SEMINAR NASIONAL TEKNOKA

Vol. 7, 2022
ISSN No. **2502-8782**

# Implementasi Load Balancing pada Local Traffic Manager Menggunakan Algoritma Round Robin dan Pengaruhnya Terhadap Performansi Jaringan

**Fadli Sirait[1)], Malia[2)], Akhmad Wahyu Dani[3)], & M.Ainur Rofiq[4)]**
[1,2,3,4]Universitas Mercu Buana, Jl. Meruya Selatan No. 1, Kembangan, Jakarta Barat, Telp +62(21) 5840816
Website: https://www.mercubuana.ac.id, E-mail: fadli.sirait@mercubuana.ac.id

***Abstrak***

*Pertumbuhan trafik yang semakin meningkat menyebabkan beban server yang semakin meningkat. Peningkatan beban server berakibat terjadinya overload pada server. Implementasi load balancing merupakan metode yang dapat digunakan untuk mengatasi permasalahan overload pada server. Pada penelitian ini round robin adalah algoritma load balancing yang digunakan, yang diimplementasikan pada F5 Big-IP sebagai load balancer. Selain F5 Big IP, juga terdapat Debian sebagai server, ubuntu sebagai client yang sudah terinstall httperf yang berfungsi untuk melakukan request http secara banyak dalam satu waktu. Sementara itu, performansi jaringan dapat ditentukan dari parameter Throughput, Round trip time, Packet loss, dan Delay. Dengan merujuk hasil eksperiment, sistem load balancing yang dirancang mampu melakukan pemerataan beban request atau beban permintaan kearah server. Sementara itu, penggunaan dengan penambahan jumlah server juga dapat meningkatkan throughput sebesar 1.195%, mengurangi packet loss 0.986%, mengurangi delay 0.848%, dan RTT 0.859%.*

***Kata kunci**: load balancing, round robin, traffic manager, performansi, jaringan*

***Abstract***

*The load on the server increased dramatically due to the increase in traffic. As server load increases, server workloads become overloaded. Implementing load balancing into the system is one approach that may be taken to deal with this issue. The load balancing algorithm employed in this study is round robin, which is implemented on the load balancer F5 Big-IP. In addition to F5 Big-IP, there is a debian that serves as a server, and Ubuntu that acts as a client and has httperf installed, allowing it to request HTTP in a massively at once. Throughput, Round trip time, Packet loss, and Delay are the parameters that are employed. Referred on the experiment results, it is clear that the developed system is capable of distributing the request load to the server. In the meantime, adding more servers can improve throughput by 1.195%, reduce packet loss by 0.986%, minimize the response time (RTT), and decrease delay by 0.848%.*

***Keywords**: load balancing, round robin, traffic manager, performance, network*

## 1 PENDAHULUAN

Secara global pertumbuhan pengguna internet diproyeksikan tumbuh dari 3.9 miliar pada tahun 2018 menjadi 5.3 miliar ditahun 2023, terlihat adanya kenaikan 0.3 miliar pengguna setiap tahunnya [1]. Pertumbuhan *traffic* yang begitu pesat berdampak terhadap peningkatan *work load* pada server. Dengan demikian diperlukan sebuah perancangan yang tepat dan dapat diandalkan untuk dapat mengatasi permasalahan ini, sehingga diperoleh kualitas jaringan yang dapat memenuhi kualitas yang diperlukan oleh pengguna. Pengelolaan *traffic* kearah *server* yang tidak efektif akan menyebabkan salah satu server mengalami beban kerja yang berlebihan. Salah satu solusi efektif yang memungkinkan untuk diimplementasikan terhadap permasalahan tersebut adalah dengan adanya pendistribusian beban kinerja menggunakan konsep *load balancing*.

