https://jurnaleeccis.ub.ac.id/
e-ISSN(Online): 2460-8122

Jurnal EECCIS Vol. 16, No. 1, April 2022
pp 15-19

Pembagian Beban Trafik pada *Cluster Server*

Mila Kusumawardani [1], Nugroho Suharto [1], M. Nanak Zakaria [1]
1st Teknik Telekomunikasi Jurusan Teknik Elektro, Politeknik Negeri Malang
Email: mila.kusumawardani@polinema.ac.id, sahoq@yahoo.co.id, nanak712000@gmail.com

***Abstract*— The problem with the server is the number of *user*s accessed at the same time. To overcome this, the concept of *cluster* server can be used. The method used in this article is to build 3 server *cluster*s and 1 *load balancer* from 4 Nanopi-2 and implement the round robin algorithm in coordination settings. The results obtained are that the distribution of traffic load is assessed between 3 servers with an average at the time the server receives *request*s from *client*s is 33.57%, 32.64%, and 33.8%, and the average at the time of the server gave *response*s were 33.47%, 34.14% and 35.35%**

***Index Terms*—*cluster server*, Nanopi-2, traffic load**

***Abstrak*–Permasalahan yang ada pada *server* adalah banyaknya *user* yang mengakses dalam waktu bersamaan. Untuk mengatasinya, konsep *cluster server* dapat digunakan. Metode yang digunakan dalam artikel ini adalah membangun 3 *cluster server web* dan 1 *load balancer* dari 4 buah Nanopi-2 serta mengimplementasikan algoritma *round robin* dalam pengaturan kerjanya. Hasil yang diperoleh adalah bahwa terjadi pembagian beban trafik yang berimbang di antara 3 *server* dengan rata-rata pada saat *server* menerima *request* dari *client* adalah 33.57%, 32.64%, dan 33.8%, serta rata-rata pada saat *server* memberikan *response* adalah 33.47%, 34.14%, dan 35.35%**

***Kata Kunci*— beban trafik, *cluster server*, Nanopi-2**

I. PENDAHULUAN

Kebutuhan komunikasi yang tinggi berdampak pada banyaknya jumlah *user* (*client*) yang mengakses suatu *server*. Banyaknya *user* berkaitan dengan kepadatan trafik yang terjadi. Kondisi ini dapat menimbulkan overload, karena secara perangkat CPU dan RAM dari *server* tidak mampu memberikan layanan terhadap semua trafik. Dari kondisi ini diperlukan penambahan kapasitas *server* yang salah satunya dengan menggunakan konsep *cluster server* dan load balancing. Melalui *cluster server* maka *server* tunggal dapat ditambah kapasitasnya oleh sejumlah *server*. Dengan dua atau lebih *server* yang saling berbagi beban lalu lintas, masing-masing akan berjalan lebih cepat karena beban tidak berada pada 1 *server* saja. Ini merupakan dampak positif karena ada lebih banyak sumber daya untuk memenuhi permintaan *client*.

Adanya sejumlah *server* dalam suatu *cluster* memerlukan pengaturan dalam sistem kerjanya. Untuk itu diperlukan pembagi beban (*load balancer*) yang bekerja berdasarkan suatu algoritma. Bentuk algoritma sederhana yang banyak digunakan adalah round robin yang membagi beban trafik secara bergilir dan berurutan dari satu *server* ke *server* lain sehingga membentuk putaran [1]. Round robin menggunakan *time slice* (waktu irisan) untuk mengeksekusi proses.

Pada *load balancer*, *request* dari *client* merupakan suatu proses. Misalkan pada sebuah *request* HTTP, jika terdapat lebih dari 1 *client* yang mengakses domain maka pada saat itu algoritma round robin berfungsi agar *load balancer* dapat mengirim *request* dari tiap *client* secara berurutan ke sejumlah *server* yang ada di bagian *back-end*.

Penelitian yang telah dilakukan [2] mengenai *cluster server* berbasis SBC (*Single Board Computer*), menggunakan 2 RaspberryPi dan sebuah *load balancer*. Penelitian lain [3] membandingkan kemampuan RaspberryPi3 dengan Intel *Rack Server* dalam hal waktu respon dan kecepatan transfer data.

