interestingly, the two approaches were also complimentary. In situations where data predominated (i.e. "heavy" data, and fewer network requests), BoostEdge provided a larger boost via its data optimization capabilities. In cases where the data was relatively small, or "light", but there were many network transactions required, SPDY provided an increased proportion of the overall boost. The general effect is that the relative level of improvement remained fairly consistent over various types of websites.

## VIII. Acknowledgements

We would like to thank the many people who contributed to the success of this work. First, the many students who participated and bore the brunt of the work – Simon Sibomana, Guntawee Tiwapong, Panat Tayaporn, and David Lin-Shung Huang. Several faculty were kind enough to advise us on various problems and decisions – particularly Prof. Collin Jackson and Prof. Patrick Tague. We'd also like to thank the team at ActivNetworks for providing the BoostEdge appliance and spending many hours helping us work through technical problems – Joel Levee, Abdelhafid Meziani and Caroline Paulin. We also received very useful advice and feedback from the Google SPDY team - Jim Roskind and Mike Belshe. Finally, this project could not have happened without the support and encouragement of Christian Martin, of the Institut Telecom's Silicon Valley Office.

## IX. References:

1. "Making Web Apps and Networks Go Faster" http://www.activnetworks.com/BoostEdge/overview/
2. "SPDY: An experimental protocol for a faster web" http://www.chromium.org/spdy/spdy-whitepaper
3. http://www.alexa.com/siteinfo
4. "YSlow user guide" http://developer.yahoo.com/yslow/help/
5. "GNU Wget 1.13.4 User Manual" http://www.gnu.org/software/wget/manual/wget.html
6. "Chromium Benchmarking Extension" http://www.chromium.org/developers/design-documents/extensions/how-the-extension-system-works/chrome-benchmarking-extension

# X. Appendices

1. **Appendix 1: Test profile - 25 websites, 1 Mbps, 100 ms latency, no packet loss.**

2. **Appendix 2: Test profile - 25 websites, 1 Mbps, 100 ms latency, 0.9% - 1.0% packet loss**

## Appendix 1: Test profile - 25 websites, 1 Mbps, 100 ms latency, no packet loss

| | HTTP | HTTP_through_FlipServer | SPDY_through_Flipserver | SPDY_through_FlipServer_w_BoostEdge | HTTP_through_FlipServer_w_BoostEdge | HTTP_w_BoostEdge |
|---|---|---|---|---|---|---|
| **washingtonpost** | 16774.5 | 17808.6 | 16147 | 6453.8 | 8541.1 | 7749.3 |
| **cnn** | 15940 | 16997.5 | 15121.9 | 6143.9 | 8315.5 | 7243.5 |
| **mapquest** | 13702.8 | 14378.3 | 12893.6 | 5602 | 7408.9 | 6731.3 |
| **Answers** | 10729.3 | 11588.3 | 10767.5 | 4427 | 5254.3 | 4467 |
| **ehow** | 14148.5 | 14992.1 | 14283.5 | 4668.8 | 5379.8 | 4888 |
| **foxnews** | 1853.5 | 2305.8 | 2155.6 | 1616.4 | 1644.6 | 1468.6 |
| **yahoo** | 10835.4 | 11299 | 10270.8 | 3903.3 | 5076.3 | 4445 |
| **salesforce** | 9667.6 | 8464.9 | 8085.1 | 4345.6 | 5258.3 | 4342.4 |
| **adobe** | 8760.3 | 10377.1 | 8596.3 | 3581.1 | 5679.9 | 4100.9 |
| **godaddy** | 8137.1 | 8868.6 | 8351.1 | 3732.4 | 4720.9 | 3953.1 |
| **bbc** | 9293.6 | 10856.1 | 9460.1 | 4704.8 | 6510.5 | 5167.5 |
| **yelp** | 9035.8 | 9563.3 | 8793.9 | 4340.6 | 5458.6 | 4774.3 |
| **careerbuilder** | 7215.4 | 8771.3 | 7878.8 | 2918.3 | 4281.1 | 3023.8 |
| **aol** | 6313 | 7258.3 | 6317.8 | 2835.8 | 3661.9 | 3114.4 |
| **bestbuy** | 8364.4 | 9775.8 | 7891.9 | 4024.3 | 5912.6 | 4813.1 |
| **ebay** | 5679.5 | 6252.6 | 5796 | 3189.9 | 3814 | 3195.5 |
| **comcast** | 6553 | 7936.8 | 6558.9 | 2877.6 | 4163.3 | 3159.9 |
| **wordpress** | 4412.4 | 6286 | 4462.8 | 2815 | 3942.9 | 2992.1 |
| **usps** | 5788.5 | 6802.9 | 5288.5 | 3159.6 | 4909.8 | 4019.3 |
| **foxsport** | 5613.8 | 7371.4 | 5629 | 3220 | 3142.8 | 2996.3 |
| **typepad** | 4318 | 5321.5 | 4408.5 | 2733.3 | 3775.1 | 3068.4 |
| **cj** | 7862.8 | 8806.8 | 6625.1 | 4801.1 | 7185.8 | 6369.4 |
| **huffingtonpost** | 3717.4 | 3820.9 | 3951.3 | 2032.6 | 2130.4 | 1702.6 |
| **att** | 6586.1 | 7442.9 | 6272 | 2505.4 | 4057.3 | 2887.5 |
| **latimes** | 21806.3 | 23976.5 | 20894 | 8498.3 | 11438.9 | 9966.8 |

