

Layer-4 Load Balancer on Programmable Data Plane using IP Hash and Weighted Round Robin

Muchlis Ramadhan Usman, Muhammad Arief Nugroho, Ibnu Asror,

^{1,2,3}Fakultas Informatika, Universitas Telkom, Bandung

¹muchlisramadhan@students.telkomuniversity.ac.id,

²arif.nugroho@telkomuniversity.ac.id,

³iasror@telkomuniversity.ac.id

Abstrak

Layer-4 Load balancer digunakan di data center untuk mendistribusikan permintaan yang datang dari klien ke beberapa server yang terletak di data center. Ada dua persyaratan untuk membangun penyeimbangan beban Layer-4 yang layak. Yang pertama adalah memastikan keseragaman distribusi permintaan di beberapa server dan afinitas koneksi. IP Hash adalah algoritma load balancing yang umum digunakan untuk mengimplementasikan penyeimbangan beban Layer-4 tetapi dapat menyebabkan ketidakseimbangan beban di beberapa server. Weighted Round-Robin diusulkan untuk mencegah ketidakseimbangan beban di beberapa server. Implementasi load balancing menggunakan programmable data plane daripada menggunakan penyeimbangan beban perangkat keras atau perangkat lunak tambahan. Hasilnya menunjukkan bahwa WRR mampu mengurangi ketidakseimbangan beban di beberapa server dengan mencapai distribusi permintaan yang seragam. WRR mencapai throughput 13% lebih tinggi tetapi memberikan response time 2% lebih tinggi daripada IP Hash dan memiliki lebih sedikit packet loss daripada IP Hash saat menangani permintaan HTTP

Kata kunci: Layer-4, Load Balancer, Connection Affinity, IP Hash, Weighted Round Robin, Data Plane

Abstract

Layer-4 load balancer used in data center to distribute requests coming from client to multiple servers located in data center. There are two requirements to build proper Layer-4 load balancing. The first is ensuring the uniformity of request distribution across multiple servers and connection affinity. IP Hash is a common load balancing algorithm that is used to implement Layer-4 load balancing but can cause load imbalances across multiple servers. Weighted Round-Robin is proposed to prevent load imbalance across multiple servers. The implementation of load balancing is using a programmable data plane rather than additional hardware or software load balancing. The result shows that WRR is able to mitigate load imbalance across multiple servers by achieving uniform request distribution. WRR achieves 13% higher throughput but gives 2% higher response times than IP Hash and has less packet loss than IP Hash when handling HTTP requests.

Keywords: keyword Layer-4, Load Balancer, Connection Affinity, IP Hash, Weighted Round Robin, Data Plane

1. Introduction

1.1 Background

In the data center there are multiple servers operated to handle large requests from clients. To manage this request load balancing is needed [1]. Layer-4 load balancer can be used to distribute requests across multiple servers. Layer-4 load balancing makes the request load balancing decision based on information in Layer-4 that is transport layer [2]. Layer-4 in this study refers to the fourth layer of the Open System Interconnection System (OSI). Layer-4 Load Balancer sending a request to corresponding server by using 5-tuple information of such as IP Address Source, IP Address Destination, Port Source, Port Destination, and Transport protocol to determine the server to handle the request coming from client [2].

To build proper Layer-4 Load balancer there are two requirements that need to be met [3] [4]. First is uniformly distributing requests from clients to servers in the data center. Second is ensuring packet forward to the same server from the first packet forwarded to establish a session until the last packet to close a session from client to server or called connection-affinity.

The first requirement that is uniformly distributing the packet is important to split large requests coming from clients and distribute to multiple servers located in a data center to prevent load imbalance. The second requirement is ensuring Connection-affinity [4] [5]. This concept is achieved when the set of packets to establish a session or called flow sent consistently to the same server until the last packet to close the session and not send it to different server. The purpose of ensuring Connection-affinity is to prevent request failure. The IP Hash algorithm is used in load balancing systems such as Hardware, Software, and Software Defined Network. The IP Hash algorithm is used for Layer-4 Load Balancing in data centers because of algorithm capability in distributing request across multiple servers and ensure connection affinity [6] [7] [8].

1.2 Problem statement

However, IP Hash load balancing suffers from load imbalance [3]. IP Hash relies only on hash calculation to determine server index, this can lead to load imbalance.

To mitigate IP Hash Problem WRR algorithm is proposed. The WRR algorithm works by distributing the request in turn for each server but also takes consideration of server capability in handling n number of requests from client [9]. By taking consideration of server capability in handling request, WRR can mitigate load imbalance cause by IP Hash that only depend on hash function to determine server index.

1.3 Purpose

WRR is scheduling algorithm that derives from Round-Robin algorithm, In Round robin the request is forwarded to the server in turn starting from the first server in index to the last server in index. WRR is similar to Round Robin, but the difference in WRR is the weight is assigned to the servers [5]. The Weight is determined by network administrator after observing the capability of servers in handling request, such as link bandwidth from load balancer to the server and/or server specification. To maintain connection affinity in WRR algorithm, Hash function from IP Hash is used and combined with WRR Algorithm.

Implementation of IP Hash and WRR algorithms can be done in hardware, software, or SDN load balancing using F5 hardware, NGINX software, or Openflow Protocol for software-defined networks[6] [8] [10]. But recently there is an emerging paradigm that is Programmable Data Plane. Programmable data plane is a concept that allows programmability of packet processing network directly in data plane [11]. With this concept load balancing can be applied directly in programmable switch without the need for middleware. IP Hash and WRR algorithm can be implemented directly in Programmable switch using P4 (Programming Protocol-independent Packet Processors) [12] programming language.

In this study WRR is proposed to mitigate load imbalance caused by IP Hash algorithm. By using WRR, each server will be assigned weight to prevent load imbalance. Implementation of IP Hash and WRR will be implemented using P4 programming language in BMv2 programmable switch.

2. Related study

This section covers some of related the study conduct in context of Layer-4 load balance. The related study of this paper is presented in the form of a table. In Table 1 consists of Title of the paper, method used in the paper, result of the paper, and the relevance of paper cited with the author paper.

Table 1 Cited paper

No		
1	Title	Cheetah: A High-Speed Programmable Load-Balancer Framework With Guaranteed Per-Connection-Consistency [3]
	Method	Stateful Cheetah load balancer Stateless Cheetah load balancer
	Result	Have comparable outcome performance and have 3.5x cycles per packet less than state of the art load balancing
	Relevance	Raise similar problem statement, state that hash mechanism load balance cause imbalance in data center Use P4 language to implement load balancing
2	Title	Analisis Performansi Load Balancing menggunakan Algoritma Round Robin dan Weighted Round Robin pada P4-Programmable Switch [13]
	Method	Round Robin algorithm Weighted Round Robin algorithm
	Result	As shown Weighted Round Robin algorithm perform better that Round Robin in term of QoS parameter like Throughput, Jitter and latency
	Relevance	Use P4 language to realize the load balancing system Use Weighted Round Robin algorithm
3	Title	Increasing SDN Network Performance using Load Balancing Scheme on Web Server [8]
	Method	IP Hash algorithm Least Connection IP algorithm
	Result	IP Hash 17% more optimal than Least Connection Algorithm
	Relevance	The algorithm used in the paper is IP hash algorithm. IP Hash algorithm is used to maintain connection affinity The implementation of load balancing is in context of Software Defined Network using OpenFlow API, which dependent on the control plane
4	Title	Analysis of Load Balancing Performance using Round Robin and IP Hash Algorithm on P4 [14]
	Method	IP Hash algorithm Round Robin algorithm
	Result	Round Robin gives better results of fairness index. For throughput the result is comparable between the two algorithms, while the response time parameter result is very contrast in comparison, IP Hash is 12,6 ms and Round Robin is 3,31 ms
	Relevance	Implementation of algorithm load balance in data plane is the same using IP Hash algorithm
5	Title	SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs [15]
	Method	Hashing method to reduce memmory usage Table to store connection state
	Result	Shows that a programmable swith asic able to replace thousand of software load balancer
	Relevance	Use P4 programming language to implement load balancing in programmable switch

2.1 Layer-4 Load balancing in data center

Layer-4 Load balancing in multi-tier data center architecture is located in between router and backend server [3]. Is figure 1 shows the traditional data center load balancing architecture. The architecture is organized in 3 different tiers, the 1st tier consists of BGP Router using ECMP. The second tier is the layer-4 Load balancer that will be the focus of this study. And the Third tier is Layer-7 load balancer that operates in the application layer.