Jenis SBC selain Raspberry Pi adalah Nanopi-2. Fitur yang dimiliki adalah prosesor Samsung S5P4418 Quad Core A9@1.4GHz, RAM DDR3 1G 32bit, antarmuka video dan tampilan yang baik, serta dua slot MicroSD. Nanopi-2 juga memiliki pin header GPIO Raspberry Pi yang mendukung modul GPIO eksternal Raspberry Pi dan Arduino. PCB-nya sudah dilengkapi dengan Chip Nirkabel dan Bluetooth AP6212 on-board sehingga mendukung *mode* Wi-Fi 802.11 b / g / n, mode AP, BLE 4.0 dan HS.

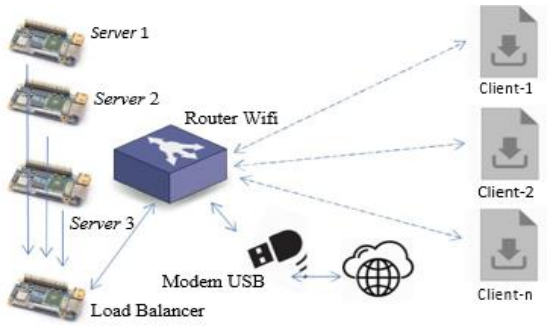
Penelitian ini membahas mengenai pembagian beban trafik pada *cluster server* dan *load balancer* berbasis Nanopi-2. *Server* yang dimaksudkan adalah *web server* yang akan diakses oleh sejumlah *client*. *Cluster server* terdiri dari 3 buah Nanopi-2 dan 1 *load balancer*. *Software* yang digunakan untuk mengimplementasikan Nanopi-2 sebagai *load balancer* adalah HAProxy, yang akan aktif jika file konfigurasi untuk *front-end* dan *back-end* telah diatur. *Front-end* digunakan untuk mengidentifikasi perangkat yang akan digunakan sebagai *load balancer* sedangkan *back-end* digunakan untuk mengidentifikasi perangkat *server* yang akan diatur oleh *load balancer*.

Untuk melakukan analisis terhadap kinerja jaringan digunakan Wireshark. Program ini dapat merekam semua paket yang lewat di jaringan dan menyeleksi serta menampilkan detail dari paket tersebut. Paket data yang dapat ditangkap misalnya adalah paket TCP, UDP, dan ARP. Dengan demikian dapat diketahui bahwa Wireshark terutama oleh administrator jaringan untuk mendapatkan informasi mengenai kondisi jaringan.

II. METODE PENELITIAN

A. Perencanaan Jaringan dan Identikasi Kebutuhan

Sebuah *cluster server* yang akan dibangun terdiri dari 3 *server* serta *load balancer* terhubung pada satu *router wifi*. *Load balancer* akan bekerja untuk mengatur beban trafik dari *server* dengan algoritma round robin menggunakan *software* HAProxy. Koneksi internet diperoleh dari modem USB menggunakan jaringan 3G. Topologi jaringan ditunjukkan dalam Gambar 1.



Gambar 1. Topologi jaringan

Berdasarkan perencanaan tersebut dilakukan identifikasi atas keperluan sistem, yaitu 4 buah Nanopi-2 dan *server* serta perlunya dilakukan instalasi web *server* dan HAProxy. Apabila HAProxy *load balancer* selesai diunduh dan di-*install* selanjutnya adalah menambahkan konfigurasi pada HAProxy agar dapat bekerja sesuai algoritma.

Algoritma round robin akan aktif ketika sudah ditetapkan pada bagian konfigurasi *back-end*. Sebab, IP pada setiap *server* ditentukan pula pada bagian konfigurasi tersebut. Dengan demikian, algoritma round robin dapat membaca IP *cluster server* untuk melakukan *scheduling*.