Edison & Juan et al [2] mengimplementasikan *load balancing* yang berbasiskan round robin pada sebuah *prototype* jaringan nirkabel institusi. Implementasi yang diusulkan menunjukkan peningkatan nilai *throughput* dibandingkan dengan skenario convensional. Rifaldi & Helmi [3] menggunakan *load balancing* untuk menyeimbangkan pembagian *work load* ke *server* dengan menggunakan algoritma *round robin* pada *local traffic manager*. Andrey et al [4] menyatakan bahwa implementasi *load balancing* diperlukan untuk mendesain sebuah sistem operasi yang efisien. Pada penelitian dilakukan implementasi *load balancing* pada *local traffic manager* dan dilakukan pengukuran terhadap parameter performansi pada jaringan yaitu berupa *throughput, round trip time, packet loss*, dan *delay*.

## 2 LANDASAN TEORI

### 2.1 *Load Balancing*

*Load balancing* merupakan sebuah metode yang berguna untuk pembagian *traffic load* pada banyak jalur koneksi secara berimbang, sehingga dapat membuat *traffic* dapat berjalan secara maksimal.

*Dipresentasikan pada Tanggal 3 Desember 2022*
*Copyright © 2022 FTII - UHAMKA. - All rights reserved*

Terutama terkait dengan nilai *throughput*, meminimalisir *time respond* dan menghindari terjadinya *overload*. Pada saat jumlah pengguna pada sebuah server telah melebihi jumlah maksimal dari kapasitas yang sudah ditentukan maka pada saat itulah *load balancing* dapat digunakan [5]. Layanan *load balancing* menyediakan akses ke sumber daya jaringan yang didistribusikan di antara beberapa *host* lain, sehingga tidak terpusat, memungkinkan seluruh jaringan komputer beroperasi dengan stabil. Dengan menyebarkan pekerjaan secara merata, *load balancing* dapat meningkatkan daya tanggap aplikasi. Hal ini juga dapat meningkatkan ketersediaan aplikasi dan situs web bagi pengguna.

## 2.2 *Quality of Service (QoS)*

*Quality of Service* merupakan hasil kolektif dari berbagai parameter performansi yang menjadi penentu tingkat kepuasan dari para pengguna terhadap suatu layanan [5]. Studi terhadap QoS dilakukan dengan tujuan untuk melakukan optimalisasi terhadap kapasitas jaringan terhadap berbagai jenis *service*, tanpa melakukan investasi yang besar dengan harus melakukan penambahan infrastruktur jaringan. Sementara itu, terdapat berbagai jenis aplikasi yang juga memiliki berbagai jenis kebutuhan yang berbeda. Salah satu yang dapat dijadikan contoh adalah *data transaction* mempunyai karakter yang sensitif terhadap *distortion* akan tetapi kurang sensitif terhadap *delay*.

Adapun beberapa hal yang dapat dijadikan acuan sebagai unjuk kerja pada sebuah jaringan adalah sebagai berikut:

1. *Throughput*
   *Throughput* adalah jumlah total paket yang berhasil terdeteksi di suatu tujuan selama periode tertentu dibagi dengan durasi periode tersebut [6].

***Tabel 1*** Standarisasi *Throughput*

| ***Kategori*** | ***Throughput (%)*** |
|---|---|
| *Sangat Bagus* | *76 - 100* |
| *Bagus* | *51 - 75* |
| *Sedang* | *25 - 50* |
| *Buruk* | *<25* |

2. *Delay*
   Delay adalah total waktu yang dibutuhkan paket untuk melakukan perjalanan melalui jaringan dari pengirim ke penerima [6]. *Delay* dari pengirim ke penerima pada dasarnya terdiri dari *hardware delay*, *access delay*, dan *transmission delay*. *Delay* yang paling umum untuk sebuah *traffic* adalah t*ransmission delay*.