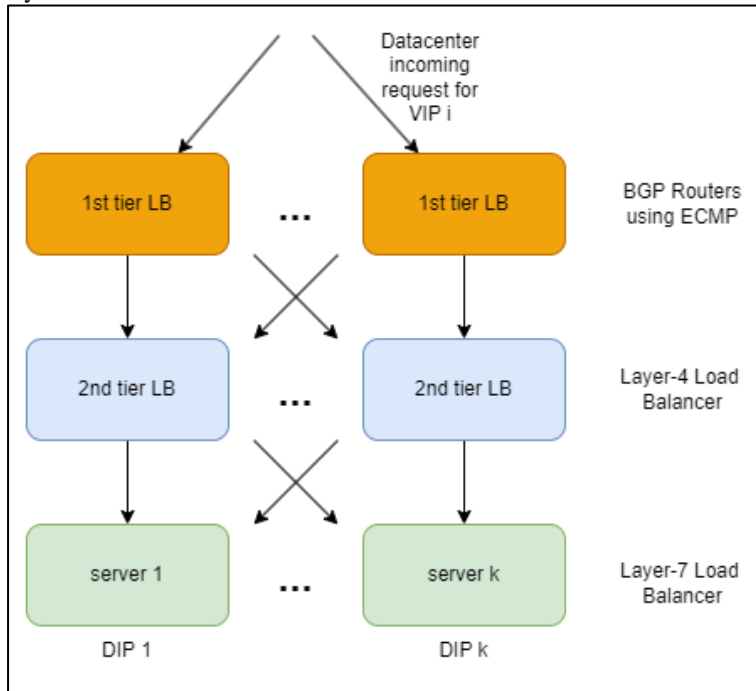


Figure 1 Traditional data center load balancer architecture

Request coming from client to specific service in data center is correspond to Virtual IP (VIP). Multiple servers that provide the service in data center are identified with Direct IP Address (DIP) with unique IP Address as identifier for each back-end server. Each DIPs are associated with a VIP to represent as one services

2.1.1 Traditional Layer-4 Load Balancing

Implementation load balancing in data center using Hardware or Software load balancing is place in between switch and back-end server. As shown in figure 2, additional hardware or software is used for load balancing. This can add additional delay when handling requests from client to server.

Implementing load balancing system in programmable switch can remove the need for additional load balancing hardware or software. As shown in figure 3, when the request comes from client through internet it directly goes to switch, and the packet processing will be performed in the programmable switch. This resulting in reduction of cost in provisioning additional hardware or software load balancing and increase the Throughput and Response time because the packet is get processed in line rate [15].

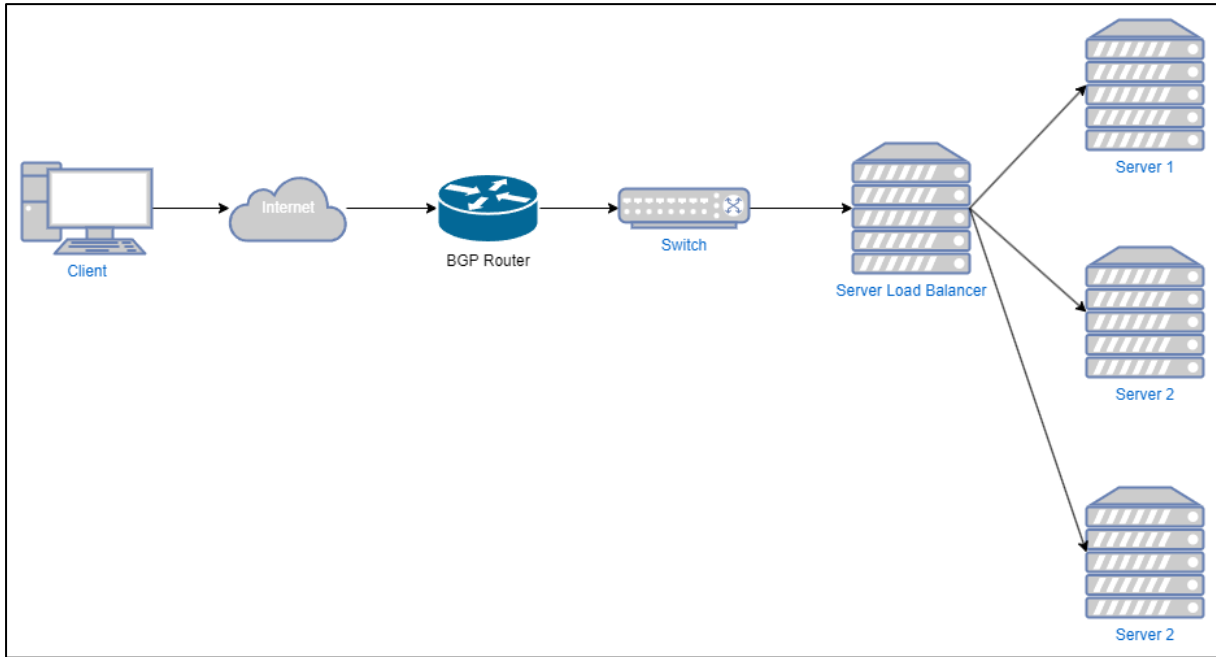


Figure 2 Traditional Layer-4 Load Balancing

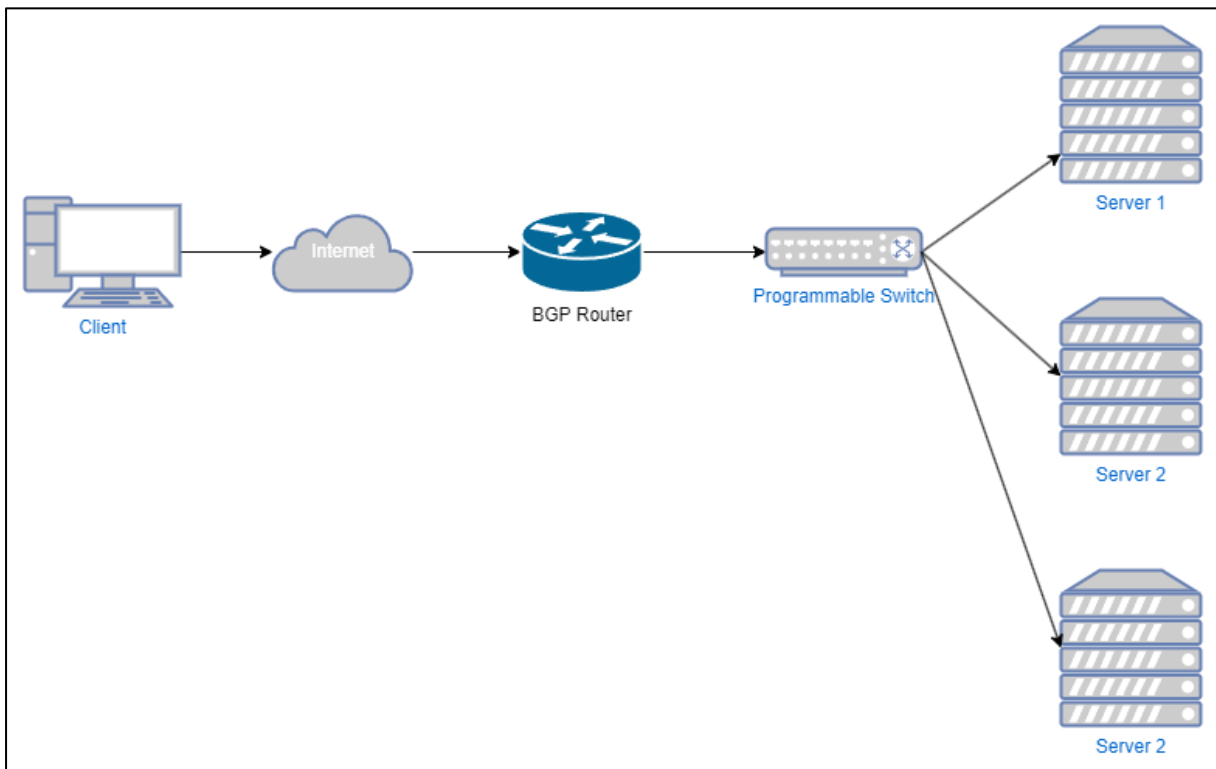


Figure 3 Programmable data plane load balancing using programmable switch

2.2 Layer-4 load balancing.

Multiple servers that are in data center being advertised to outside world by virtual IP address (VIP). This can be achieved by leverage layer-4 load balancer by mapping the direct IP address (DIP) to load balancer. Layer-4 load balancer distribute the packet from clients to multiple in data center and balance the load across the server [16]

1. Uniformly distribute request

Uniformly distributing requests across multiple servers located in data center is important because it can prevent a server from getting overload or underload. When a server is not overloaded

or underloaded it can give best overall performance such as High Throughput, low response time and avoid request failure.

