Report 1: Rudy - A rudimentary web server

Bernardo González Riede

September 12, 2017

1 Introduction

As mentioned in the assignment paper, the purpose of this assignment is to learn:

- *the procedures for using a socket API*
- *the structure of a server process*
- *the HTTP protocol*

The knowledge which is acquired through these concepts mark the fundaments of communication through a network.
Sockets are the base for all communications towards external processes. Sometimes they are even used for communicaion between processes residing on the same machine.
HTTP is a widely used protocol for web servers.

2 Main problems and solutions

One problem encountered was how to simulate different scenarios and participants. Since the exercise is only a small web server, all test could have been realized on the same machine (running Windows 10). Still, the thirst for performing real world test with physical networks and GNU/Linux servers was far too attractive. Therefore the test involved three participants:

Name	CPU	RAM	Location	OS
clientLocal	4-core	4GB	KTH Kista	Windows 10
server	1-core	512MB	Frankfurt	Ubuntu-server 16.04
clientRemote	1-core	512MB	Frankfurt	Ubuntu-server 16.04

Table 1: Information about the participants

The two Ubuntu machines were hosted on Digital Ocean's Datacenter in Frankfurt, Germany. It's a virtualized datacenter, hence the incomplete description about their hardware.

3 Evaluation

5 scenarios were used to test the web server.

- A: Rudy & test executed on clientLocal.
- B: Rudy & test executed on server.
- C: Rudy on server & test on clientLocal.
- D: Rudy on server & test on clientRemote.
- E: Rudy on server while test was executed on both clients simulatenously.

The following table results from doing 1000 request to the server. An artificial delay of 40ms was introduced in a second run to simulate file handling.

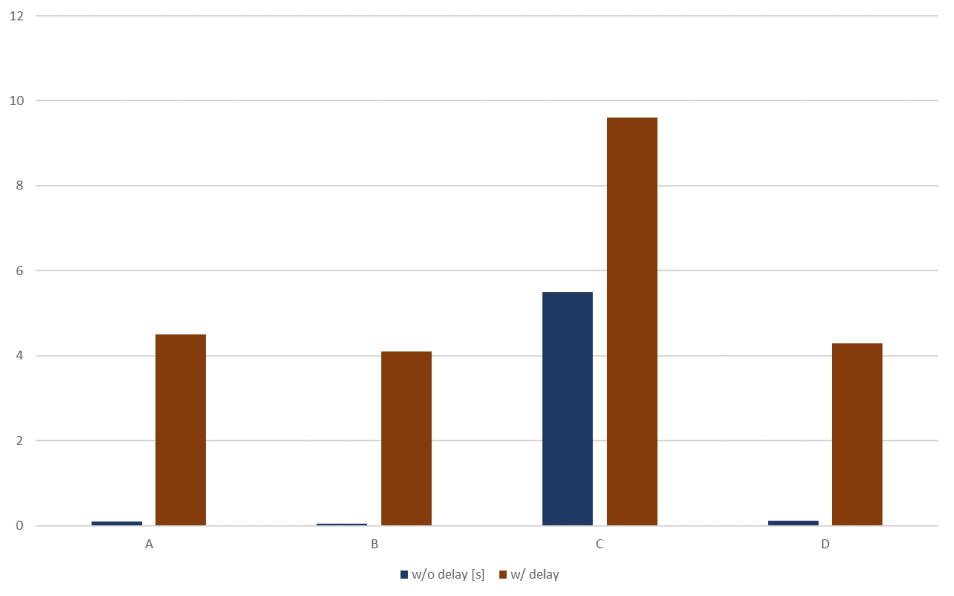

Figure 1: Comparision of scenarios and artificial delay introduced.

This shows a clear impact of the artificial delay. Scenario E is special and has to be interpreted. In the case without an artificial delay, it's impact is $<1\%$, but sometimes more noticeable. Contrary to this, running the server with an artificial delay between request elevates the time it takes to complete by 50%.

4 Conclusions

The problem gave some good ideas about how to implement a server to respond via HTTP instead of the normal messagin of Erlang. This gives a more universal approach since sockets and HTTP are in many, if not all progamming languages, present.