***Tabel 2*** Standarisasi *Delay*

| ***Kategori Delay*** | ***Delay*** |
|---|---|
| *Sangat Bagus* | *<150 ms* |
| *Bagus* | *150-300 ms* |
| *Sedang* | *300-450 ms* |
| *Buruk* | *>450 ms* |

3. *Packet loss*
   *Packet loss* adalah parameter yang menggambarkan kondisi yang menunjukkan jumlah paket yang hilang. Hal tersebut dapat terjadi disebapkan oleh *collision* dan *congestion* pada jaringan, dan hal ini mempengaruhi semua aplikasi karena pengiriman ulang mengurangi efisiensi jaringan secara keseluruhan bahkan mesikipun bandwidth cukup tersedia untuk aplikasi tersebut [6].

***Tabel 3*** Standarisasi *Packet Loss*

| ***Kategori Packet Loss*** | ***Packet Loss(%)*** | ***Indeks*** |
|---|---|---|
| *Sangat Bagus* | *0* | *4* |
| *Bagus* | *3* | *3* |
| *Sedang* | *15* | *2* |
| *Buruk* | *25* | *1* |

## 2.3 *Perangkat Load Balancer*

Penelitian ini menggunakan perangkat yang berfungsi sebagai *load balancer* yang dapat berfungsi sebagai *Application Delivery Network* (AND) dan layanan aplikasi. Teknologi pada load balancer yang digunakan berfokus pada keamanan, pengiriman, kinerja, dan ketersediaan aplikasi web. Salah satu module yang paling popular yang ditawarkan pada *platform load balancer* yang digunakan adalah *local traffic manager*. Kekuatan sebenarnya dari *local traffic manager* (LTM) adalah *FullProfxy*, untuk meningkatkan koneksi sisi *client* dan *server*, selain itu dapat juga membuat keputusan *load balance* pada *performance*, *availability*, dan persistensi. Di LTM menunjukkan bahwa, biasanya server ditempatkan pada data *centre* yang sama dengan menggunakan *load balance*.

## 2.4 *Round Robin*

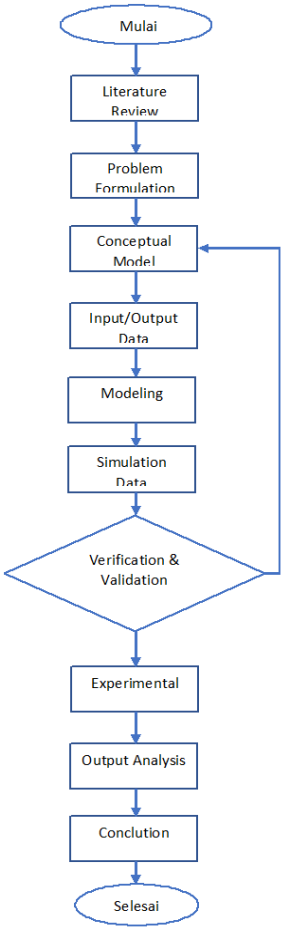
Algoritma *Round Robin* merupakan algoritma yang paling sederhana dan sering dimanfaatkan pada sebuah perangkat *load balancing*. Algoritma ini mendistribusikan secara bergantian dan berurutan dari server ke server, membentuk putaran [7].

*Copyright © 2022 FTII-UHAMKA. - All rights reserved*


*ISSN No. 2502-8782*

## 3 METODOLOGI PENELITIAN

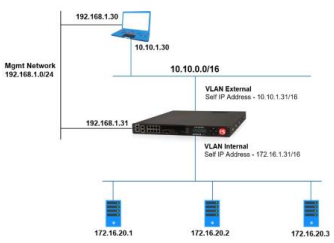
Bagan alur metodologi penelitian yang digunakan dalam penelitian ditunjukkan pada Gambar 1.