2. Ensuring Connection Affinity

When a new connection is established to a server the subsequent packet needs to be sent to same server as first establish packet, this called connection affinity [17]. Connection affinity is required to build a layer-4 load balancing ensuring that connection sessions are maintained during requesting service and not get terminated and reset.

2.3 IP Hash

In load balancing systems like hardware, software, and software-defined networks [6] [7] [8]. The IP hash algorithm is used because the IP Hash algorithm can distribute requests among several servers in a data center and guarantee connection affinity. It is utilized for Layer-4 Load Balancing in data centers. To create the hash value, IP Hash takes 5-tuples information such IP Address Destination, IP Address Source, TCP Port Destination, TCP Port Source, and Transport Protocol. Since the hash value produced from the five-tuple information will remain constant for each packet of a flow arriving from the client, it will be utilized to guarantee connection affinity.

2.4 WRR

The Round-Robin method is the source of the WRR scheduling algorithm. In Round-Robin, requests are sent to each server in turn, starting with the first server in the index and ending with the last server in the index. WRR and Round Robin are similar, WRR differs is that the servers are given a weight [5]. After assessing the server capabilities to handle requests.

2.5 Programmable data plane

Programmable Data Plane is a concept that allows for direct programming of data plane hardware or software to customize network device forwarding behavior [18]. Historically, firmware determined data plane behavior, but with SDN popularity, data plane programmability is crucial for adapting network designs to changing traffic patterns, application needs, and security risks.

2.6 Programmable Switch

A network switch with customizable packet processing operations is called a programmable switch in a programmable data plane [2]. Network operators and developers can build and implement their own packet processing logic using programmable switches, in contrast to standard switches that have fixed functions and protocols hardcoded by suppliers.

2.7 Programming protocol-independent packet processors (P4)

P4 is a programming language that is specifically used to program data planes. By using P4 the network operator can implement additional functionality and packet processing to the switch. It has the ability to define packet header and match action for each packet that traverse the switch [19]

3. System design

3.1 Topology

The topology system used for this study is shown in figure 4, the client will make HTTP request from the internet to load balancer IP Address (VIP Address) that represent as single service. In this case the load balancer is Programmable Switch that is BMv2 Switch. Behind VIP Address is consisting of multiple servers to handle the request. The multiple servers are identified with unique IP address (DIP Address) that will be chosen by load balancer to handling the request from client.

3 Server used in this study based on data center topology. Usually, data centers employ multiple servers for applications that will be used by the client [20]. Also to analyze the load distribution and the fairness of the load balancing system algorithm.

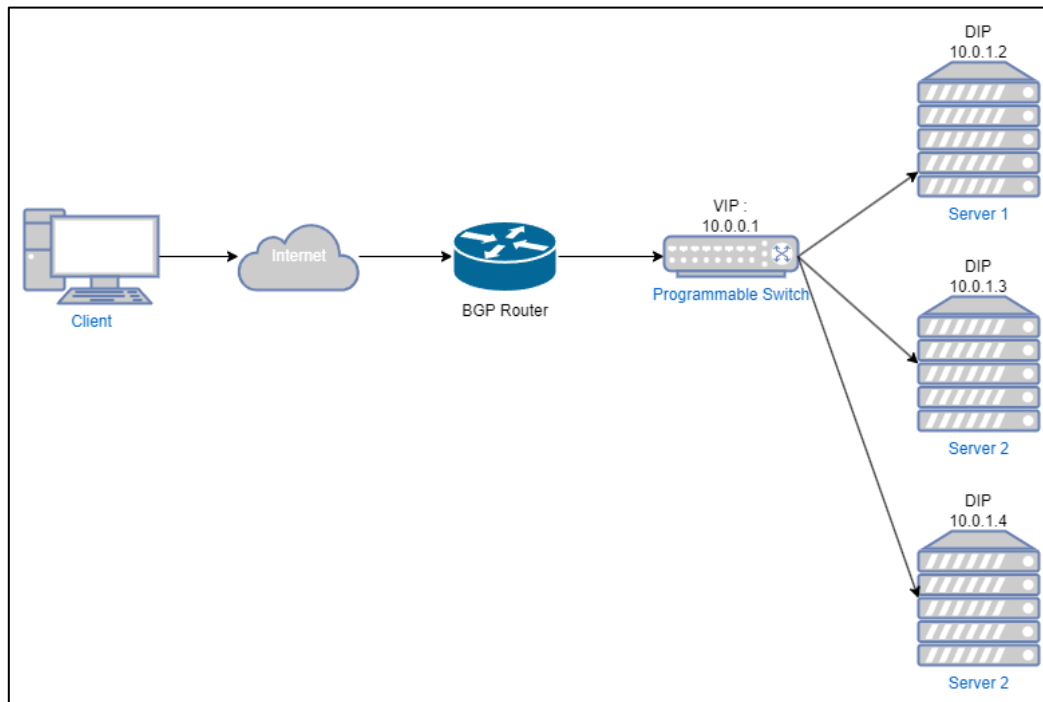


Figure 4 Topology

3.2 IP Hash load balancing

IP Hash load balancing works on programmable data plane by parsing the packet header first and examine the field of the packet. In packet header there are 5 tuple information that is IP Source, IP Destination, Port Source, Port Destination, And transport Protocol. Transport protocol particularly TCP is checked if it is new connection or not. This can be done by looking at TCP flag and checking whether it is SYN flag. If it is a new connection the P4 program will assign the connection to the server.

To assign new connections to the server, the Hash algorithm is used to generate hash value. The parameter used to generate hash value is 5-tuple information. The next following packet after the first TCP packet always be the same thus the hash value generated will remain the same. Therefore, the connection affinity can be guaranteed.

In figure 2 shows how IP Hash algorithm mechanism. Starting when a request comes from client then the TCP/IP protocol header is used for Hash parameters. Hash value generated will be used to maintain connection affinity and to determine server index.

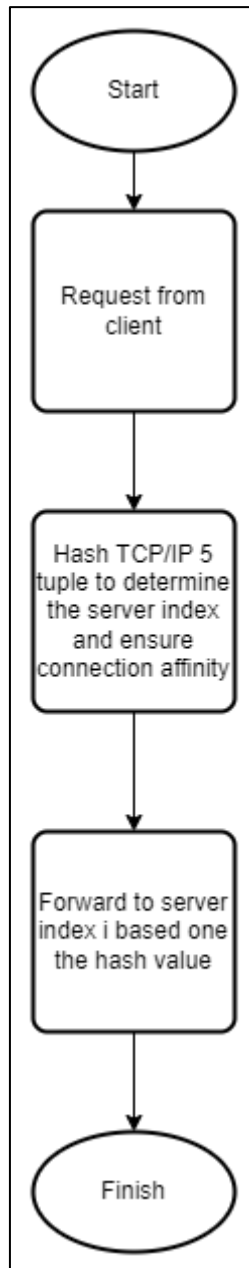


Figure 5 IP Hash mechanism

3.3 Weighted round robin load balancing

To achieve efficient load balancing system WRR distribute packet to server from client based on determined weight set by network administrator [5]. The parameter to determine server weight. This way can help take consideration of varying resource of server and enhance system performance.

