Assignment 4: Webserver Latency Analysis

Stephen Chambers

April 19, 2015

1 The Question

The goal of this assignment is explore latency of a simple request response interaction implemented using a variety of methods. The system consists of a client requesting a simple piece of information from a server, the server responds with the information, the client receives it, and measures, as precisely as possible, the time the entire transaction took.

This paper specifically will look at how latency varies between two different application layer protocols designed to serve as webservers. These two protocols are HTTP and Node.js. Additionally, encryption is used as an additional variable for both protocols to determine whether or not it has a significant impact on latency.

2 Introduction

The language chosen for this assignment is Python. The timing methods being used is the *time* module. This module was chosen for its timing precision. It has a confidence interval of +-0.0762ms when measuring how long a sleep operation of 100ms took. Since downloading a simple web page should take on the order of milliseconds, this timing module should be sufficiently accurate for this study.

Latency is hypothesized to increase by approximately 150% with security added. Encryption requires additional bytes transferred to maintain security, and the encryption handshake adds additional latency as well. Furthermore, Node.js is hypothesized to have a latency of approximate equivalence to Apache. The overhead for both protocols should not be significant enough to impact latency on such a small load.

3 Experiment

The python library *requests* is used for all server requests in order to maintain empiricism. Using a browser may introduce additional variables into the experiment and is not desirable. The page being downloaded is a *Hello World* text file. It only thirteen bytes long in order to avoid the size of the file having an impact on the experiments. The self signed cert used for encryption was made using the *openssl* library.

4 Results

Table 1 below shows a matrix of the mean latency in milliseconds of downloading a small text file with Apache, Node.js, and the absence or presence of encryption. Each mean latency value has four decimal places. As stated above, the confidence interval for each measurement is +-0.0762ms.

	Apache	Node.js
Encryption	11.1100ms	47.1209ms
No Encryption	1.7977ms	1.8725ms

Table 1: Result matrix of Apache, Node.js, encryption, and no encryption

5 Discussion

Encryption increased latency far more than originally anticipated. Additionally, Node.js was affected drastically more than Apache with the presence of encryption. One potential hypothesis of why this occurred was that Node.js did not cache the encryption key and certificate when the first request was sent, while Apache did cache the information. However, more experiments would need to be run to determine the actual cause.

6 Conclusion

Both Apache and Node.js behaved similarly with no encryption. However, the presence of encryption increased the latency of both application protocols significantly. Additionally, Node.js was affected significantly more than Apache was. In conclusion, while Apache and Node.js had similar latency performance with no encryption, the hypothesis was ultimately proved incorrect as encryption affected the latency of both protocols at a higher impact than anticipated.