***Gambar 1*** *Diagram alir penelitian*

Pada penelitian ini, *load balancing* digunakan dalam perancangan sistem yang dibangun untuk server. Adapun algoritma yang digunakan untuk konsep load balancing pada sistem ini adalah dengan perangkat F5 Big-IP LTM sebagai medianya. Untuk web server yang digunakan adalah linux Debian, dan linux ubuntu sebagai client yang didalamnya sudah terinstall *httpef.*

1. Topologi
   Topologi yang digunakan dalam penelitian ini ditunjukkan pada Gambar 1.




***Gambar 2*** *Topologi jaringan yang digunakan*

Gambar 1 memperlihatkan topologi yang diimplementasikan adalah jenis topologi star, dimana semua koneksi berpusat pada Big IP. Pada topologi ini, digunakan tiga buah *server* untuk menjawab request dari client. Pada penelitian ini, hanya digunakan satu buah *client* yang berfungsi untuk melakukan *request.* Sementara itu, big ip merupakan *device* yang berfungsi untuk melakukan *load balancing* kearah *server*.

2. Konfigurasi
   Ada beberapa hal yang perlu dilakukan saat mengatur load balancing untuk F5, yaitu dengan menambahkan node (alamat IP perangkat server) ke IP Big F5. Langkah selanjutnya adalah dengan membuat *server pool* baru dengan node yang ditambahkan sebelumnya.

```
[root@labTAMalia:Active:Standalone] config # tmsh list ltm node
ltm node 172.16.20.1 {
    address 172.16.20.1
}
ltm node 172.16.20.2 {
    address 172.16.20.2
}
ltm node 172.16.20.3 {
    address 172.16.20.3
}
```

***Gambar 3*** *Node Configuration*

Selanjutnya, server virtual dibuat berisi *server pool* yang dibuat pada Langkah sebelumnya.

```
[root@labTAMalia:Active:Standalone] config # tmsh list ltm node
ltm node 172.16.20.1 {
    address 172.16.20.1
}
ltm node 172.16.20.2 {
    address 172.16.20.2
}
ltm node 172.16.20.3 {
    address 172.16.20.3
}
```

***Gambar 4*** *Node Configuration*

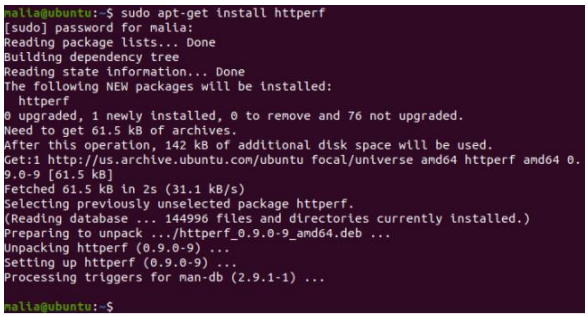
Dan yang terakhir dilakukan adalah dengan melakukan request dari *client* untuk melihat hasilnya.

```
[root@labTAMalia:Active:Standalone] config # tmsh list ltm virtual
ltm virtual vs_http {
    destination 10.10.100.1:http
    ip-protocol tcp
    mask 255.255.255.255
    pool http_pool
    profiles {
        tcp { }
    }
    source 0.0.0.0/0
    translate-address enabled
    translate-port enabled
    vs-index 2
}
```

***Gambar 5*** *Virtual Server Configuration*

*Copyright © 2022 FTII-UHAMKA. - All rights reserved*

Selain konfigurasi disisi big ip, dilakukan juga instalasi *software* httper di *client*, yang berfungsi untuk melakukan *request* secara banyak kearah server. Gambar 5 menunjukkan proses dari instalasi *software* httperf di sisi client.



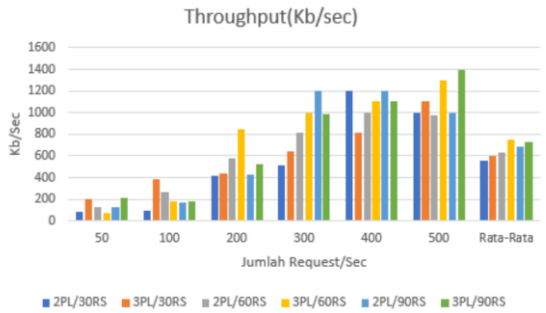

***Gambar 6*** Proses instalasi *httperf*

## 4 HASIL DAN PEMBAHASAN

Pada penelitian dilakukan beberapa skenario pengumpulan data dan dibandingkan dengan standar performansi jaringan menurut standard thypoon. Parameter yang diukur adalah *throughput*, *delay*, RTT, dan *packet loss*. Selain itu, juga dilakukan pengukuran terhadap *laoad balancing* dan *high availability*.