In Figure 3 shows WRR algorithm mechanism, starting by assigning server weight. The weight is considered based on network administrator after observing the server specification, such as CPU, memory, link bandwidth from switch to server, etc. When request comes from client, the packet get process directly by programmable switch based on the P4 code and algorithm that have been deployed in programmable switch. P4 code algorithm that executed in the programmable switch check whether the request exceed server index i weight, if not send to server index i , and if the request is already exceed server index i weight the request get send to server index $i+1$.

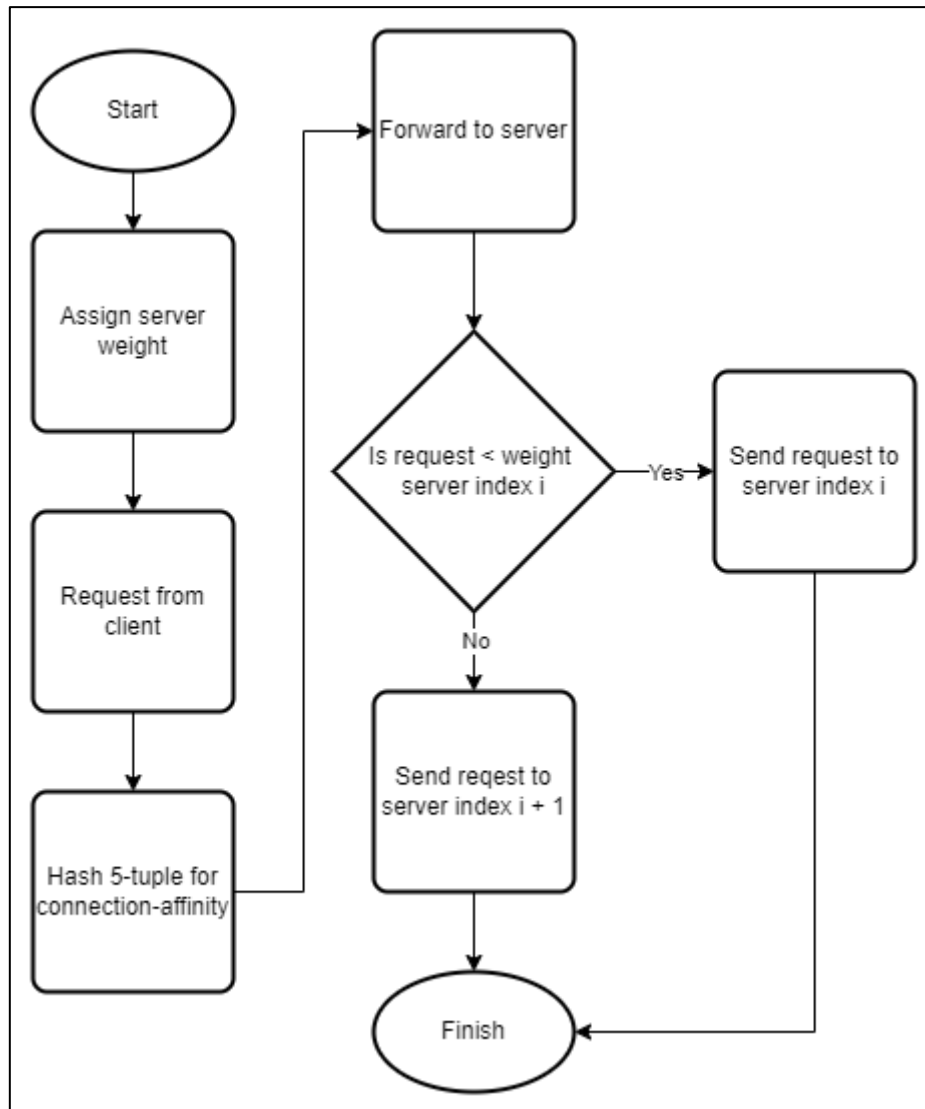


Figure 6 Weight Round-Robin mechanism

3.4 Block diagram of system implementation

This section describes the step by step in applying network load balancing system in virtual environments

In Figure 4 show how load balancing system using P4 language set up in virtual environment

1. Install virtualbox on host machine, then install ubuntu in virtualbox
2. Install P4 development [21] tools in ubuntu
3. Creating topology of load balancing system using mininet python API
4. Writing IP Hash and WRR algorithm in P4 language
5. Compile P4 source code using P4 compiler
6. Creating routing table logic and configure the routing table logic in BMv2 switch
7. Simulate the load balancing system in mininet
8. If routing table logic accepted by BMv2 switch, then load balancing is ready to conduct the performance test

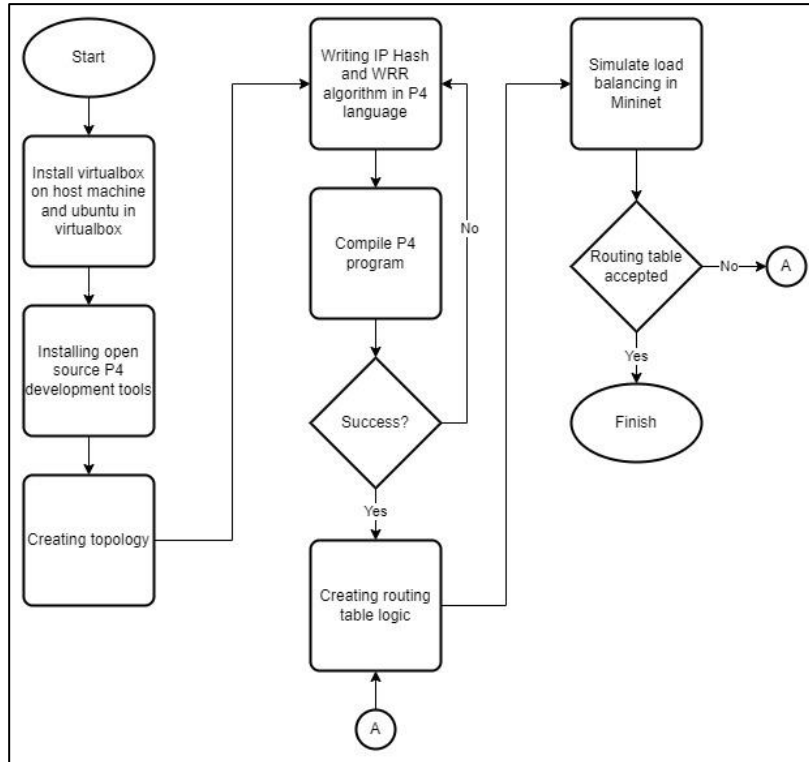


Figure 7 Block diagram of system implementation

3.5 System specifications

The hardware and software that will be used to build load balancing systems listed in table 1 and 2 to conduct the load balancing performance.

Table 2 Hardware specifications

No	Hardware	Description
1	CPU	Intel i7 8750
2	GPU	NVIDIA GTX 1060
3	RAM	24 GB DDR4
4	Storage	118 GB SSD + 256 SSD

Table 3 Software specifications

No	Software	Description
1	Operating system	Ubuntu 22.04
2	Tools	P4 development tool
		BMv2 Switch
		Mininet
		Apache bench

3.6 Test parameters

In this section test parameters used for test performance will be explained. Parameter for Quality of Service (QoS) for test Load Balancing performance based on standatization [25]. To measure the load imbalance in all the servers in the data center, the parameter that will be used is Request distribution and Fairness index [26].

In this test scenario the parameters conducting performance analysis based that will be used is listed below:

1. Throughput

Throughput measures the rate of packets that can be sent or received through a network in a specific timeframe. Throughput results reflect the capacity or efficiency of a network. In table 4 shows the categorization of Throughput qualities.