**Table 1: Table of Mean Doc load Time for Test profile - 25 websites, 1 Mbps, 100 ms latency, no packet loss**

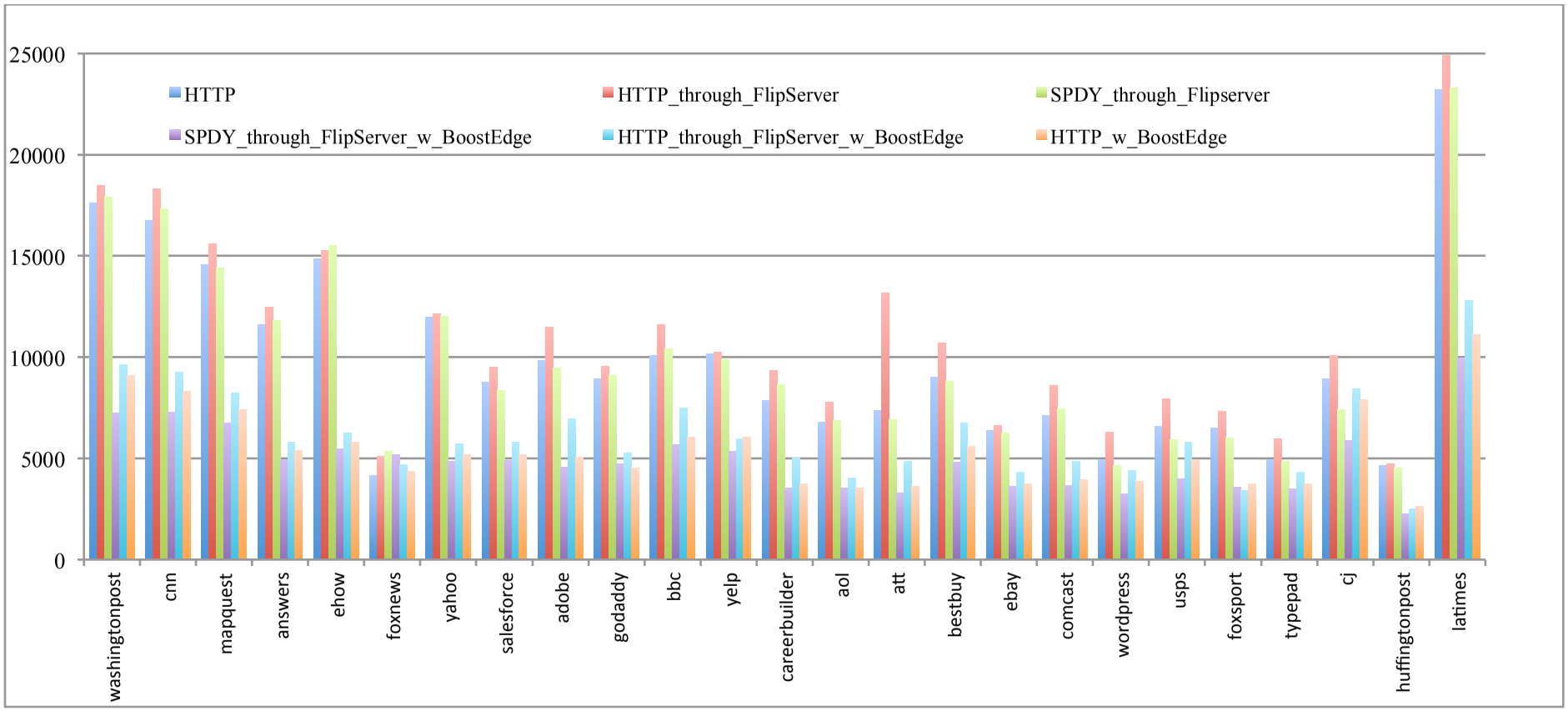


**Chart 1: Plot of Mean Doc load Time for Test profile - 25 websites, 1 Mbps, 100 ms latency, no packet loss**

## Appendix 2: Test profile - 25 websites, 1 Mbps, 100 ms latency, 0.9% - 1.0% packet loss

| | HTTP | HTTP_through _FlipServer | SPDY_through _Flipserver | SPDY_through _FlipServer_w_ BoostEdge | HTTP_through _FlipServer_w_ BoostEdge | HTTP_w_ BoostEdge |
|---|---|---|---|---|---|---|
| **washingtonpost** | 16961.1 | 18123.3 | 17401.3 | 7004.4 | 8613.5 | 8264.3 |
| **cnn** | 16131.3 | 17877.8 | 16420 | 6634 | 8628.1 | 7746.1 |
| **mapquest** | 13674.9 | 15952 | 13786.8 | 6049.1 | 7852.3 | 6625.1 |
| **Answers** | 11097.6 | 12236 | 11144.5 | 4694.4 | 5344.3 | 5059.8 |
| **ehow** | 14516.5 | 15745.6 | 15535 | 5073.3 | 5733.3 | 5111.3 |
| **foxnews** | 1955.9 | 3229.8 | 2259.4 | 1659.5 | 1644 | 1544 |
| **yahoo** | 11084.6 | 11916 | 11579.3 | 4499.5 | 5104.4 | 4976.3 |
| **salesforce** | 8808.6 | 9628.3 | 8773.9 | 4492.3 | 5390.8 | 4670 |
