Programming Algorithms of load balancing with HA-Proxy in HTTP services

Programación de algoritmos de balanceo de carga con HA-Proxy de servicios HTTP

José Teodoro Mejía Viteri[1,*], María Isabel Gonzáles Valero[1,†], and Ángel Rafael España León[1,‡].
[1]Universidad Técnica de Babahoyo, Ecuador.
dyhack@hotmail.com;mery_2523@hotmail.com;arel_setnet@hotmail.com

Received: August 15, 2017 — **Accepted:** September 15, 2017

How to cite: Mejía Viteri, J., Gonzáles Valero, M. I., & España León, Ángel R. (2018). Programming Algorithms of load balancing with HA-Proxy in HTTP services. *Journal of Science and Research: Revista Ciencia e Investigación, 3(CITT2017), 100-105.* https://doi.org/10.26910/issn.2528-8083vol3issCITT2017.2018pp100-105

***Abstract*—The access to the public and private services through the web gains daily protagonism, and sometimes they must support amounts of requests that a team can not process, so there are solutions that use algorithms that allow to distribute the load of requests of a web application in several equipment; the objective of this work is to perform an analysis of load balancing scheduling algorithms through the HA-Proxy tool, and deliver an instrument that identifies the load distribution algorithm to be used and the technological infrastructure, to largely cover implementation. The information used for this work is based on a bibliographic analysis, field study and implementation of the different load balancing algorithms in equipment, where the distribution and its performance will be analyzed. The incorporation of this technology to the management of services on the web, mproves availability, helps business continuity and through the different forms of distribution of the requests of the algorithms that can be implemented in HA-Proxy to provide those responsible for information technology systems with a view of their advantages and disadvantages.**

***Keywords*—Technology Communications, Availability of Information, Data Security.**

***Resumen*—El acceso a los servicios públicos y privados a través de la web gana protagonismo diario, y a veces debe soportar cantidades de solicitudes que un equipo no puede procesar, por lo que existen soluciones que utilizan algoritmos que permiten para distribuir la carga de las peticiones de una web aplicación en varios equipos; el objetivo de este trabajo es realizar un análisis de algoritmos de programación a través de la herramienta HA-Proxy de balanceo de carga y entregar un instrumento que identifica el algoritmo de distribución de carga a utilizar y la infraestructura tecnológica, para cubrir en gran parte puesta en práctica. La información utilizada para este trabajo se basa en un análisis bibliográfico, estudio de campo y aplicación de los algoritmos en el equipo, donde se analizará la distribución y el rendimiento de equilibrio de carga diferentes. La incorporación de esta tecnología a la gestión de servicios en la web, mproves la disponibilidad, continuidad del negocio ayuda a y a través de las diferentes formas de distribución de las solicitudes de los algoritmos que pueden ser implementadas en HA-Proxy para proporcionarlos responsable de sistemas informáticos con el fin de sus ventajas y desventajas.**

***Palabras Clave*—Tecnología de comunicaciones, La disponibilidad de información, Seguridad de datos.**

INTRODUCTION

The internet in the beginning allowed to visualize contents of static pages and the client acceded to review information on some subject of interes. but as time went by companies began to provide email services, chat, besides giving a communication option also incorporated messages that served as marketing to encourage users to buy some type of article or service, but today the internet has become an accessible and mobile network, because the voice, data and video traveling through different networks is now performed through this large network called cloud so described by its complexity of interconnections,implemented at the global level that allows to carry out formalities in governmental institutions,bank, and at the same time buy and sell through the websites of companies.

Nowadays, public institutions and private companies offer trading processes in their websites and archieved with other services such as: data bases, streaming, e-mail, VoIP and applications created with programming languages that access and save information, and allows the accessible from a computer or mobile.