Proses *scheduling* algoritma dilakukan oleh *load balancer* dengan IP 210.155.43.2, dimulai dari *server* pertama, dengan IP 210.155.43.4 dan bergiliran ke *server* selanjutnya sesuai urutan pada konfigurasi bagian *back-end*. Selanjutnya IP address yang digunakan adalah IP statis versi 4 kelas C dan disetting untuk *router*, *load balancer*) dan 3 *server*. Rincian IP adalah 210.155.43.1 (sebagai IP *router* wifi), 210.155.43.2 (sebagai IP *load balancer)*, 210.155.43.4 (sebagai IP *server* 1), 210.155.43.6 (sebagai IP *server* 2), dan 210.155.43.8 (sebagai IP *server* 3).

B. Instalasi dan Konfigurasi

Setelah semua alat dipersiapkan, maka selanjutnya adalah instalasi web *server* dan instalasi *load balancer*. Jika *server* 1 telah dikerjakan, maka *server* perlu dites melalui IP host *server* 1 yakni 210.155.43.4 pada jaringan lokal. Cara termudah untuk mengetesnya adalah dengan mengakses IP tersebut melalui *web browser.*

C. Instalasi dan Konfigurasi

Setelah semua alat dipersiapkan, maka selanjutnya adalah instalasi web *server* dan instalasi *load balancer*. Jika *server* 1 telah dikerjakan, maka *server* perlu dites melalui IP host *server* 1 yakni 210.155.43.4 pada jaringan lokal. Cara termudah untuk mengetesnya adalah dengan mengakses IP tersebut melalui web *browser*.

Ketika semua web *server* pada *cluster server* telah siap, selanjutnya adalah memasang *load balancer*. *Software load balancer* yang digunakan adalah HAProxy. Untuk memastikan *load balancer* dan *cluster server* berhasil terkoneksi, kerja *load balancer* perlu dicek. Yakni dengan cara mengakses IP host *load balancer*, 210.155.43.2. Apabila muncul halaman yang sama dengan halaman yang diakses pada IP *host server*, maka jaringan ini siap digunakan.

D. Pengujian dan Analisis

Tahap selanjutnya adalah menguji kemampuan pembagian beban *load balancer*. Pengujian *load balancer* ini ditinjau melalui hasil session, menggunakan tabel statistik HAProxy. Apabila angka pada masing-masing *server* pada kolom "*session*" memiliki nilai "*max*" yang sama, maka beban trafik berhasil dibagi rata pada masing-masing *server*.

Sebagai bahan analisa cara kerja *load balancer*, maka diperlukan untuk mengetahui proses *routing* yang terjadi selama pengujian. Proses ini memerlukan. Wireshark sehingga dapat diketahui dan dihitung protokol yang muncul pada saat proses *routing*.

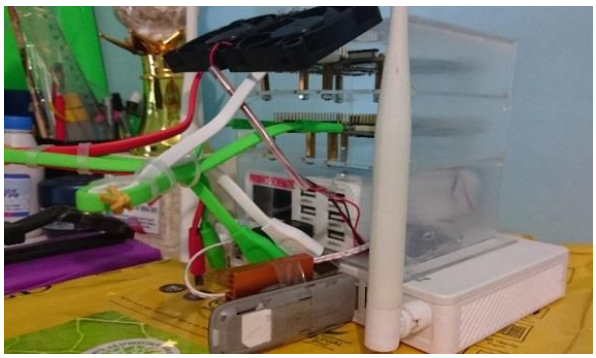
III. HASIL DAN PEMBAHASAN

A. Algoritma Round Robin pada HAProxy

Proses kerja *load balancing* dengan algoritma round robin pada HAProxy *load balancer* adalah sebagai berikut a) Sejumlah *client* mengirimkan *request* melalui URL yang diakses, b) *Load balancer* menerima *request* tersebut dari *router*, c) Algoritma round robin mengurutkan *request* sesuai arrival time sehingga membentuk antrian (queue), d) *Request* yang pertama kali datang disalurkan menuju *server* 1, *request* kedua menuju *server* 2 dan *request* ketiga menuju *server* 3, e) CPU pada masing masing *server* memproses *request* tersebut menghitung burst time pada satu *request*, f) Jika masih terdapat *request*, round robin akan menyalurkan *request* tersebut kembali sercara berurutan mulai dari *server* satu, sehinggak terjadi waiting time diantara proses-proses yang dikerjakan oleh *server, g)* Jika *server* yang dituju sibuk (*busy*) maka *request* selanjutnya dialihkan oleh round robin menuju ke *server* lain yang memiliki burst time kecil, atau yang sudah menyelesaikan satu proses, h) Jika semua *server* sibuk, maka *request* tersebut tetap disalurkan *load balancer* untuk diproses oleh semua *server* yang ada, namun akan memenuhi *server* sehingga *server* down atau nonaktif untuk sementara, i) *Request* yang gagal dikerjakan *server* diretransmission sehingga *request* dapat dikerjakan *server* kembali hingga *request* tersebut selesai, j) Proses atau *request* yang sudah selesai dikerjakan oleh *server* yakni yang sudah berupa *response* dikembalikan menuju *load balancer* untuk dikirimkan ke *client*.