| **adobe** | 9178.6 | 10889.6 | 9233.4 | 3700.4 | 5942.3 | 4338.5 |
| **godaddy** | 8147.4 | 9523.1 | 8791.1 | 3962.9 | 4814.8 | 3993.1 |
| **bbc** | 9509.3 | 10960.8 | 10181 | 5040.5 | 6680 | 5292.8 |
| **yelp** | 9931.4 | 10811.4 | 9627.9 | 4923.1 | 5617.8 | 5400 |
| **careerbuilder** | 7236 | 8903.6 | 8413.6 | 3285.6 | 4592.9 | 3374 |
| **aol** | 6731.4 | 7969.6 | 6831.6 | 2913 | 3851.8 | 3373.6 |
| **att** | 6753 | 10562 | 8401 | 4311.1 | 6205.4 | 3015 |
| **bestbuy** | 8336.4 | 7756.1 | 6550.4 | 2683.3 | 4102.3 | 4956.5 |
| **ebay** | 5834.8 | 6830.9 | 6101 | 3215 | 3888.1 | 3302.5 |
| **comcast** | 6879.6 | 8424.6 | 6830.5 | 2986.3 | 4593.8 | 3406.8 |
| **wordpress** | 4545.5 | 5778.8 | 4564.3 | 2882.3 | 3871.1 | 3157 |
| **usps** | 5989.3 | 7181.3 | 5468.4 | 3363.8 | 4987.1 | 4311.5 |
| **foxsport** | 6212.6 | 7511.4 | 6634.3 | 3347.8 | 3269 | 3206 |
| **typepad** | 4683.1 | 6008.6 | 4837.9 | 2973.1 | 3746.9 | 3103.9 |
| **cj** | 8426.5 | 9439.9 | 7592 | 5482.8 | 7518.9 | 6564.6 |
| **huffingtonpost** | 4465.4 | 6019.6 | 4275.8 | 2206.8 | 2238.3 | 1983.3 |
| **latimes** | 22403.4 | 24110.1 | 23872.4 | 9497.5 | 11847.1 | 10152.3 |