All this is possible through the web server ”which is the heart of the web and defines how web clients request objects to a server and as servers transfer them to clients”, (Kurose and Ross, 2009), that is to say it is in charge of managing the requests of services that are requested to the pages that provide services of the companies, thanks to this it increases the profitability but also the requests of the clients and often causes the page is not available or the delivery of information to customers is slow, causing discomfort and sometimes lost customers and money for poor service satisfaction,thanks to this it increases the profitability but also the requests of the clients often causes the page is not available or the delivery of information to customers is slow causing discomfort and sometimes lost customers and money for poor service satisfaction this is why availability plays an important role, so that the services of the company are active for the client with a minimum response time.

This availability that is the characteristic of a system to remain active before contingency, this implies higher costs because the implementation of high availability is directly

[*]Magíster en Informática Empresarial
[†]Magíster en Informática Empresarial
[‡]Magíster en Informática Empresarial

proportional to the cost in technological infrastructure, it is for this reason that the research aims to implement and analyze the results of the techniques that allow to manage the load that a web server receives.

And the following topics are considered successively: Web Servers, protocol HTTP (Hypertext, Transfer; Protocol), High availability, the different algorithms that exist and the tools for their implementation and analyze the response time results in the requests to the web server.

WEB SERVER, HA-PROXY AND LOAD BALANCING ALGORITHMS

Web Server

A Web server is an application that allows you to answer and respond to requests originating from browsers, providing resources requested through the protocol http or securely through the protocol HTTPS.

The hypertext transfer protocol – HTTP is a protocol of the layer of the TCP/IP (Transmission Control Protocol/Internet Protocol) (Kurose and Ross, 2009).

All companies use internal processes to manage their information flows and external business processes in order to interact with their partners, customers and employees in the value chain. Both processes can be managed with Web services to reduce costs and boost business agility (Cáceres Alvarez et al., 2011).

Intern Web Services: belong to the companies hat uses them and can serve to connect different companies departments, such as Sales, Human Resources, Finance and Production. For example, a financial application can call in real time a web service that converts euros into dollars or the Sales Department can call another to query certain information from the customer database (Cáceres Alvarez et al., 2011).

External Web Services: allow companies to exchange services over the Internet. A company interested in a particular service can go to a directory of Web services like UDDI (Universal Description, Discovery and Integration) to look for it. An example of these public records is www.xmethods.net where you can find hundreds of free web services (Cáceres Alvarez et al., 2011).

Apache

Apache is a free web server developed by Apache Server Project. Its main objective is the creation of reliable, efficient and easily extendable web server with free open source (Márquez Díaz et al., 2011).

What is HTTP?

HTTP (Hypertext Transfer Protocol) works at the application level in the OSI model for hypermedia distributed and cooperative information systems. Its main characteristic is that it allows negotiating the type of data and its representation, leaving the possibility to build systems around it independently of the transferred data (de la Mora, 1997).

GET and POST methods

The GET method asks for a specified resource representation. For security, it should not be used by applications that cause effects, since it transmits information through the URL, adding parameters to it. The POST method sends data to the server that can be part of a database, messages to a group of users, etc. Unlike the GET method, it can send any amount of data without limit and it does not send information as part of the URL. The data is included in the body of the request (Ríos Pérez et al., 2017).

High Availability

Currently, there are several implementations of high availability clusters and all of them seek to offer excellent levels of availability and efficiency of the service, always safeguarding the information security and emphasizing load balancing processes, storage and maintaining the reliability of the system through the Redundancy as main technique (Bowen and Coar, 2007).

Load Balancing Algorithms

Load Balancing Algorithms are responsible for distributing the load among the available nodes to distribute requests, decisions that are made based on different methodologies. The methods to be studied are:

Round Robin

It is one of the best known, in this case the balancer receives requests from clients regardless servers characteristics or the load that can support the first request that sends the first node configured with the web server and then to the second one until distributes one request to each available server.

Static Round Robin

It is a variant of Round Robin, but with the additional characteristic of assign o request percentage to each available web server for balancing load.

Least Connection

This algorithm appears to solve the count of requests for each one of the web servers to control the amount of open connections in the web servers.