Table 4 Throughput standarization

Throughput category	Throughput	Index
Bad	0 – 338 kbps	0
Poor	338 - 700 kbps	1
Fair	700 - 1200 kbps	2
Good	1200 kbps – 2,1 Mbps	3
Excelent	> 2,1 Mbps	4

2. Response time

Latency is used to measure the amount of time it takes from server to respond request from client. In table 5 shows the categorization of Response time qualities

Table 5 Response time standarization

Response time category	Delay
Excelent	< 150 ms
Good	150 ms – 300 ms
Fair	300 ms – 450 ms
Poor	> 450 ms

3. Packet loss

To measure the number of packets that failed to be sent to the client. Packet loss during packet transmission to client or server can be caused by collision and congestion in the network. In table 6 show the categorization of Packet loss qualities.

Table 6 Packet loss standarization

Packet loss category	Delay
Excelent	0 – 2 %
Good	3 – 14 %
Fair	12 – 24 %
Poor	> 24 %

4. Request distribution

This metric is used for measuring the number of connections established and from client to server. Analyze the packet header such as TCP SYN flag is captured to analyze the number of connections established between client and server.

5. Jain fairness index

The Jain fairness index parameter is to measure that every server in the data center gets fair share of requests coming from client. The value of the fairness index is from 0 meaning some of the servers get higher or lower load to 1 meaning that all the servers get equal load. In figure 8 shows the formula of Jain fairness index.

$$\frac{\left(\sum_{i=1}^n x_i\right)^2}{n \sum_{i=1}^n x_i^2}$$

Figure 8 Jain fairness index formula

3.7 Test scenario

In this section will be explain how the test performance of load balancing of both IP Hash and WRR algorithm will be conduct

1. Throughput and Response Time test scenario

To conduct Throughput and Response time test scenario for load balancing in performance in Programmable Switch is by using Apache Bench software. In figure 5 shows the command used for test load balancing performance, the number of reqeues from 20.000, 40.000, 60.000, 80.000 and 100.000 is performed to test load balancing performance. The amount of request is chosen based on the amount of request used in data center load balancing to induce load inbalance in the server [3].

```
p4@p4:~/Downloads/tugas_akhir_2/ip_hash$ mx h1
root@p4:/home/p4/Downloads/tugas_akhir_2/ip_hash# ab -n 1000 http://10.0.0.1:8000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Completed 1000 requests
Finished 1000 requests

Server Software:      SimpleHTTP/0.6
Server Hostname:     10.0.0.1
Server Port:         8000

Document Path:       /
Document Length:     910 bytes

Concurrency Level:   1
Time taken for tests: 5.019 seconds
Complete requests:   1000
Failed requests:     0
Total transferred:   1065000 bytes
HTML transferred:    910000 bytes
Requests per second: 199.26 [#/sec] (mean)
Time per request:    5.019 [ms] (mean)
Time per request:    5.019 [ms] (mean, across all concurrent requests)
Transfer rate:       207.24 [Kbytes/sec] received
```

Figure 9 Throughput performance testing

2. Request loss test scenario

To conduct request loss of packet during test performance can be done by using tool Wireshark, in Wireshark the network interface of server 1, 2 and 3 is captured during request test performance. After request test performance done, the packet lost during test performance is observed by using filter function in Wireshark tool. As shown in figure 6 is the command used to observe how many packets are lost.

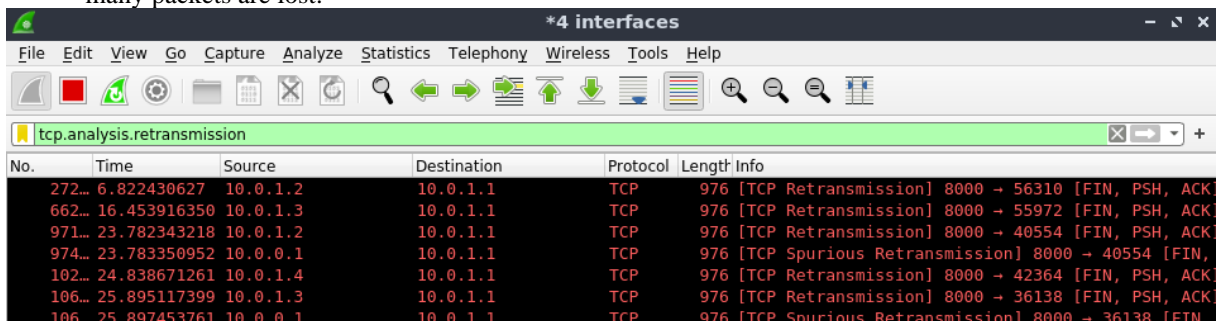


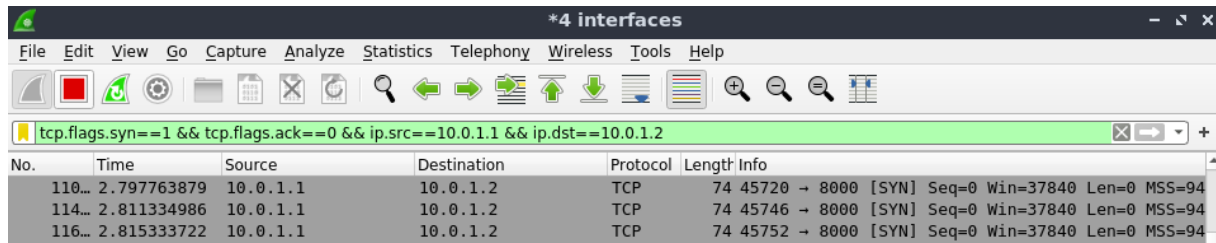
Figure 10 Filtering packet loss

3. Request distribution test scenario

To conduct Request distribution of client generated using Apache Bench tool to each server running Python Simple HTTP server by capture first TCP packet when start a connection between client

and server. The first TCP captured during runtime test performance is TCP SYN, this is because SYN is the first TCP Packet sent to establish a connection between client and server.

In figure 7 shows Wireshark tool capture first TCP SYN packet of every TCP session request from client.



No.	Time	Source	Destination	Protocol	Length	Info
110...	2.797763879	10.0.1.1	10.0.1.2	TCP	74	45720 → 8000 [SYN] Seq=0 Win=37840 Len=0 MSS=94
114...	2.811334986	10.0.1.1	10.0.1.2	TCP	74	45746 → 8000 [SYN] Seq=0 Win=37840 Len=0 MSS=94
116...	2.815333722	10.0.1.1	10.0.1.2	TCP	74	45752 → 8000 [SYN] Seq=0 Win=37840 Len=0 MSS=94

Figure 11 Wireshark tool filtering request distribution

4. Evaluation

4.1 Result and analysis of Throughput

To conduct Throughput performance of load balancing algorithm, 20.000, 40.000, 60.000, 80.000 and 100.000 request is executed. Throughput performance result between IP Hash and WRR in figure 12 shows that WRR algorithm give higher Throughput performance with value 13% bigger that IP Hash. WRR uniformly distributes requests across server resulting in better Throughput performance than IP Hash. WRR preventing the servers from getting overloaded thus server can perform more optimal in handling request

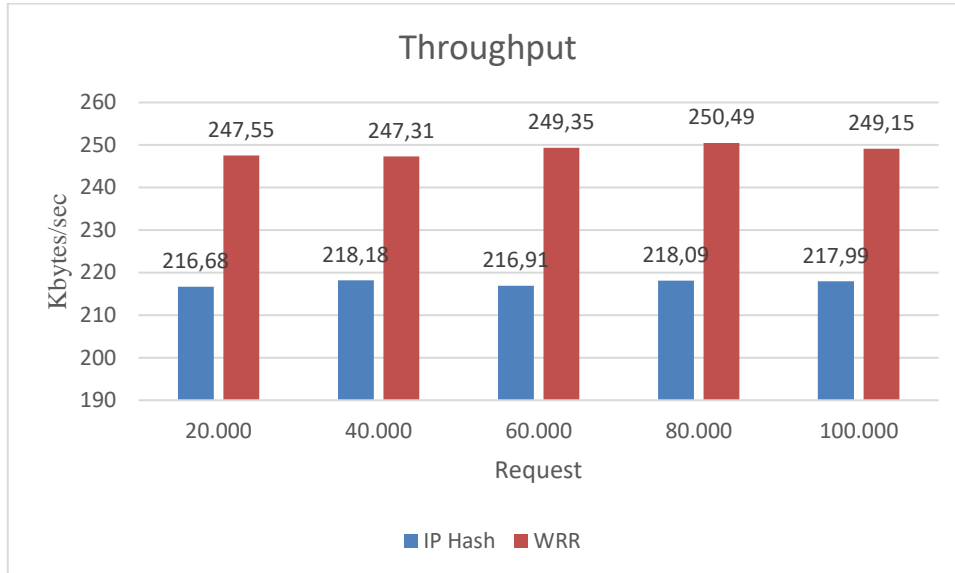


Figure 12 Troughput result

4.2 Result and analysis of response time

To conduct response time performance of load balancing algorithm, 20.000, 40.000, 60.000, 80.000 and 100.000 request is executed. In figure 13 shows that IP Hash perform better by having 2% less Response Time than WRR for every request. IP Hash perform better than WRR because when a request coming from client, Hash value is used to determine server index, while in WRR needs to check first wheter a server can accept request based on the weight assing to the server and then determine server index

