Cluster implementation on mini Raspberry Pi computers using Round Robin Algorithm

E Rohadi[1], A Amalia[2], A Prasetyo[1], M F Rahmat[1], A Setiawan[2], I Siradjuddin[2]

[1] Department of Information Technology, State Polytechnic of Malang, Jalan Soekarno Hatta 9, Malang, Indonesia
[2] Department of Electrical Engineering, State Polytechnic of Malang, Jalan Soekarno Hatta 9, Malang, Indonesia

Email: amalia@polinema.ac.id

Abstract. The incredible growth of the internet over the past five years makes the data traffic increase significantly. When a single server gets requests from many clients, the overload will likely occur, so requests from many clients cannot be handled optimally. This results in decreased server performance. In this work, cluster implementation on mini raspberry pi computers using the load balancing method is proposed. This system used the Round Robin Algorithm in the scheduling process in each CPU queue. The implementation of server cluster consisting of Raspberry Pi with the addition of load balancing is done to overcome the limited resources on the server. When clients in the amount of 100, 200, and 500 generated by sending 1000 requests simultaneously on each test, the response times obtained are 67.3 ms, 95.3 ms, and 182.3 ms, while throughputs are 62.50 Kbps, 76.42 Kbps, and 79.62 Kbps, respectively. The results show that the responses time and throughputs are stable with Round Robin algorithm contribution. That means the Round Robin Algorithm has availability in server clustering purposes. The system promises to be an alternative server solution that is cheaper and overcomes resource limitations on the server.

1. Introduction

The incredible growth of the internet over the past five years makes the data traffic increase significantly. When a single server gets requests from many clients, the overload will likely occur, so requests from many clients cannot be handled optimally. This results in decreased server performance and end users often experience poor response time. A famous example is Amazon.com website when they suffered from a forty-minute downtime due to an overload of the website on November 2000 [1]. There are many options for dealing with these problems, one of them is clustering. Clustering offers a solution for handling even distribution of access loads from one server to another by using a load balancing method [1-3]. In load balancing, there are two methods used, specifically static methods and dynamic methods. There are several algorithms for load balancing on static methods such as Round-Robin, Ratio, etc. While in the dynamic method there are also several algorithms such as Least Connection, Predictive, etc. [4].

In this work, cluster implementation on mini raspberry pi computers using the load balancing method is proposed. This system used the Round Robin Algorithm in the scheduling process in each CPU queue. The Round Robin (RR) algorithm is one of the simplest load balancing algorithms and effective for CPU utilization [5, 6]. These algorithms decide when and for how long each system runs, they make

alternatives about perceptibility, turnaround time, ready time, response time and other system characteristics [7]. There is a waiting time (quantum time) in sequence when the transfer process between servers occurs. The moment of the quantum time runs out or the process is finished, the CPU will be allocated to the next process [8]. In this scheduling algorithm process, there are no CPU processes that prioritized, all processes on each CPU get the same time allotment. This study aims to determine that prototype clustering on raspberry pi can be an alternative server solution that is relatively cheaper and for resource limitations on the server.