**Table 2: Table of Mean Doc load Time for Test profile - 25 websites, 1 Mbps, 100 ms latency, 0.9%-1.0% packet loss**

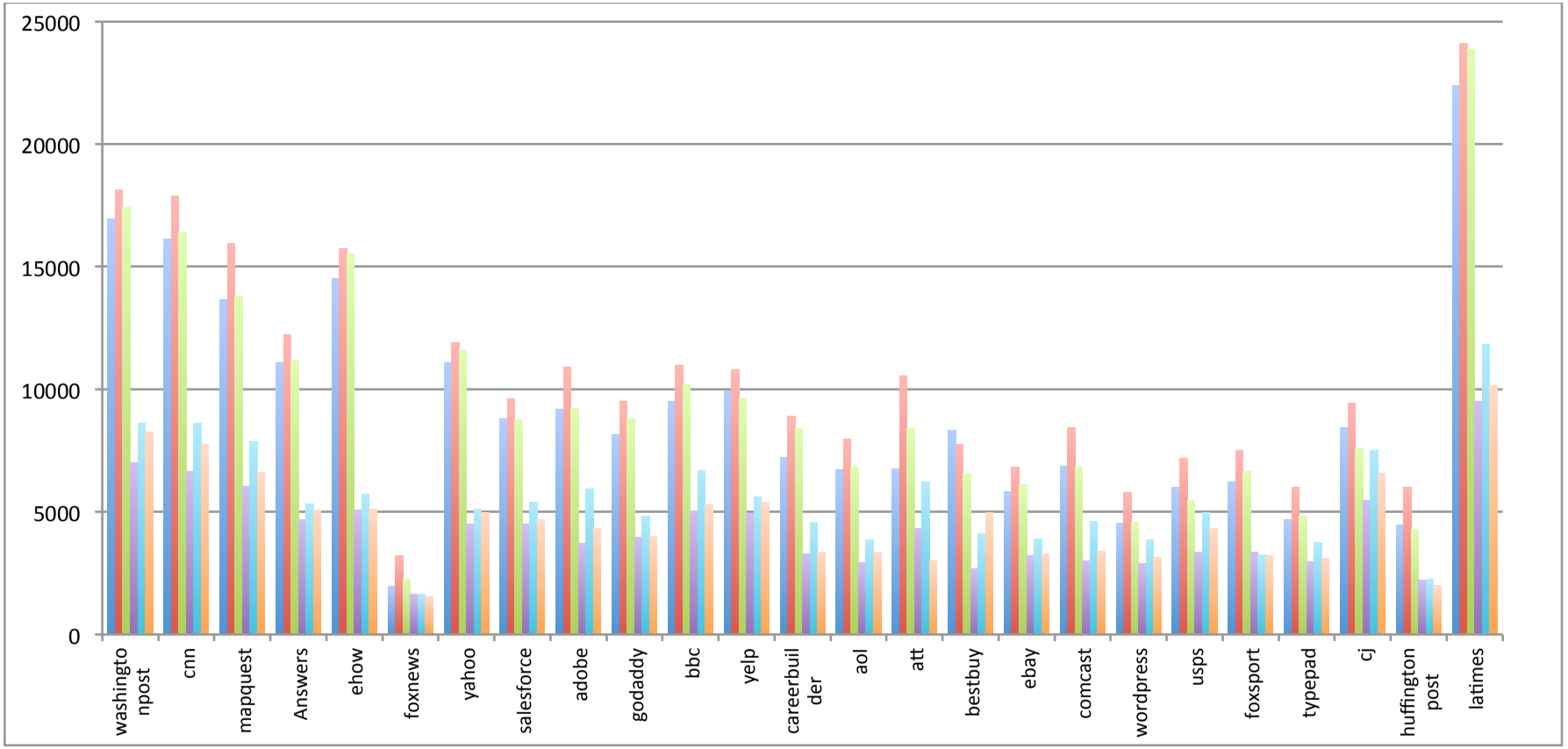


**Chart 2: Plot of Mean Doc load Time for Test profile - 25 websites, 1 Mbps, 100 ms latency, 0.9%-1.0% packet loss**