Latency analysis of a request response interaction

Varsha Thirumakil
Department of Computer Science
Assignment-4
vt2000@wildcats.unh.edu

Abstract*— In a computer network, latency is an expression of how much time it takes for a packet of data to get from one designated point to another. This can be represented in a request-response interaction as the time required for a network communication to travel from the source to the destination and back to the source. In this paper, we analyze the expected performance considering the protocol header overhead, number of packets sent and propose a framework to evaluate the findings of the simulation. This has been acheived by carefully conducting an experiment, performing statistical analysis and quantitative comparisons of the timings.

***Keywords*— latency, routing, RTT**

I. Introduction

Today web performance circles has a simple goal that is to make the internet faster. Tackling latency is a top priority for the performance industry. Clients and servers exchange messages in a request-response message pattern. This analysis is made based on four protocol combinations ipv4 and http, ipv4 and https, ipv6 and http, ipv6 and https using **node.js.**

II. Speculative Idea

Internet Protocol is a set of technical rules that defines how computers communicate over a network. There are currently two versions: IP version 4 and IP version 6 that are to be analyzed practically and theoretically. The technical functioning of the Internet remains the same with both versions but the major difference between IPv4 and IPv6 is the number of IP addresses and in the header size.

From the above, I believe that there may be delay and overhead in the ipv6 compared to ipv4 because of the larger header size and the address size but practically experiments negate these views. Hence several data sets are analyzed and conclusions are drawn which contradicts the theoretical results.

III. Description Of Experiment

The implementation is based on the

A. Simulator

In this paper, implementation of a request-reponse simulation is based on the Node.js framework. Node.js is more commonly known for creating highly scalable server applications. [2] The Node.js execution model is event-driven, and emphasizes asynchronous programming in order to maximize scalability.

B. Experiment Setup

Given a server and a client the basic experiment works by client requesting few tens of bytes of data from the listening server. The server responds back to the client with the data using the callback function.

C. Measures

The **round trip time or the latency** of the request-response interaction is calculated in the experiment. To be more confident with the results we also explore the measures like overhead of protocol, header size, number of packets sent.

IV. Description of the Experiment

In the network layer we use the **Ipv4 and Ipv6** protocols to experiment. The underlying protocol used is **TCP** in the transport layer.

Http server is created using the http module of Node.js. The callback function begins by calling the response.writeHead() method. This method sends an http status code and a collection of response headers back to the client. Server returns the code 200, which indicates success.
Https server makes a request to a secure web server. To create an **HTTPS server** self-signed certificate is utilized along with the Node's built-in https module. To establish the **SSL** connection the certificate and a private key has been created. Client thus receives the response from the server.Once the response is received the round trip time is calculated using the timing methods.
RTT(Round Trip Time) is calculated as the difference between the start time when the request has made and the end time when the reponse has been produced. The round trip times are hence tabulated and the the findings are analyzed. To explore the latency effectively among the protocols client-server interactions are perfomred for different sets of information.

From Table 1 and Table 2 we can conclude that https has very high latency compared to http. This is not completely due to the encryption that goes on in the https server. HTTPS requires an initial handshake which can be very slow. The actual amount of data transferred as part of the handshake isn't huge but for very small requests, this can be quite a bit of overhead. It is also analyzed that making lots of short requests over HTTPS will be quite a bit slower than HTTP, but if you transfer a lot of data in a single request, the difference will be insignificant.

Analyzing the findings in-terms of ipv4 and ipv6 we can derive that ipv6 has a better performance in comparison to ipv4. Though in some cases both have similar latencies, ipv6 is efficient in many ways like IPv6 header structure more flexible to Ipv4. Ipv6 provides confidentiality, authentication and data integrity whereas IPv4 ICMP packets are often blocked by corporate firewalls. [1] Thus more efficient routing, packet processing, simplified network configuration, security makes Ipv6 efficient protocol compared to Ipv4.

Table 1: Tabulated Results of HTTP (ipv4 vs ipv6)

Http + ipv4(ms)	Http + ipv6(ms)
17	15
15	14
14	13
13	13

Table 1: Tabulated Results of HTTPS(ipv4 vs ipv6)

Https + ipv4(ms)	Https + ipv6(ms)
72	71
78	77
76	72
70	70

For instance if latency of https is 70ms and http is 17ms. It can be seen latency of https is approximately four times more than http. At the start of the experiment its noted that latencies show difference in performance but over time we can observe the sorted latencies in either protocols. Average standard deviation is observed in both the set of protocols.

CONCLUSION

Thus a simple request response interaction was implemented analyzing different simulation times, it can be concluded that the underlying protocol plays an important role in determining the performance of a transaction.

REFERENCES

[1]http://www.networkcomputing.com/networking/six-benefits-of-ipv6/d/d-id/1232791?

[2] http://www.sitepoint.com/an-introduction-to-node-js/