HA-Proxy

It's a free solution, very quickly and reliable that offers high availability and allows commutation by error and load balancing and proxy for TCP and HTTP based on applications. When HAProxy is running in HTTP mode, both the request and the response are completely analyzed and indexed, so it is possible to construct matching criteria in almost any of the contents.

Apache-jmeter

The Apache JMeter application is open source software, a 100 % pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to test performance on both static and dynamic resources.

METHODOLOGY

Bibliographic and Documentary Analysis

Collection of information was carried out and are represented in a unified systematic way to facilitate it compression, through analysis and synthesis, through indexing, annotation.

Programming Algorithms Implementation

This research is developed by the virtualization of 3 computers, one is used as load balancer, and two web servers to receive requests.

Figure 1. Test Equipment Layout.
Source: Prepared by the authors.

In each computer was realized the installation of Centos 7 virtual Operative System, through the Virtual Box 5.1.26 tool with the following characteristics:

Table 1. Characteristics of the computers used for the implementation

Processor	Intel® Core i7 CPU 2.60GHZ
RAM	1 GB
Swap	1.5 GB
Hard Drive	15 GB

Source: Prepared by the authors.

The installation of the Apache 2.2.4.6 server in two virtual computers that receive the requests from the load balancer called Server1, Server2.

The install of an HA-Proxy tool to manage requests from clients, allows to implement algorithms in the load balancer.

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#
#   http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
#
#---------------------------------------------------------------------

#---------------------------------------------------------------------
# Global settings
```

```
#---------------------------------------------------------------------
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    #   2)   configure   local2   events   to   go   to   the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats
```

```
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000
```

```
#---------------------------------------------------------------------
#new configuration for statistics
listen stats :9000
mode http
stats enable
stats uri /haproxy
stats realm HAProxy\ Statistics
stats auth haproxy:password
stats admin if TRUE
#---------------------------------------------------------------------
#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend  main *:5000
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app


#---------------------------------------------------------------------
# static backend for serving up images, stylesheets and such
#---------------------------------------------------------------------
backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check
```

```
#---------------------------------------------------------------------
-------
# round robin balancing between the various backends

backend app
balance     roundrobin
#server  app1 127.0.0.1:5001 check
#server  app2 127.0.0.1:5002 check
#server  app3 127.0.0.1:5003 check
#server  app4 127.0.0.1:5004 check

#new configuration
#——
frontend webapp
    bind *:80
    default_backend webserver
#
backend webserver
    balance roundrobin
    server server1 192.168.15.110:80 check
    server server2 192.168.15.111:80 check
```

Figure 2. Ha-proxy configuration file 1. Load Balancer Configuration.

Source: Prepared by the authors.

The install of Apache-jmeter 3.2 tool allows to make requests to our load balancer y this will distribute according to the implemented algorithm.

The Apache-jmeter app make the configuration of the threads that is the number of clients that make requests to the load balancer and are processed by server 1, server 2. The main purpose is watching the programming algorithm development with HA-Proxy tool, also, with the top command we locate the number of the process assigned to the HA-Proxy tool and see how it behaves the processor and memory consumption, these are the essential parameters to see the load that is receiving, in this case, with HA-Proxy server.

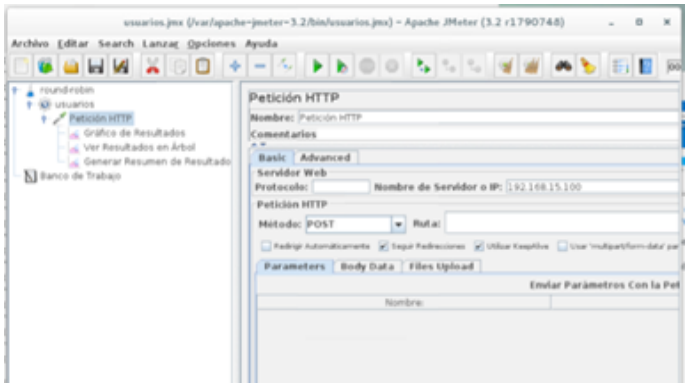

Figure 3. HTTP in Apache-Jmeter Requests Configuration.

Source: Prepared by the authors.

Number of users configuration from http requests to the load balancer.

Figure 4. Thread Group Configuration.

Source: Prepared by the authors.

In the following figure, you can see the IP configuration of our balancer, it receives the users request and use the POST method because it is used in the development of dynamic web sites to send data to the server and can be processed, this means to make an update action or data entry, because, the GET method only performs information requests. The number of threads for the test is 10000.

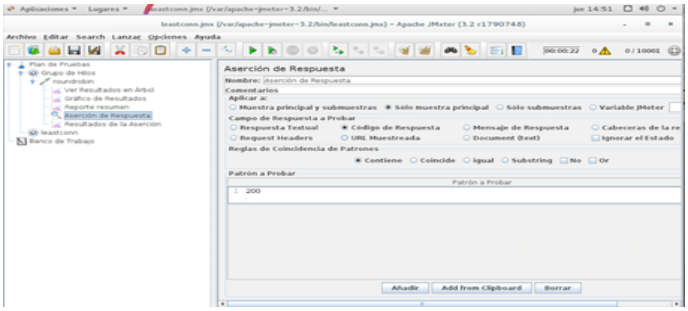

Figure 5. Configuring assertions.

Source: Prepared by the authors.

Finally, the insertion is performed to show the results of the “Response Assertion”, Response Assertion”, in order to verify that all requests to the load balancer have been successfully performed, for this we add the string pattern ”200”, which means that the web page has been served correctly in the following graph shows the configuration options.

RESULTS

Implementation of Programming Algorithm