2. The descriptions of system

2.1. System architecture

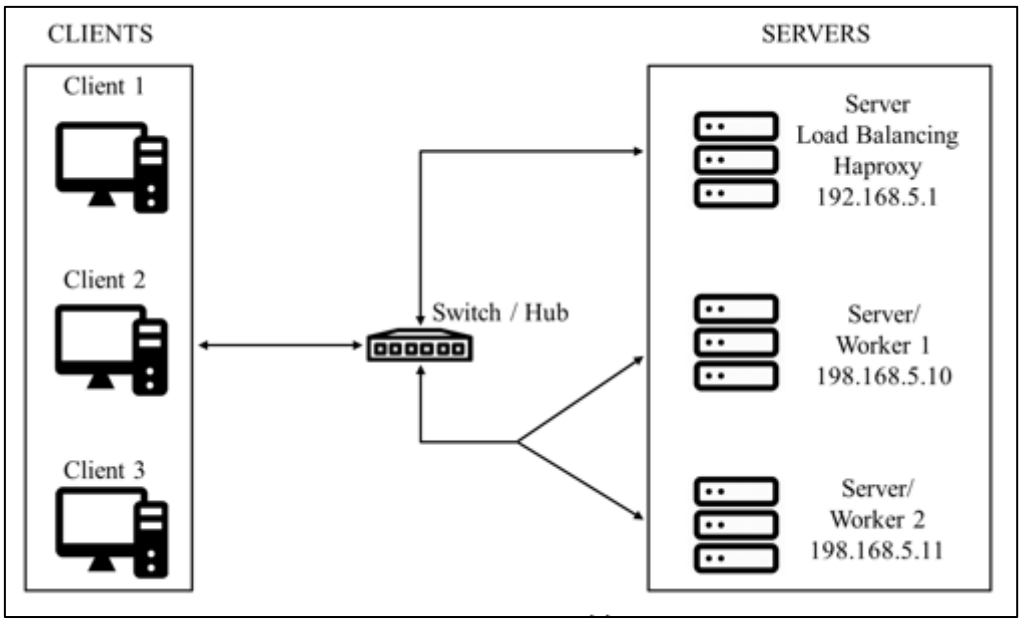

Figure 1. System architecture.

Figure 1 shows the topology used to implement the system. The system consists of several devices, which is:

- Client, user / client that accessing the web server.
- Switch / Hub, device that serves to forward packets on the network a device that acts as a network policy against the network.
- Web Server, a device that provides web services or applications.

The HAProxy system configuration for the server cluster used Raspberry Pi in its implementation. Haproxy itself is one of the most popular open-source loads balancing software, which also offers high availability and proxy functionality. HAProxy aims to optimise resource usage, maximise throughput, minimise response time, and avoid overloading any single resource [2][9]. There are 3 Raspberry Pi as tools for server cluster with details of 1 Raspberry Pi as Frontend Server and 2 Raspberry Pi as Backend server or worker. Frontend server used to configure the HAProxy configuration such as how the Round Robin load balancing algorithm runs and manages the work of the server workers such as IP and port settings for the backend server worker. Backend server used as a server worker which can be added as needed. The more server workers registered at the frontend server make the frontend server stronger when handling the traffic loads.

Each server uses Nginx a powerful web server that can do important things such as load balancing, HTTP caching, or be used as a reverse proxy. The IP addresses configuration on the backend server uses a static IP where server 1 uses IP 192.168.5.10 and on server 2 uses IP 192.168.5.11. In this work, load balancing that was designed using Raspberry pi has only one network interfaces to forward requests from members of the server cluster with IP 192.168.5.1. To overcome the problem of connection between the client and the HAProxy server, the authors added an additional interface in the form of a Wi-Fi adapter with a DHCP IP configuration.

2.2. Block diagram

Figure 2 shows a block diagram that explains when a client computer requests a static page to the frontend server that has the HAProxy application installed. The HAProxy application determines which server will respond to client requests so that no server is down due to heavy load. In the server cluster process, the Round Robin Algorithm works as a server load balancing so that the connection traffic between servers runs smoothly. The Round Robin algorithm runs to determine the turn of processes in each CPU queue.

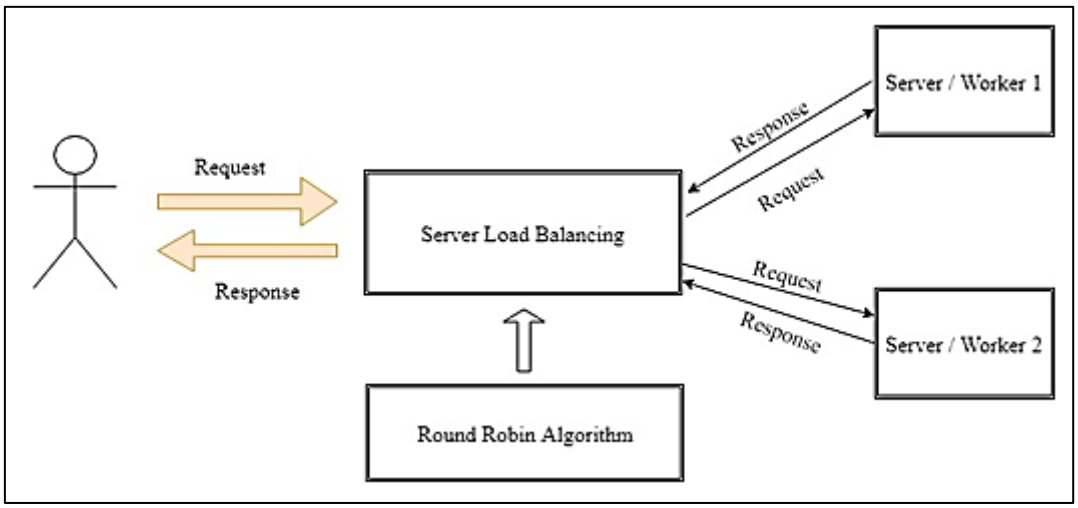

Figure 2. Block diagram.