### *4.1 Throughput*

Parameter *throughput* dalam penelitian ini mewakili banyaknya jumlah permintaan yang direspon oleh *webserver* dalam waktu tertentu.




***Gambar 7*** *Pengujian throughput dengan variasi pool member dan request persession*

Gambar 7 menunjukkan hasil pengujian nilai *throughput* dengan menggunakan dua jenis *pool member* yaitu 2 dan 3 *pool member* (2 dan 3 PL). Sementara itu jumlah *request persession* yang dilakukan adalah 30, 60, dan 90 *request persession* (30,60, dan 90RS).

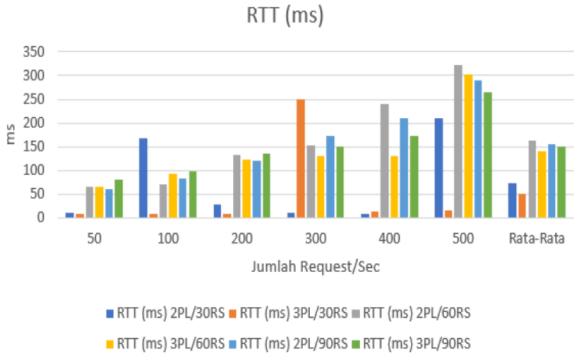
Gambar 7 menunjukkan bahwa semakin tinggi jumlah permintaan per detik, semakin baik kinerjanya. Hal ini terjadi karena semakin banyak *packet* yang dikirim persecondnya maka waktu pemrosesannya juga akan semakin sedikit.

Selain itu, Gambar 7 juga menunjukkan perbandingan antara nilai *throughput* dengan menggunakan 2 *pool member* dan 3 *pool member*. Kenaikan nilai *throughput* untuk setiap *request* yang dilakukan cukup signifikan, baik dengan menggunakan 2 *pool member* ataupun dengan 3 *pool member*. Secara rata-rata, nilai *throughput* dengan menggunakan 3 *pool member* adalah lebih baik apabila dibandingkan dengan menggunakan 2 *pool member*.

Implementasi *load balancing server* dengan menggunakan *pool member* yang berbeda juga dapat meningkatkan *throughput*, hal ini dapat dilihat pada rata-rata *throughput* dengan menggunakan 3 *pool member* adalah lebih bagus dibandingkan dengan menggunakan 2 *pool member*. Hal ini menunjukkan bahwa penambahan jumlah *server* berdampak terhadap peningkatan nilai *throughput load balancing server*. Dengan menambahkan jumlah *pool member* ke arah *server* dapat meningkatkan nilai *thtoughput* sebesar 1.195%.

### 4.2 RTT

*Round-trip time* (RTT) adalah waktu yang dibutuhkan sebuah paket untuk sampai dari klien ke server.




***Gambar 8*** *Pengujian RTT dengan variasi pool member dan request persession*

Gambar 8 menunjukkan hasil pengujian nilai *throughput* dengan menggunakan dua jenis *pool member* yaitu 2 dan 3 *pool member* (2 dan 3 PL). Sementara itu jumlah *request persession* yang dilakukan adalah 30, 60, dan 90 *request persession* (30,60, dan 90RS).