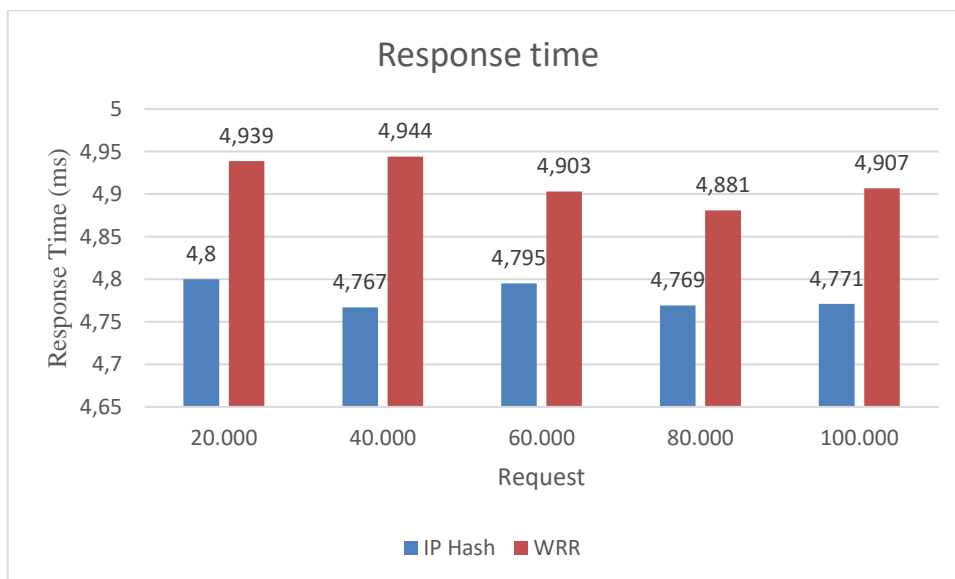


Figure 13 Response time result

4.3 Result and analysis of Packet loss

In figure 14 shows that IP Hash give less packet loss in majority request. But in 100.000 request it shows that IP gives bigger packet loss, this indicates that when IP Hash get bigger request it cause load imbalance in the server and can cause network congestion.

Based on the packet loss standarization refer to section 3.6, it shows both algorithm gives beter packet loss result because the value is less than 2%

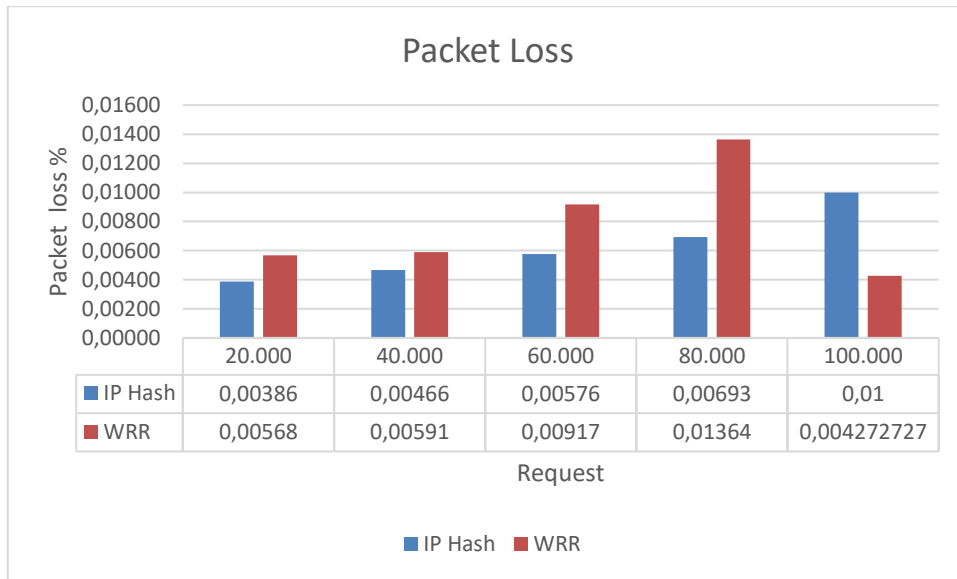


Figure 14 Packet loss result

4.4 Result and analysis of Request distribution IP Hash and WRR

In figure 15 shows the result of request distribution of IP Hash algorithm and WRR after sending 20.000 request to 3 servers. It Shows that IP Hash cause imbalance in 3 of the servers. Server 1 receives fair load, server 2 receive lower load, while server 3 receive higher load than other 2 server. When using WRR algorithm it shows that WRR receive equal load and able to mitigate load imbalance cause by IP Hash algorithm.

Result of request distribution of IP Hash and WRR for 40.000, 60.000, 80.000 and 100.000 request can be seen in attachmet section. The result shows a similar pattern as discussed before; the result shows that IP Hash cause imbalance in some servers while WRR distributes requests to the 3 servers equally.

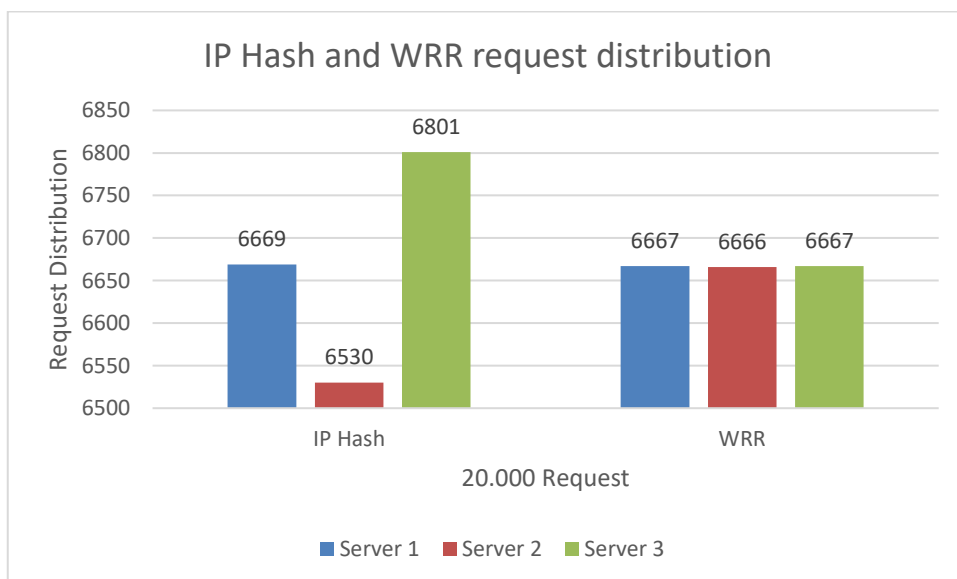


Figure 15 IP Hash and WRR 20.000 request distribution result

4.5 Jain fairness index IP Hash and WRR

Based on the request distribution discussed before, the fairness index can be calculated from request distributed to 3 servers. Index value for fairness index is between 0 to 1, 1 index value meaning that all servers receive request equally while below 1 meaning that some servers get underloaded or overloaded.

In figure 16 shows the fairness index value of IP Hash for each request. IP Hash index is below 1 because some servers receive unequal requests.