2.3. System flowchart

Figure 3 shows the system flowchart used to implement the system. Explanation of the load balancing system flowchart is as follows:

- The client computer sends HTTP request to the server cluster load balancing.
- The HAProxy load balancing server receives HTTP request from clients by distributing requests to workers using the Round Robin Algorithm.
- If overload occurs, the load balancing server will send an error message and make requests to other workers that are not overloaded.
- The worker will send a request from the client to the load balancing server.
- The load balancing server will send responses to client computers in the form of HTML pages.
- The client receives the request in the form of an HTML page from the load balancing server.

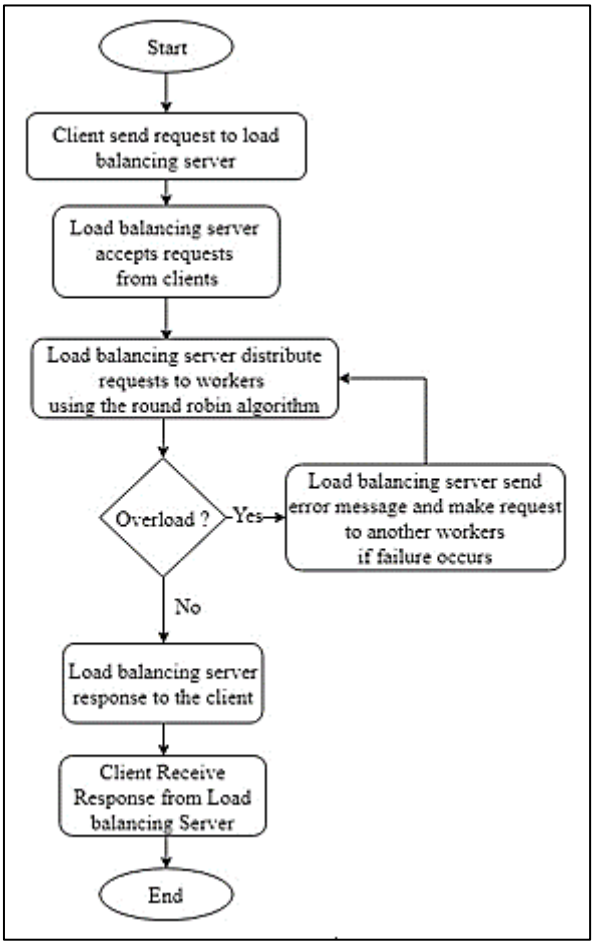

Figure 3. System flowchart.

3. Results and discussions

This section discusses the results of the proposed Round Robin algorithm and compares it with the Least Connection algorithm. The comparison with Least Connection algorithm is done in order to examine the Round Robin performances [4]. The testing performed on a local network with the same scheme as in Figure 3. To view web content, the client must access the IP server load balancing 192.168.5.1 which then forwarded to the worker. Figures 4 and 5 show a simple web display when testing is done to prove that the server worker is running properly.

Figure 4. The web displays of server cluster worker 1.