Dari Gambar 8, dapat dilihat perbandingan nilai RTT yang diperoleh dari setiap *request* dengan menggunakan 2 *pool member* maupun dengan menggunakan 3 *pool member*. Pada umumnya nilai RTT yang paling tinggi diperoleh Ketika jumlah *request* 500. Sementara itu, nilai RTT yang paling rendah diperoleh Ketika jumlah requestnya adalah 50, baik dengan menggunakan 2 *pool member* ataupun dengan menggunakan 3 *pool member*. Kondisi ini menunjukkan bahwa semakin banyak permintaan yang dibuat per detik, semakin tinggi nilai RTT. Sebaliknya, semakin sedikit permintaan per detik,

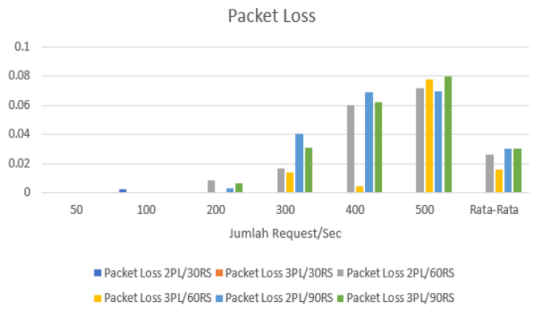
*Copyright © 2022 FTII-UHAMKA. - All rights reserved*

semakin rendah nilai RTT. Ini berkaitan dengan fakta bahwa semakin banyak paket yang dikirim per detik, semakin banyak sesi yang harus ditangani oleh server. Hal ini menyebabkan nilai RTT yang lebih tinggi.

Implementasi *load balancing* server dengan menggunakan *pool member* yang berbeda juga dapat mengurai RTT dilevel aplikasi, hal ini dapat diliat dengan rata-rata RTT dengan menggunakan 3 *pool member* adalah lebih baik apabila dibandingkan dengan menggunakan 2 *pool member*. Hal ini menunjukkan bahwa penambahan jumlah *server* berdampak terhadap penurunan nilai RTT. Penambahan jumlah *pool member* ke arah server dapat mengurangi nilai RTT sebesar 0.859%.

### *4.3 Packet Loss*

*Packet loss* adalah jumlah paket yang hilang selama proses transmisi dari sumber ke tujuan.

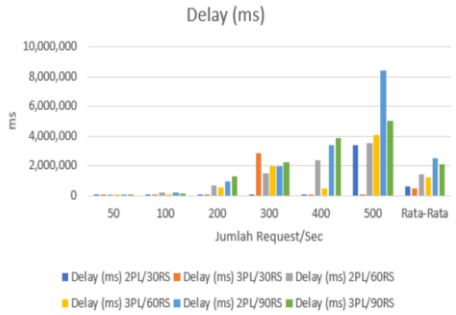



***Gambar 9*** *Pengujian Packet loss dengan variasi pool member dan request persession*

Gambar 9 menunjukkan perbandingan *packet loss* pada 2 *pool member* dan 3 *pool member* sehingga dapat disimpulkan semakin banyak *session* persecond, semakin tinggi *packet loss.* Demikian juga pada saat semakin sedikit *session persecond* maka semakin kecil *packet loss* yang terjadi. Dengan kata lain, dapat dinyatakan dinyatakan bahwa nilai *packet loss* lebih tinggi ketika menggunakan 2 *pool member* apabila dibandingkan dengan menggunakan 3 *pool member*. Dengan menambahkan jumlah *pool member* ke arah *server*, maka nilai *packet loss* dapat berkurang sebesar 0.986%, dimana hal ini dapat dikatakan cukup signifikan dengan nilai *packet loss* yang kecil.

### *4.4 Delay*

*Delay* adalah penundaan waktu untuk proses selama data dikirim dari titik sumber ke titik tujuan.




***Gambar 10*** *Pengujian delay dengan variasi pool member dan request persession*

Gambar 10 menunjukkan perbandingan *delay* pada 2 *pool member* dan 3 *pool member*, sehingga dapat disimpulkan semakin banyak *session persecond* yang terjadi, semakin tinggi *packet loss.* *Packet loss* yang paling rendah terjadi pada saat session persecond semakin kecil. Selain itu, dari Gambar 10 dapat juga dilihat bahwa nilai *packet loss* lebih tinggi pada saat menggunakan 2 *pool member* dibandingkan Ketika menggunakan 3 *pool member*.