B. Pembuatan Casing Nanopi-2

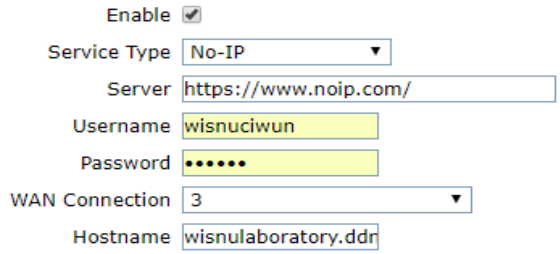
Untuk tujuan perlindungan terhadap Nanopi-2 dan kerapian sistem maka dibuatkan *box* dari bahan akrilik dengan keterangan untuk bagian sisi kanan dan kiri sebesar 12 x 9 cm, untuk alas sebesar 15 x 12 cm, serta untuk dudukan Nanopi-2 11 x 9 cm sebanyak 2 buah dan 1 buah untuk bagian penutup atas. Nanopi-2 yang telah dikemas ditunjukkan dalam Gambar 2.



Gambar 2. Nanopi-2 dalam casing

C. *Konfigurasi Router*

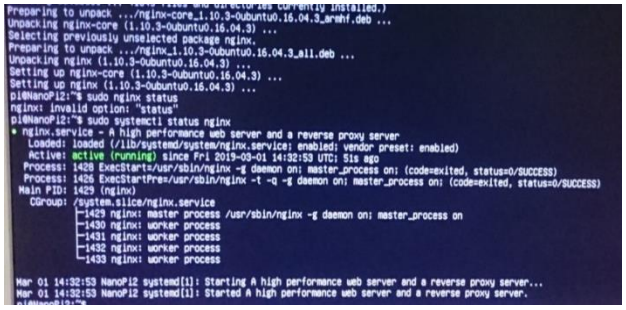
Router yang digunakan adalah ZTE F609. Konfigurasi dilakukan setelah DNS (*Domain Name Server*) dibuat. Caranya adalah dengan memasukan IP 192.168.1.1 pada web *browser*. Kemudian mengisi kolom *user* dan password dengan mengetik "*user*". Setelah laman terbuka, memilih tab "*application*" yang berada di kiri dan memilih layanan "DDNS". Kemudian mengisi data sesuai dengan akun yang telah dibuat sebelumnya dan selanjutnya disubmit. Hasilnya ditunjukkan dalam Gambar 3.



Gambar 3. Konfigurasi DDNS pada *router*

D. *Persiapan Web Server*

Web *server* yang diinstal adalah *nginx*. Setelah statusnya sudah menyatakan bahwa telah ter-*install* dengan baik selanjutnya perlu diketahui hasilnya pada *web browser*. Untuk itu perlu diketahui IP *host* yang di-*broadcast* oleh AP (*Access Point*) pada Nanopi-2. Kondisi web *server* yang aktif ditunjukkan dalam Gambar 4.



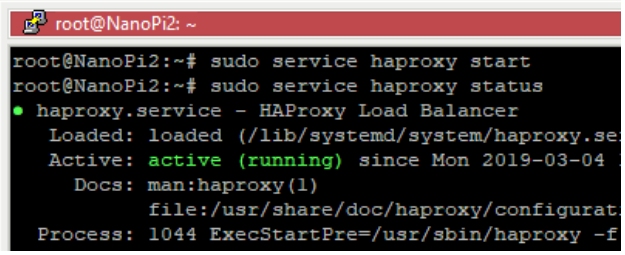
Gambar 4. Status web *server* aktif

E. *Persiapan HAProxy Load balancer*

Load balancer menggunakan *software* HAProxy yang bukan merupakan *software* utama tetapi bersifat tambahan. Hal ini mengakibatkan diperlukan tambahan *repository* atau sumber *package* Linux. Untuk itu dilakukan instalasi *Personal Package Archive* (PPA).