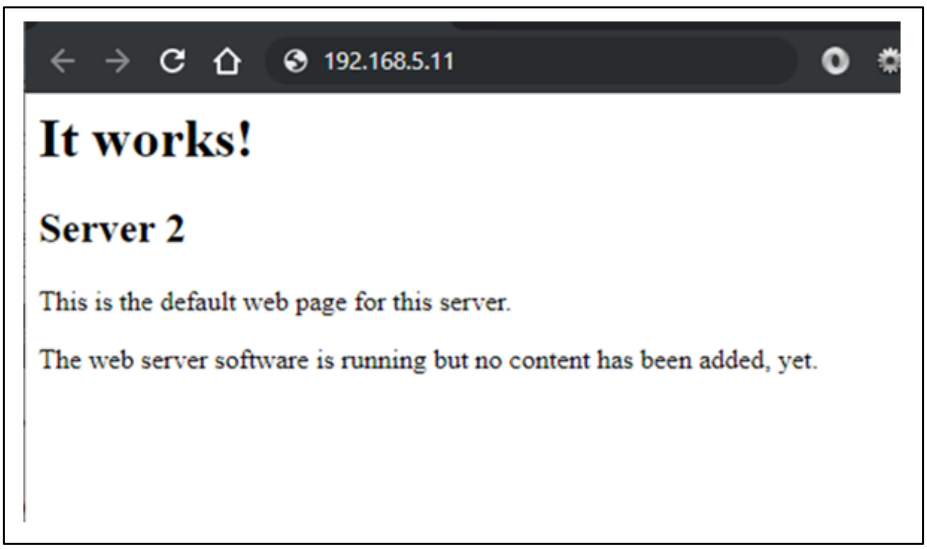

Figure 5. The web displays of server cluster worker 2.

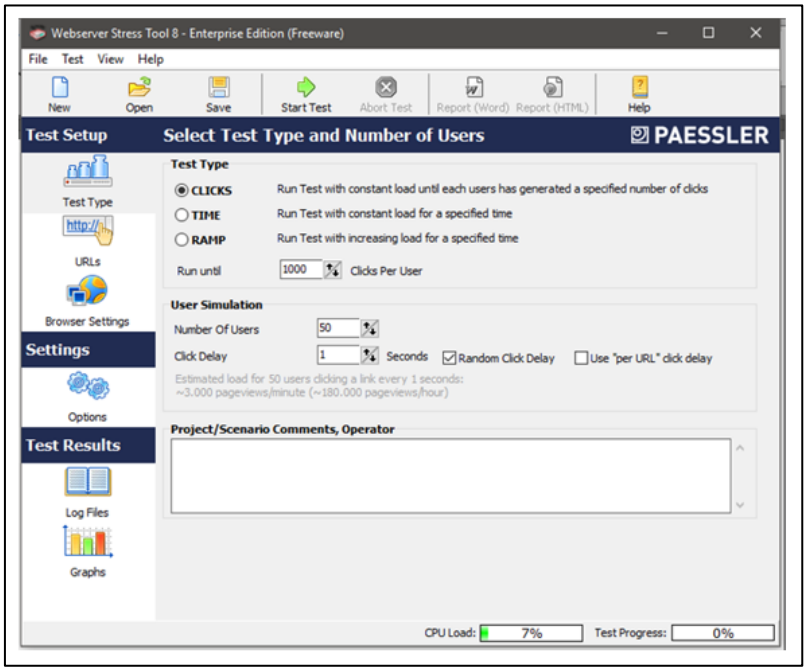

Figure 6. Stress webserver tools.

Stress webserver tool used for testing as shown in Figure 6 to test QoS (Quality of Service) performance on the client-side. The test examines parameters such as throughput, response time, and request loss as a reference in three times testing to see whether the system running well or not. The test results then calculated so that the average value obtained.

3.1. Single web server testing scenario (without load balancing)

Testing is done by generating several requests sent from clients to a single web server and received directly by the web server as shown in Figure 7. After that, a single web server will immediately respond to the client computer without intermediaries from the load balancing server. In this test, the author generates clients in the amount of 100, 200, and 500 by sending 1000 requests simultaneously on each test. Repeated testing is required up to 3 times to get accurate results.

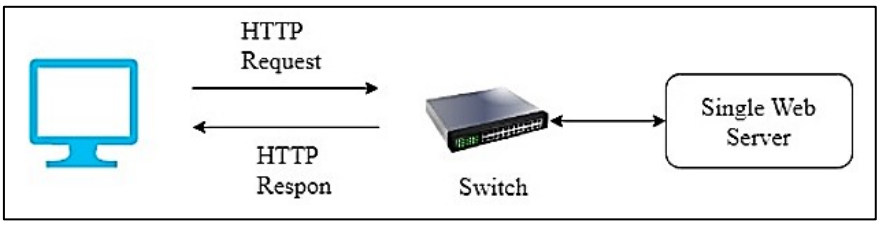

Figure 7. Single web server testing scenario.