Implementasi *load balancing server* dengan menggunakan jumlah *pool member* yang berbeda dapat mengurang *delay* dilevel aplikasi. Hal ini tercermin dari rata-rata jumlah *delay* yang menggunakan 3 *pool member* lebih baik dibandingkan dengan menggunakan 2 *pool member*. Dengan demikian dapat disimpulkan bahwa penambahan jumlah *sever* dapat menurunkan jumlah *delay*. Dengan menambahkan jumlah *pool member* ke arah *server* dapat mengurangi *delay* sebesar 0.0848%.

### *4.5 Load Balancing*

Untuk memastikan *load balancing* berfungsi maka dilakukan pengujian dengan melakukan *request* http ke arah *server* dan melihat statistik dari *pool* big ip.

***Tabel 4*** *Pengujian load balancing dengan variasi pool member dan request persession*

| ***Connection /Request*** | ***Connection PL1*** | ***Connection PL2*** | ***Connection PL3*** | ***Total Connection/Request*** |
|---|---|---|---|---|
| *100* | *33* | *34* | *33* | *100* |
| *200* | *67* | *66* | *67* | *200* |
| *300* | *100* | *100* | *100* | *300* |
| *400* | *133* | *134* | *133* | *400* |
| *500* | *167* | *166* | *167* | *500* |
| *Total Connection/ Server* | *500* | *500* | *500* | *1500* |

Dari tabel 4, terlihat bahwa setiap *connection* yang datang dari *client* selalu dibagi sama rata kesetiap *server*. Dengan *request* 100, terdapat 33 *connection* di *server* pertama, 34 *connection* di server kedua, dan 33

*Copyright © 2022 FTII-UHAMKA. - All rights reserved*

*connection* di *server* ketiga. Jika dijumlahkan, *connection* yang dilayani oleh masing-masing server maka didapat totalnya adalah 100 *connection*. Setelah dilakukan pengujian dengan beberapa jumlah *connection*, diperoleh total semua *connection* adalah 1500, dengan *connection* yang dilayani di setiap *server* adalah 500 *connection*. dari pengujian ini, dapat disimpulkan bahwa untuk *load balancing* dengan algoritma round robin sudah berjalan sesuai dengan yang seharusnya, yaitu adanya pemerataan beban pada beberapa server.

### *4.6 High Availability*

*High availability* (*HA*) adalah sebuah konsep pada infrastruktur yang dapat menjamin pelayanan *server* tetap tersedia. Untuk mengetahui *server* tersebut aktif atau tidak, di Big ip dapat menggunakan *healty check*, dimana big ip akan melakukan ping secara periodik ke ip *server* tersebut. Ketika ip tersebut tidak bisa diping dari big ip dengan rentang waktu tertentu, maka big ip akan menandai *server* tersebut *down*. Dan *traffic* tidak di*forward* lagi ke *server* tersebut. Untuk data pengujian *high availability* ada di tabel 5.

***Tabel 5*** *Pengujian High Availability*

| ***IP Pool*** | ***IP Server 1*** | ***IP Server 2*** | ***IP Server 3*** | ***Status Pool*** |
|---|---|---|---|---|
| | *172.16.20.1* | *172.16.20.2* | *172.16.20.3* | *Semua Pool UP* |
| | *172.16.20.5* | *172.16.20.2* | *172.16.20.3* | *Pool Server 1 Down, Pool Server 2&3 UP* |
| *172.16.20.1 172.16.20.2 172.16.20.3* | *172.16.20.1* | *172.16.20.4* | *172.16.20.3* | *Pool Server 2 Down, Pool Server 1&3 UP* |
| | *172.16.20.1* | *172.16.20.2* | *172.16.20.5* | *Pool Server 3 Down, Pool Server 1&2 UP* |
| | *172.16.20.4* | *172.16.20.2* | *172.16.20.5* | *Pool Server 1&3 Down, Pool Server 2 UP* |