```
frontend webapp
    bind *:80
    default_backend webserver
#nuevaconfig2
backend webserver
    balance roundrobin
    server server1 192.168.15.110:80 check
    server server2 192.168.15.111:80 check
```

Figure 6. Implementation of Round Robin Programming Algorithm con HA-Proxy.

Source: Prepared by the authors.

In the figure below, you can see the implementation in the backed web server section of the round robin as the programming algorithm requests.

Processor Consumption and Round Robin Memory.

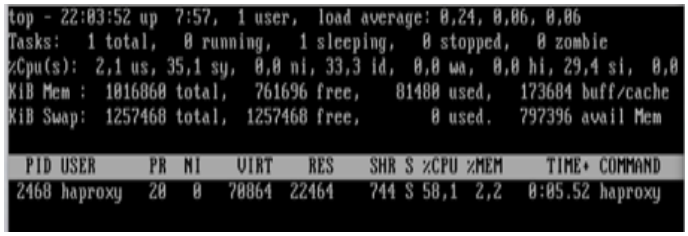

Figure 7. Top command in the Load Balancer.
Source: Prepared by the authors.

In this picture, we see the amount of percentage using of the processor is 58.1 % and 2.2 & Memory.

Etiqueta	# Muestras	Media	Min	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec
Petición HTTP	10000	33	3	262	29,95	0,00%	987,7/sec	371,33
Total	10000	33	3	262	29,95	0,00%	987,7/sec	371,33

Figure 8. Result of Apache-JMeter with Round Robin Algorithm Implementation.
Source: Prepared by the authors.

As seen in the table, the average time taken by a request is 33 seconds, the maximum time invested in a request is 262 seconds and the minimum is 3 seconds, and the number of requests processed per second w ere 987.7 per second, the amount of data processed is 371.33 Kilobits per second.

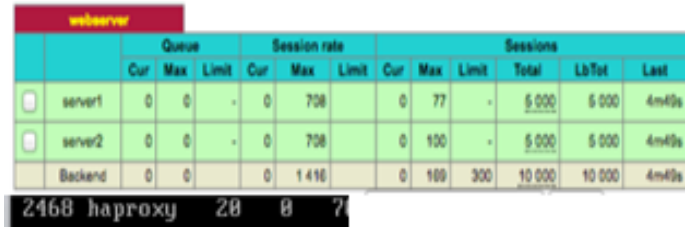

Figure 9. Result of a Number of Petitions per server by an HA-Proxy Tool.
Source: Prepared by the authors.

The following picture shows that 10000 requests are distributed equally in both servers.

Static Round-Robin Algorithm Scheduling

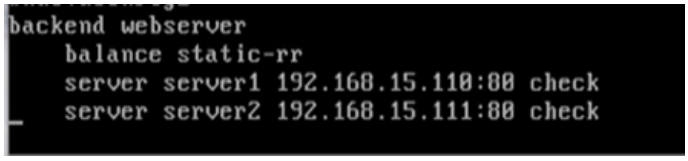

Figure 10. Static Round Robin with Ha-Proxy Configuration.
Source: Prepared by the authors.

In the graphic below we can see in the backed webserver of the static round-robin scheduling as a programming algorithm in the requests and has a 50 % of load for each web server, because these are not with the same characteristics and can configure with different loading in each server in percentage number of load.

Static Round-Robin Processor and Memory Consumption.

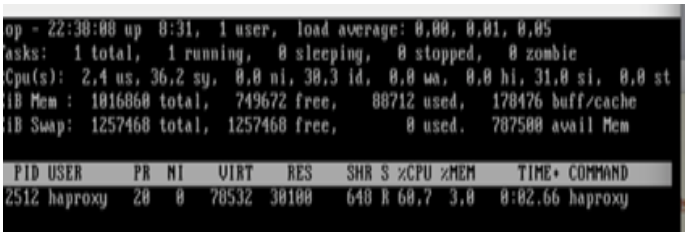

Figure 11. Top Command in the Load Balancer.
Source: Prepared by the authors.

In the following graphic, we can see the amount of processor using percentage (60.7 %) and memory (3.0 %).

Etiqueta	# Muestras	Media	Min	Máx	Desv. Está...	% Error	Rendimiento	Kb/sec
Petición HT...	10000	169	3	1471	242,63	0,00%	889,8/sec	334,56
Total	10000	169	3	1471	242,63	0,00%	889,8/sec	334,56