Table 1 shows the results of single web server QoS testing. The results show that the response time corresponds to the number of clients. On the other hand, experimentally throughput shows that the number of clients affects significantly. That means the system needs a load balancing procedure [10].

Table 1. Single web server QoS test results.

The number of clients	The number of Request	Response Time (ms)	Request Loss (%)	Throughput (Kbps)
100	1000	78	0	16.33
200	1000	144	0	12.71
500	1000	202	0	8.47

3.2. Web server cluster using round robin algorithm testing scenario (with load balancing)

The test scenario performed on two web servers or workers with a load balancing server as shown in Figure 8. The client sends a request to the load balancing server IP address with IP 192.168.5.10, then the load balancing server will accept the request sent by the clients. The requests sent by the client will be distributed to workers using the Round Robin scheduling algorithm [11, 12]. All requests from clients to workers will pass through the load balancing server as intermediaries to avoid overloaded traffic. QoS performance testing on the client-side is done by generating several clients using the webserver stress tool. To find out the performance significantly, the number of clients will be increased gradually. In this test, the author generates clients in the amount of 100, 200, and 500 by sending 1000 requests simultaneously on each test. Repeated testing is required up to 3 times to get accurate results. The results are shown in Table 2 below. The load balancing procedure of Round Robin algorithm achieves the stability of the throughputs. Furthermore, the number of clients doesn't affect the throughput performance due to all accesses through the load balancing server.

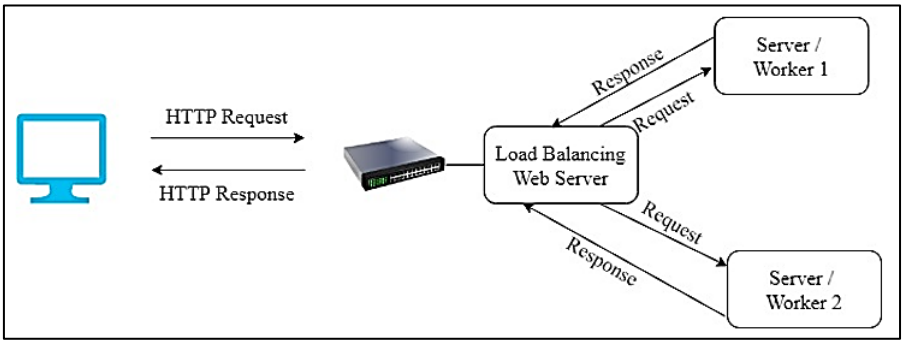

Figure 8. Web server cluster testing scenario.

Table 2. Web server cluster QoS using round robin algorithm test results.

The number of clients	The number of Request	Response Time (ms)	Request Loss (%)	Throughput (Kbps)
100	1000	67.3	0	62.50
200	1000	95.3	0	76.42
500	1000	182.3	0	79.62

3.3. Web server cluster using least connection algorithm testing scenario (with load balancing)

The implementation of the least connection algorithm has a similar test scenario as shown in Figure 8. The results show that the Least Connection algorithm gives the same performance as the Round Robin Algorithm as shown in Table 3. Therefore, the throughputs a bit lower than Round Robin Algorithm due to random server load balancing access. Figures 9 and 10 show the throughput and response time with a different number of clients, respectively.

Table 3. Web server cluster QoS using least connection algorithm test results.

The number of clients	The number of Request	Response Time (ms)	Request Loss (%)	Throughput (Kbps)
100	1000	400	0	32.50
200	1000	385	0	46.42
500	1000	350	0	55.40

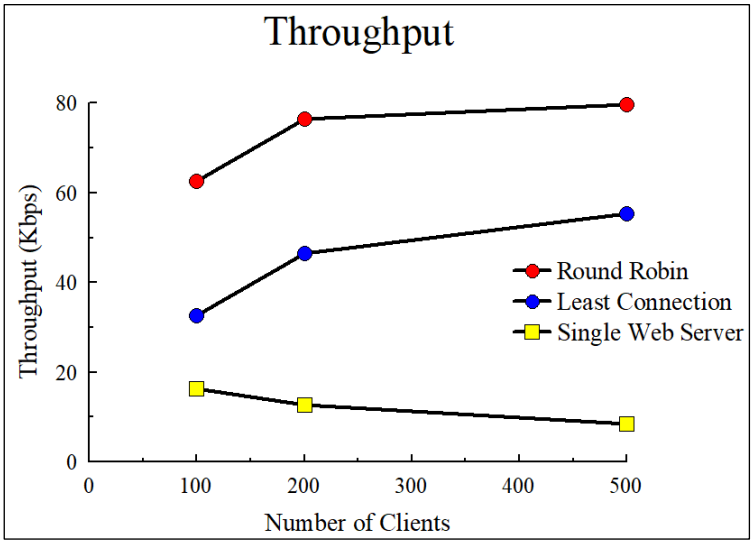

Figure 9. The graph of throughput.