Dari data pada tabel 5, terlihat bahwa ketika salah satu ip di *server* dirubah, maka dari big-ip akan memberi tanda bahwa *server* tersebut adalah *down* dan tidak bisa melayani trafik yang datang dari *subscriber*. Dari data tabel testing yang dilakukan, dapat dinyatakan *high availability* telah bekerja, yaitu ketika salah satu ip di *server* berubah. Terlihat bahwa status *pool member node* ketiga *down*. Dan ketika dilakukan *request* http dari *client*, *node* yang telah diberi tanda tidak lagi menerima trafik dari *client*, dan *trafic* hanya dilayani oleh *pool member* yang statusnya adalah aktif.

## 5 SIMPULAN

Implementasi *load balancing* pada *local traffic* dengan menggunakan algoritma *round robin* memberikan dampak yang cukup signifikan pada performansi jaringan, diindikasikan dengan adanya penambahan jumlah *pool member* dapat menaikkan nilai *throughput* sebesar 1.195%, dan mengurangi *packet loss* sebesar 0.986%, mengurangi *delay* sebesar 0.848%, dan pengurangan nilai RTT sebesar 0.859%. Sementara itu, dengan menggunakan *load balancing*, maka *availability* atau ketersediaan setiap server akan tetap terjaga. Sistem *load balancing* pada penelitian berjalan dengan baik, hal ini dapat dilihat dengan mengecilnya *overload* yang terjadi pada sisi *server* karena *request* dari *client* dibagi sama rata kes emua *server* yang aktif.

## KEPUSTAKAAN

[1] Cisco, “Cisco Annual Internet Report (2018–2023),” 2020. [Online]. Available: http://grs.cisco.com/grsx/cust/grsCustomerSurvey.html?SurveyCode=4153&ad_id=US-BN-SEC-M-CISCOASECURITYRPT-ENT&KeyCode=000112137.

[2] E. F. Aza and J. P. Urrea, “Implementation of Round-Robin load balancing scheme in a wireless software defined network,” *2019 IEEE Colomb. Conf. Commun. Comput. COLCOM 2019 - Proc.*, pp. 1–6, 2019, doi: 10.1109/ColComCon.2019.8809180.

[3] R. A. Putra and H. F. Lutfi, “Implementasi Load Balancing Algoritma Round Robin Pada F5 Big-IP Local Traffic Management,” *Teknologi*, no. December, 2019, [Online]. Available: https://www.researchgate.net/profile/Rifaldi-Putra-2/publication/338187273_Implementasi_Load_Balancing_Algoritma_Round_Robin_Pada_F5_Big-IP_Local_Traffic_Manager/links/5e0585264585159aa49d457b/Implementasi-Load-Balancing-Algoritma-Round-Robin-Pada-F5-Big-.

[4] A. Chernov, Y. Ivanskiy, I. Len, and N. Amelina, “Network Traffic Load Balancing Protocol for Different Priority Traffic,” 2022, doi: 10.1109/DCNA56428.2022.9923119.

[5] A. Rahmatullah and F. MSN, “Implementasi Load Balancing Web Server menggunakan

*Copyright © 2022 FTII-UHAMKA. - All rights reserved*

Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi," *J. Nas. Teknolodi Sist. Inf.*, vol. 3, no. 2, 2017.

[6] R. Wulandari, "ANALISIS QoS (QUALITY OF SERVICE) PADA JARINGAN INTERNET (STUDI KASUS : UPT LOKA UJI TEKNIK PENAMBANGAN JAMPANG KULON –LIPI)," *J. Tek. Inform. dan Sist. Inf.*, vol. 2, no. 2, 2016.

*Copyright © 2022 FTII-UHAMKA. - All rights reserved*