Figure 12. Result of Apache-JMeter with Static Round-Robin Programming Algorithm Scheduling As we can see in the table below, the average time taken by the request is 169 seconds, which take times to process 10000 request to the server, the maximum time invested by a request is 1471 seconds and the minimum is 3 seconds, the number of processed requests per second were 889.8 per second, the amount of processed data are 334.56 Kb/s.
Source: Prepared by the authors.

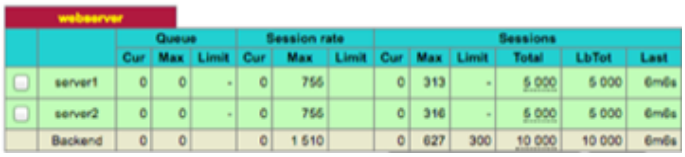

Figure 13. HA-Proxy tool Number of requests per server.
Source: Prepared by the authors.

In the graphic below we can see that 10000 requests are distributed equally in both servers.

Least Connection Algorithm

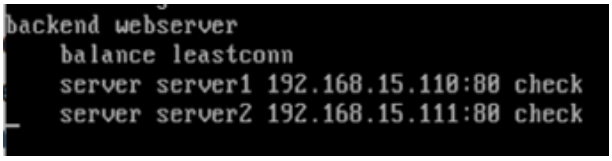

Figure 14. Least Connection Configuration with HA-Proxy.
Source: Prepared by the authors.

In the picture below, we see the backed webserver section setting of the least connection memory as programming algorithm in the requests.

Least Connection processor and memory consumption.

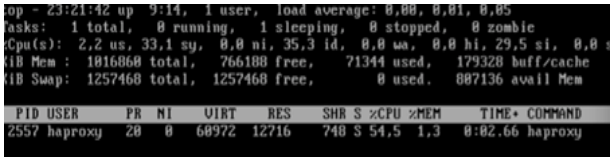

Figure 15. Top Command in the Load Balancer.
Source: Prepared by the authors.

Below, we can see the amount of percentage of processor using (54.5 %) and memory (1.3 %).

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Está...	% Error	Rendimiento	Kb/sec
Petición HT...	10000	49	3	1726	112,94	0,00%	693,9/sec	260,88
Total	10000	49	3	1726	112,94	0,00%	693,9/sec	260,88

Figure 16. Results of Apache-JMeter with Least Connection Algorithm Setting.
Source: Prepared by the authors.

As we can see in the table below, the average time taken by the request is 49 seconds, which take times to process 10000 request to the server, the maximum time invested by a request is 1726 seconds and the minimum is 3 seconds, the number of processed requests per second were 693.9 per second, the amount of processed data are 260.88 Kb/s.

webserver

		Queue			Session rate			Sessions				
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot
☐	server1	0	0	-	0	835		0	184	-	5 237	5 237
☐	server2	0	0	-	0	779		0	183	-	4 763	4 763
	Backend	0	0		0	1 614		0	367	300	10 000	10 000

Figure 17. HA-Proxy tool Number of requests per server.
Source: Prepared by the authors.

10000 request are not equally distributed in both servers although they have the same characteristics, the server 1 received 5237 and server 2 received 4763 (see next graphic).

DISCUSSION

Table 2. Programming Algorithm Results

Algorit.	Med.	Máx	Mín	Rend.	Rend. Kb/s	Cons. CPU	Cons. mem.
Round-Robin	33	262	3	987,7 /sec	371.33	58.1 %	2.2 %
Static Round Robin	169	1471	3	889,8 /sec	334.56	60.7 %	3.0 %
Least Connection	49	1726	3	693.9 /sec	260.88	54.5 %	1.3 %

Source: Prepared by the authors.