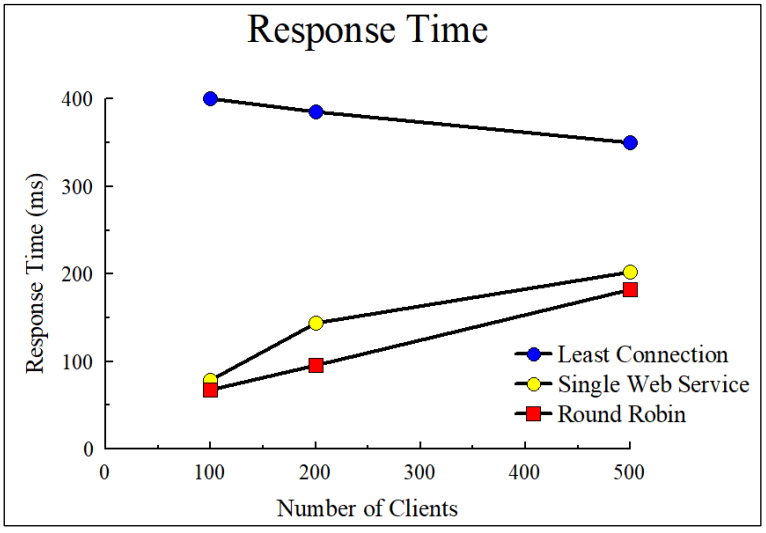

Figure 10. The graph of response time.

4. Conclusions

Study on cluster implementation by using Round Robin Algorithm has been presented. The Least Connection has been applied in order to examine the Round Robin performances. When clients in the amount of 100, 200, and 500 generated by sending 1000 requests simultaneously on each test, the

response times obtained are 67.3 ms, 95.3 ms, and 182.3 ms, while throughputs are 62.50 Kbps, 76.42 Kbps, and 79.62 Kbps, respectively. The analysis results show that the responses time and throughputs are stable with Round Robin algorithm contribution. As a result, the priority over server sequences on Round Robin Algorithm contributes in managing the optimum throughputs of the server cluster. That means the Round Robin Algorithm has availability in server clustering purposes. The system promises to be an alternative server solution that is cheaper and overcomes resource limitations on the server.

5. References

[1] Sharma D 2018 *Int. Conf. on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)* 471-476
[2] Maduranga M W P and Ragel R G 2016 *Int. Computer Science and Engineering Conference (ICSEC)* 1-4
[3] Yu A and Yang S 2018 *The Journal of Supercomputing* 1-10
[4] Suwandika I P, Nugroho M A and Abdurahman M 2018 *Int. Conf. on Information and Communication Technology (ICoICT)* 459-463
[5] Farooq M U, Shakoor A and Siddique A B 2017 *Int. Conf. on Communication, Computing and Digital Systems (C-CODE)* 244-248
[6] Reddy N S, Santhi H, Gayathri P and Jaisankar N 2018 *System and Architecture* **732** 231-240
[7] Upadhyay A and Hasija H 2016 *Information Systems Design and Intelligent Applications* **434** 457-467
[8] Ghosh S and Banerjee C 2018 *Int. Conf. on Research in Computational Intelligence and Communication Networks (ICRCICN)* 33-37
[9] Wang M and Guan J 2017 *Journal of Communications and Information Networks* **2** 30-40
[10] Sheikhi S and Babamir S M 2016 *The Journal of Supercomputing* **72** 588-611
[11] ElDahshan K, El-kader A A and Ghazy N 2017 *Int. Journal of Computer Applications* **172** 15-20
[12] Zongyu X and Xingxuan W 2015 *Chinese Control Conference (CCC)* 5804-5808

Acknowledgments
The authors would like to thank The State Polytechnic of Malang for the support to attend this conference.