Setelah penambahan *repository* dan *update source list* serta mengisntal HAProxy selanjutnya melakukan konfigurasi HAProxy dengnan menambahkan sintaks untuk *front end* dan *back end*.

Perlu dilakukan juga pengecekan hasil pembagian trafik dan status *server* dengan membuka web *browser* melalui IP host Nanopi-2 yang telah terpasang HAProxy tadi. Pengecekan yang berhasil akan menampilkan tabel dengan status *server* dan *load balancer*. Status *load balancer* yang aktif ditunjukkan dalam Gambar 5.



Gambar 5. Status *load balancer* aktif

F. *Pengujian Beban Trafik*

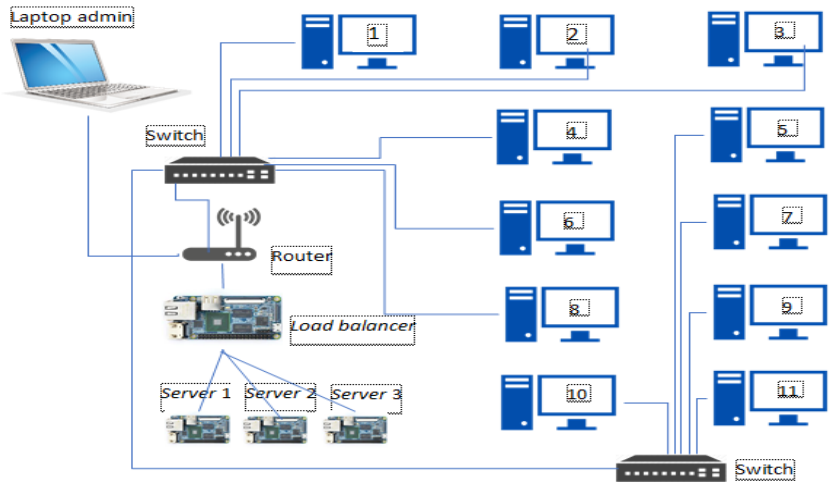
Pengujian ini bertujuan untuk mengetahui pembagian trafik antara 3 *server* yang ada dalam *cluster*. Link domain *server* yang dituju adalah wisnulaboratory.ddns.net. Terdapat 2 metode yaitu real *user* (sejumlah *user* mengakses mengakses *browser*) dan *request flooding* menggunakan *software* paessler *webstress*.

Topologi jaringan yang digunakan pada pengujian dengan real *user* ditunjukkan pada Gambar 6, dengan 11 node *user* yang mengakses *server*. Node *user* dibagi menjadi 3 kelompok tugas, yaitu menjalankan *browser* dan beraktifitas pada *website* yang telah ditentuka, menjalankan program *webstress*, dan menjalankan FTP menggunakan Filezila.

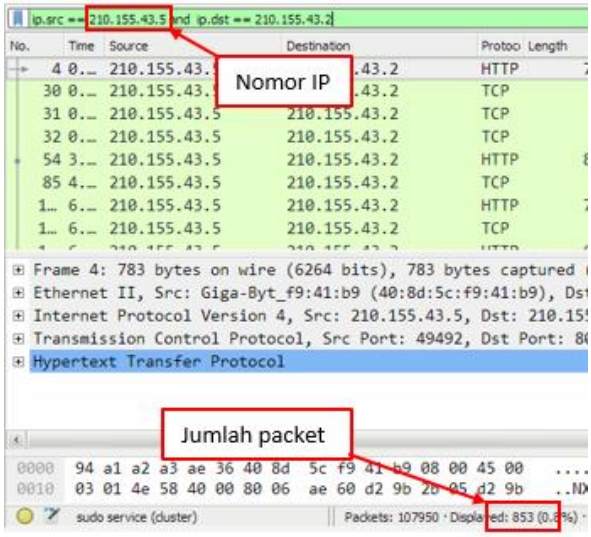
Client yang bertugas menjalankan *browser* adalah 7 komputer dengan IP (210.155.43.5), (210.155.43.9), (210.155.43.10), (210.155.43.12), (210.155.43.16), (210.155.43.17) dan (210.155.43.18). Dua komputer *client* menjalankan program *webstress* yaitu (210.155.43.7.) dan (210.155.43.11), sedangkan *client* yang menjalankan FTP adalah *client* dengan IP (210.155.43.14) dan (210.155.43.15). *Server* dan *load balancer* menggunakan IP sesuai yang telah dijelaskan pada bagian sebelumnya.