The Table.2 shows the data summary of the obtained results with each programming algorithm and it its observed that the round robin algorithm average time in seconds that can take a processed request is 33 seconds, the minimum is 3 seconds, the amount of requests per second is 987,7 and the amount of data is 371 kb per second. This shows that the Round Robin programming algorithm requests with HA Proxy efficiency is superior to the others evaluated, but it also must be pointed that in the other two algorithms it is possible to regularize the load, it means to put a higher percentage of requests trough each one of the servers that can distribute the HA-Proxy load.

CONCLUSIONS

The implementation of higher availability allows to business to improve its services, using a higher availability architecture design that depends on the networking hardware, because the studied algorithms can realize variants to load balance in percentages when the servers' characteristics are different.

The evaluated efficiency of the programming algorithms in the proposed infrastructure was favorable to Round-Robin, but it is possible to indicate that in the implementation and tests, servers with the same characteristics were used. However, the Static Round Robin algorithms, Least Connection, can be used when the infrastructure includes equipment with different characteristics that allow to establish percentages of requests that can receive each server.

This research work can be used as a methodology to evaluate the performance of high-availability programming algorithms, in different scenarios that may vary according to the case study, where the traffic analysis can perform with different technology architecture.

BIBLIOGRAPHIC REFERENCES

Bowen, R. and Coar, K. (2007). *Apache Cookbook: Solutions and Examples for Apache Administration*. .º'Reilly Media, Inc.".

Cáceres Alvarez, L. M., Bernabé, P., and Alejandro, M. (2011). Modelo de programación asíncrona para web transaccionales en un ambiente distribuido. *Ingeniare. Revista chilena de ingeniería*, 19(1):26–39.

de la Mora, S. (1997). Hablando HTTP.

Kurose, J. F. and Ross, K. W. (2009). *Computer networking: a top-down approach*, volume 4. Addison Wesley Boston, USA.

Márquez Díaz, J., Sampedro, L., and Vargas, F. (2011). Instalación y configuración de apache, un servidor web. *Revista Científica Ingeniería y Desarrollo*, 12(12):10–23.

Ríos Pérez, F. E., Polanco Carrillo, F., and Moreno Vega, V. (2017). Servidor web empotrado en un fpga para configurar un controlador maestro del sistema inteligente de tráfico cubano. *Revista Cubana de Ciencias Informáticas*, 11(2):16–28.