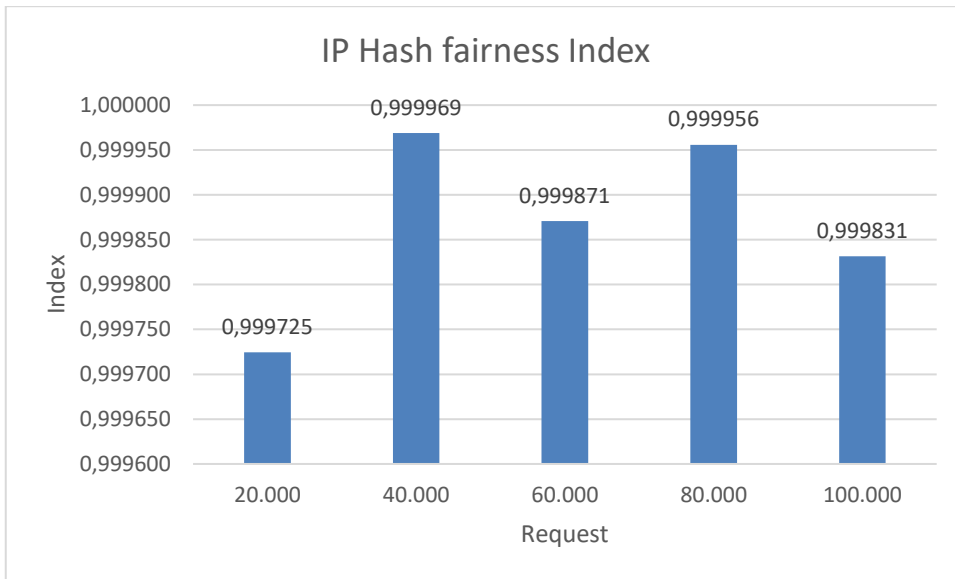


Figure 16 IP Hash fairness index result

The fairness index value for WRR is 1 for all the requests as shown in figure, this is because WRR can mitigate load imbalances and distribute request equally for every server.

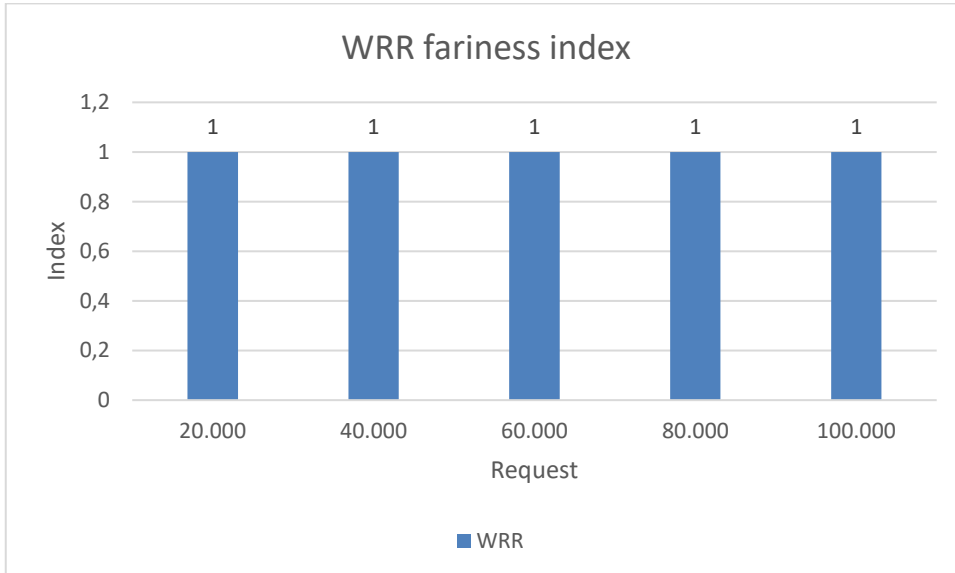


Figure 17 WRR fairness index result

5. Conclusion and suggestion

5.1 Conclusion

The load balancing algorithm was performed on multiple servers. The results showed that the WRR algorithm performed better in throughput parameter, with a 13% higher percentage than the IP Hash. This is because WRR uniformly distributes the requests across servers, preventing overloading in server and give optimal performance in handling requests. The response time performance of the IP Hash algorithm shows 2% less response time compared to the WRR algorithm. Packet loss is also analyzed, and the result shows that the IP Hash algorithm caused imbalance in some servers, while the WRR algorithm distributed requests equally to all servers. And the last test parameter is fairness index, for the IP Hash the value is below 1 and is WRR algorithms is 1 for all requests, meaning that WRR was able to mitigate load imbalances and distribute requests equally for all servers.