Tool yang digunakan untuk mengetahui banyaknya packet yang diproses oleh dan dari *server* adalah Wireshark. Melalui *tool* ini dapat diketahui kondisi trafik pada saat *request* maupun *response*. Tampilan Wireshark ditunjukkan dalam Gambar 7.

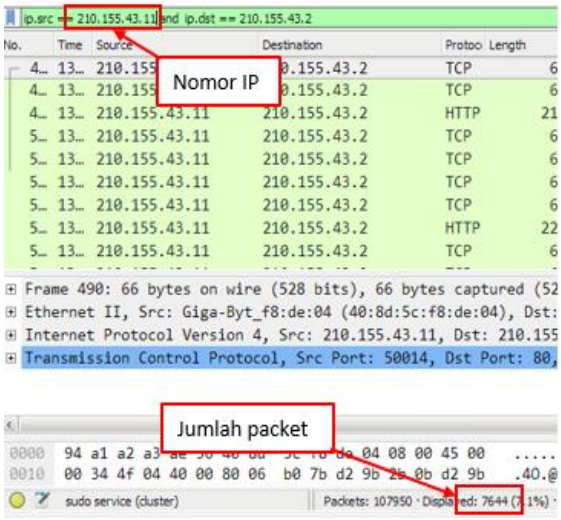
IP *client* ditunjukkan pada bagian atas (header) dari tampilan Wireshark (yaitu 210.155.43.5). Gambar 7 menunjukkan paket trafik yang dikirimkan oleh *client* dan diterima oleh *load balancer* (210.155.43.2). Banyaknya paket yang dikirimkan oleh *client* ditunjukkan pada bagian kanan bawah yaitu sebanyak 853 paket.



Gambar 6. Pengujian Jaringan



Gambar 7. Tampilan Wireshark dari *client* 210.155.43.5 (real *user*)

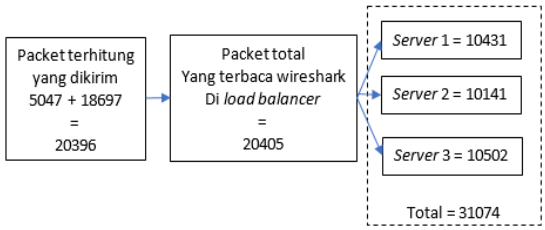


Gambar 8. Tampilan Wireshark dari *client* 210.155.43.11 (*flooding*)

Pengujian yang dilakukan dengan metode berbeda yaitu tidak menggunakan *real user* melainkan menggunakan *request flooding* dengan bantuan *tool* Paessler *webstress*. Gambar 8 menunjukkan tampilan Wireshark untuk *request flooding* dari *client* dengan IP 210.155.43.11. Diketahui bahwa banyaknya paket yang terkirim adalah 7644 paket.

Dari kedua pengujian diketahui bahwa sistem telah dapat menerima trafik baik dari real *user* maupun *request flooding*. Selanjutnya dilakukan pengujian untuk mengetahui pembagian trafik pada 3 *server*.