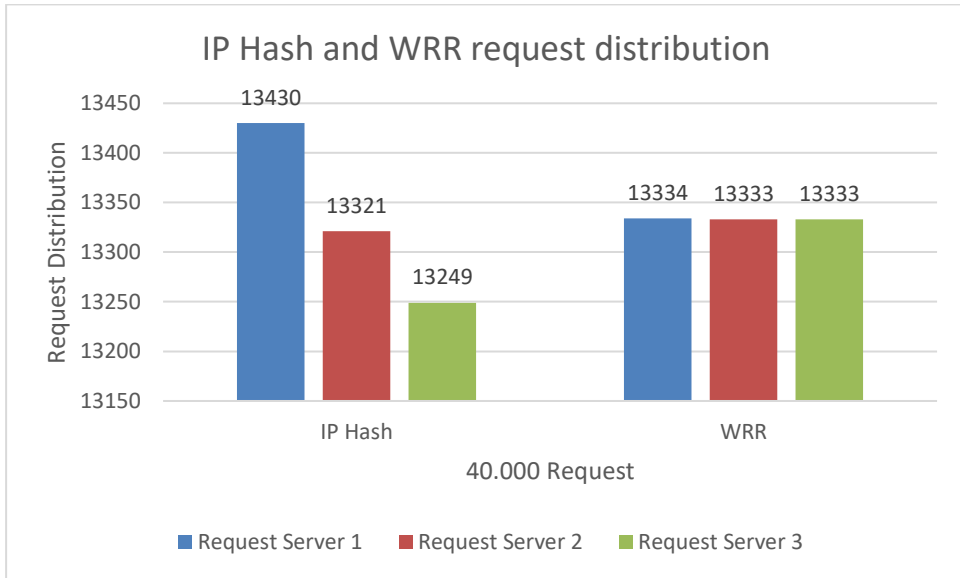
Daftar Pustaka

- [1] “What is Layer 4 Load Balancing? | Avi Networks.” Accessed: Sep. 08, 2024. [Online]. Available: <https://avinetworks.com/glossary/layer-4-load-balancing/>
- [2] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, “An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends,” *IEEE Access*, vol. 9, pp. 87094–87155, 2021, doi: 10.1109/ACCESS.2021.3086704.
- [3] T. Barbette, E. Wu, D. Kostic, G. Q. Maguire, P. Papadimitratos, and M. Chiesa, “Cheetah: A High-Speed Programmable Load-Balancer Framework With Guaranteed Per-Connection-Consistency,” *IEEEACM Trans. Netw.*, vol. 30, no. 1, Art. no. 1, Feb. 2022, doi: 10.1109/TNET.2021.3113370.
- [4] S. M. Hosseini, A. H. Jahangir, and S. Daraby, “Session-persistent Load Balancing for Clustered Web Servers without Acting as a Reverse-proxy,” in *2021 17th International Conference on Network and Service Management (CNSM)*, Izmir, Turkey: IEEE, Oct. 2021, pp. 360–364. doi: 10.23919/CNSM52442.2021.9615592.
- [5] M. S. Almhanna, T. A. Murshedi, F. S. Al-Turaihi, R. M. Almuttairi, and R. Wankar, “Dynamic Weight Assignment with Least Connection Approach for Enhanced Load Balancing in Distributed Systems,” Aug. 07, 2023. doi: 10.21203/rs.3.rs-3216549/v1.
- [6] “Persistence methods available in F5 BIG-IP.” Accessed: Aug. 24, 2024. [Online]. Available: <https://my.f5.com/manage/s/article/K26898044>
- [7] “Using nginx as HTTP load balancer.” Accessed: Jun. 08, 2023. [Online]. Available: http://nginx.org/en/docs/http/load_balancing.html
- [8] I. P. A. Suwandika, M. A. Nugroho, and M. Abdurahman, “Increasing SDN Network Performance Using Load Balancing Scheme on Web Server,” in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, Bandung: IEEE, May 2018, pp. 459–463. doi: 10.1109/ICoICT.2018.8528803.
- [9] S.-J. Hsu, C.-H. Ke, Y.-S. Chen, C.-F. Hung, and Y.-W. Lo, “Design and Performance Evaluation of a P4 based Load Balancer,” in *2019 8th International Conference on Innovation, Communication and Engineering (ICICE)*, Zhengzhou, Henan Province, China: IEEE, Oct. 2019, pp. 149–152. doi: 10.1109/ICICE49024.2019.9117548.
- [10] “Using nginx as HTTP load balancer.” Accessed: Aug. 27, 2024. [Online]. Available: https://nginx.org/en/docs/http/load_balancing.html
- [11] O. Michel, R. Bifulco, G. Rétvári, and S. Schmid, “The Programmable Data Plane: Abstractions, Architectures, Algorithms, and Applications,” *ACM Comput. Surv.*, vol. 54, no. 4, Art. no. 4, May 2022, doi: 10.1145/3447868.
- [12] “P4 – Language Consortium.” Accessed: Aug. 26, 2024. [Online]. Available: <https://p4.org/>
- [13] G. S. Amru, S. Prabowo, and M. A. Nugroho, “Analisis Performansi Load Balancing menggunakan Algoritma Round Robin dan Weighted Round Robin pada P4-Programmable Switch,” Feb. 2022.
- [14] M. R. Baihaqi, R. M. Negara, and R. Tulloh, “Analysis of Load Balancing Performance using Round Robin and IP Hash Algorithm on P4,” in *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Dec. 2022, pp. 93–98. doi: 10.1109/ISRITI56927.2022.10052975.
- [15] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, “SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, Los Angeles CA USA: ACM, Aug. 2017, pp. 15–28. doi: 10.1145/3098822.3098824.
- [16] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, “SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, Los Angeles CA USA: ACM, Aug. 2017, pp. 15–28. doi: 10.1145/3098822.3098824.
- [17] V. Olteanu, A. Agache, A. Voinescu, and C. Raiciu, “Stateless Datacenter Load-balancing with Beamer”.

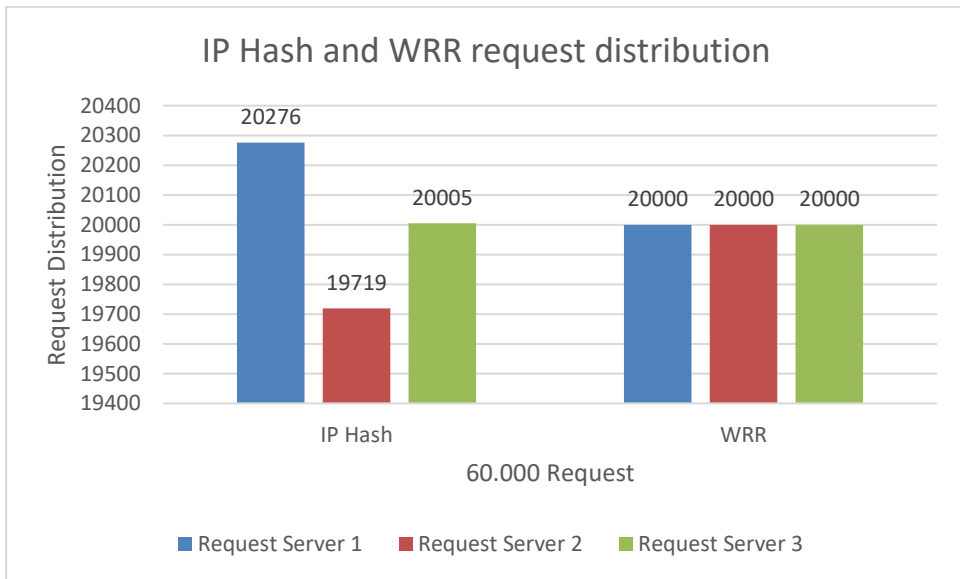
- [18] J. A. Brito, J. I. Moreno, L. M. Contreras, M. Alvarez-Campana, and M. Blanco Caamaño, “Programmable Data Plane Applications in 5G and Beyond Architectures: A Systematic Review,” *Sensors*, vol. 23, no. 15, p. 6955, Aug. 2023, doi: 10.3390/s23156955.
- [19] S. Kaur, K. Kumar, and N. Aggarwal, “A review on P4-Programmable data planes: Architecture, research efforts, and future directions,” *Comput. Commun.*, vol. 170, pp. 109–129, Mar. 2021, doi: 10.1016/j.comcom.2021.01.027.
- [20] “Comparing Layer 4, Layer 7, and GSLB techniques.” Accessed: Sep. 10, 2024. [Online]. Available: <https://www.loadbalancer.org/blog/comparing-layer-4-layer-7-and-gslb-load-balancing-techniques/>
- [21] “jafingerhut/p4-guide: Guide to p4lang repositories and some other public info about P4.” Accessed: Aug. 09, 2024. [Online]. Available: <https://github.com/jafingerhut/p4-guide>
- [23] “nsg-ethz/p4-utils: Extension to Mininet that makes P4 networks easier to build.” Accessed: Aug. 09, 2024. [Online]. Available: <https://github.com/nsg-ethz/p4-utils>
- [24] *p4lang/p4c*. (Aug. 08, 2024). C++. p4language. Accessed: Aug. 09, 2024. [Online]. Available: <https://github.com/p4lang/p4c>
- [25] “Y.1541 : Network performance objectives for IP-based services.” Accessed: Sep. 10, 2024. [Online]. Available: <https://www.itu.int/rec/T-REC-Y.1541>
- [26] R. Jain, A. Duresi, and G. Babic, “Throughput Fairness Index: Throughput Fairness Index: An An Explanation Explanation”.

Attachment

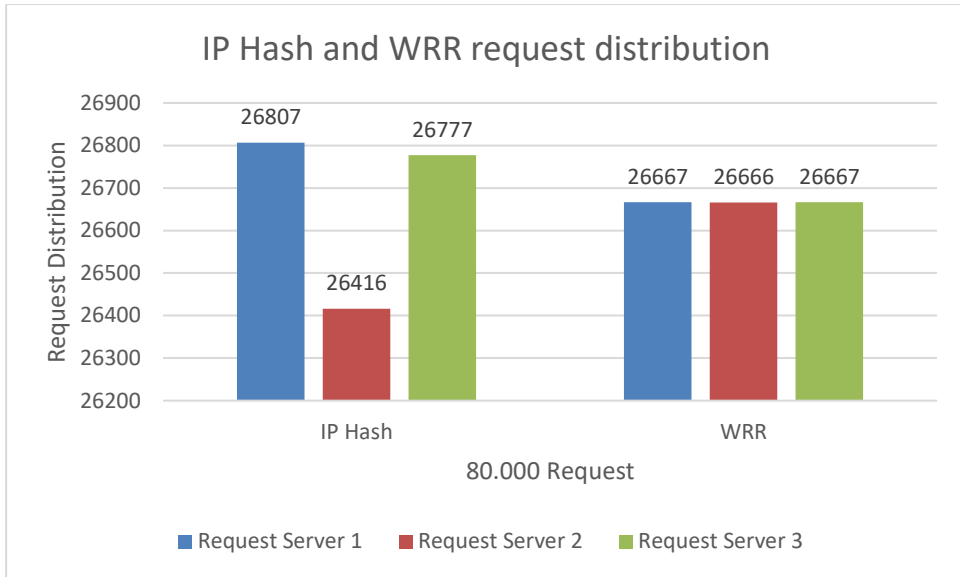
1. IP Hash and WRR 40.000 request distribution



2. IP Hash and WRR 60.000 request distribution



3. IP Hash and WRR 80.000 request distribution



4. IP Hash and WRR 100.000 request distribution