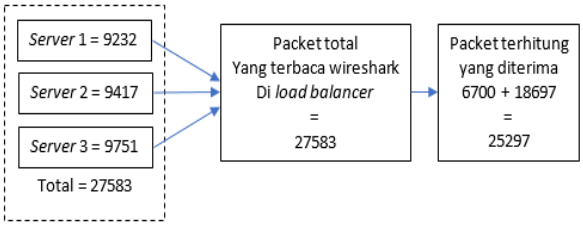
Pengujian yang berkaitan dengan pembagian trafik dilaksanakan untuk 2 arah yaitu *request* (dari *client* menuju *server*) dan *response* (dari *server* menuju *client*). Pada setiap arah, trafik yang dicatat berasal dari real *user* dan *request flooding*. Hasil yang diperoleh untuk *request* ditunjukkan pada Gambar 9, sedangkan hasil pembagian trafik pada *response* ditunjukkan dalam Gambar 10.



Gambar 9. Pembagian trafik *request* yang diterima *server*

Dari Gambar 9 diketahui bahwa real *user* (*client* 1 sampai *client* 11) mengirimkan total 5047 paket sedangkan *request flooding* menghasilkan 18697 paket sehingga total paket yang dikirimkan oleh *client* adalah 20396. Paket ini diterima oleh *load balancer* sebesar 20405, tidak sama dengan yang dikirimkan *client* karena Wireshark pada *load balancer* membaca juga paket yang lain. Jumlah yang ada pada *load balancer* akan berbeda dengan paket yang diterima oleh *server* (sebagai akibat dibacanya paket TCP, retransmisi, dan duplikasi paket). Saat menggunakan algoritma round robin dilakukan pengaturan dan pembagian trafik, sehingga paket yang diterima oleh *server* 1 adalah 10430 paket, *server* 2 sebanyak 10141 paket, dan *server* 3 adalah 10502, atau

rata-rata perbandingan pada ketiga *server* adalah 33.57%, 32.64%, dan 33.8%.



Gambar 10. Pembagian trafik *response* yang dikirim *server*

Hal yang serupa terjadi pada arah sebaliknya (*response*). Ketiga *server* membagi beban sebesar 9232 paket pada *server* 1, 9417 di *server* 2, dan 9751 paket pada *server* 1 untuk selanjutnya diterima oleh *load balancer* dan didistribusikan kembali pada *client*. Perbandingan pada ketiga *server* adalah 33.47%, 34.14%, dan 35.35%.

IV. KESIMPULAN

Penelitian ini menggunakan 4 Nanopi-2 dengan distribusi yaitu 3 difungsikan sebagai *server* dan 1 sebagai *load balancer*. Ketiga *server* berada dalam satu kelompok sehingga membentuk *cluster server*.

Hasil pengujian menunjukkan bahwa *load balancer* telah berhasil melakukan pembagian trafik yang hampir berimbang pada ketiga *server*. Pada saat menerima trafik *request* dari *client*, masing-masing *server* menerima kurang lebih 33.57%, 32.64%, dan 33.8% paket. Ketika mengirimkan trafik *response*, ketiga *server* bekerja dengan perbandingan sekitar 33.47%, 34.14%, dan 35.35%. Pembagian yang berimbang ini memberikan performa yang baik pada ketiga *server*.

Saran yang dapat diberikan adalah perlunya dikembangkan pengujian untuk mengetahui kapasitas maksimal yang dapat dilayani oleh *server* berkaitan dengan kualitas layanan yang diberikan.

REFERENSI

[1] Bourke, Tony, "*Server* Load Balancing", O'Reilly & Associates, Inc, Amerika Serikat, 2001

[2] Putra, RH, "Implementasi *Cluster Server* pada Raspberry Pi dengan Metode Load Balancing", Jurnal Edukasi dan Penelitian Informatika, Vol. 2 No.1, 2016, eISSN :2548-9364.

[3] Mados, Branislav, "Downsizing of Web *Server* Design Using RaspberryPi 3 Single Board Computer Platform", 14th International Scientific Conference on Informatics, 2017, Slovakia. IEEE publisher. **DOI:** 10.1109/INFORMATICS.2017.8327253 .

[4] Maduranga, MWP, "Comparison of Load Balancing Algorithm on Raspberi Pi *Cluster*ed Embedded Web *Server*", 20 th IEEE International Computer Science and Engoneering Conference, 2016.

[5] Rahmatulloh, Alam, "Implementasi Load Balancing *Server* Menggunakan HAProxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Sriwijaya", Jurnal Teknologi dan Sistem Informasi Vol 03 No 02, 2017.