RANCANG BANGUN WEB SERVER BERBASIS LINUX DENGAN METODE LOAD BALANCING (STUDI KASUS : LABORATORIUM TEKNIK INFORMATIKA)

Prayudi Aditya Nugraha[1], M. Azhar Irwansyah[2], Heri Priyanto[3]
Program Studi Teknik Informatika Universitas Tanjungpura[1,2,3]
e-mail: prayudi.aditya.nugraha@gmail.com[1], irwansyah.azhar@untan.ac.id[2], heripriyanto.stmt@untan.ac.id[3]

Saat ini internet sudah menjadi kebutuhan pokok di berbagai bidang kehidupan. Seiring dengan berkembangnya kebutuhan pengguna dan peningkatan permintaan pada suatu situs web maka kinerja dari web server bertambah berat. Web server yang handal tentunya mampu melayani request dari pengguna dalam jumlah yang cukup besar dalam satu satuan waktu. Namun terkadang web server mengalami down atau fail dimana web server tidak mampu lagi menangani jumlah request yang sangat besar dalam satu satuan waktu atau kadang web server mengalami masalah atau kerusakan. Seperti halnya pada Laboratorium Program Studi Teknik Informatika yang memiliki web server yang dapat diakses oleh klien. Web server tersebut sewaktu-waktu dapat mengalami down atau fail. Salah satu solusi yang dianggap tepat untuk mengatasi masalah tersebut adalah dengan menggunakan metode load balancing. Dalam membangun web server dengan load balancing ada beberapa tahap yang harus dikerjakan antara lain mengumpulkan data, lalu masuk ke proses pengembangan arsitektur jaringan dengan menambahkan load balancing pada jaringan yang sudah ada, kemudian web server diuji dan dianalisis. Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan sebagai berikut : (1) Hasil pengujian ketersediaan web server dengan load balancing dapat memberikan ketersediaan layanan lebih baik dibandingkan web server tunggal. (2) Hasil pengujian throughput web server load balancing dengan algoritma round robin memiliki throughput paling baik yaitu 0,19 MB/detik dibandingkan dengan web server tunggal dengan throughput 0,182 MB/detik dan web server load balancing dengan algoritma least connection yang memiliki throughput paling kecil yaitu 0,174 MB/detik. (3) pengujian waktu respon web server load balancing dengan algoritma least connection memiliki waktu respon tercepat yaitu 0,258 detik diikuti oleh web server load balancing dengan algoritma round robin memiliki waktu respon 0,26 detik, sedangkan web server tunggal memiliki waktu respon paling lama yaitu 0,284 detik.

Kata kunci : Ketersediaan, Load Balancing, Web Server.

I. PENDAHULUAN

Saat ini *internet* sudah menjadi kebutuhan pokok di berbagai bidang kehidupan. Bidang ekonomi, sosial, budaya maupun pendidikan memanfaatkan *internet* sebagai media penyebaran informasi dan komunikasi dengan biaya relatif murah, jangkuan yang sangat luas dan efisien. Salah satu jenis layanan dari *internet* yaitu *World Wide Web*. *World Wide Web* atau *web* adalah suatu cara mengakses informasi melalui media *internet*. *Web* menggunakan protokol HTTP untuk mengirimkan data. *Web server* bertanggung jawab melayani permintaan HTTP dari aplikasi klien yang dikenal dengan *web browser*.

Seiring dengan berkembangnya kebutuhan pengguna dan peningkatan permintaan pada suatu situs *web* maka kerja dari *web server* bertambah berat. *Web server* yang handal tentunya mampu melayani *request* dari pengguna dalam jumlah yang cukup besar dalam satu satuan waktu. Namun terkadang *web server* mengalami *down* atau *fail* dimana *web server* tidak mampu lagi menangani jumlah *request* yang sangat besar dalam satu satuan waktu atau kadang *web server* mengalami masalah atau kerusakan.

Seperti halnya pada Laboratorium Program Studi Teknik Informatika yang memiliki *web server* yang dapat diakses oleh klien. *Web server* tersebut sewaktu-waktu dapat mengalami *down* atau *fail*. Hal ini tentu akan mengganggu proses pertukaran data yang terjadi antara *web server* dan klien. Permasalahan tersebut terjadi diakibatkan karena *web server* hanya memiliki mesin tunggal atau single *web server*. Salah satu solusi yang biasa dilakukan adalah dengan cara meminimalisir proses yang berjalan pada *web server* tersebut. Namun hal tersebut tidak memberikan efek yang begitu besar untuk mengatasi *web server* yang *down*. Solusi lain yang dapat dilakukan adalah dengan meng-*upgrade* spesifikasi *web server*. Namun hal itu tentu akan menghabiskan biaya yang cukup besar. Salah satu solusi yang dianggap tepat untuk mengatasi masalah tersebut adalah dengan menggunakan metode *load balancing*.

Load balancing adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari over*load* pada salah satu jalur koneksi. Salah satu jenis *load balancer* adalah HaProxy, yaitu sebuah software *load balancer* untuk protokol TCP dan HTTP, HaProxy dipilih karena memiliki kemampuan mengontrol trafik dari masing-masing *request* data dari klien, bukan hanya berdasarkan jumlah koneksi yang masuk. Untuk mengoptimalkan *load balancing* maka perlu digunakan algoritma untuk mengatur penjadwalan jalannya proses pembagian koneksi. Algortima yang digunakan yaitu *Round Robin* dan *Least Connection*.

Oleh karena itu, untuk mengatasi permasalahan yang ada, perlu dibangun sebuah sistem penyeimbang pembagian koneksi *web server* pada Prodi Teknik Informatika yang dapat membagi beban *request* dari klien.

II. URAIAN PENELITIAN

A. Web Server

Web server adalah sebuah bentuk *server* yang khusus digunakan untuk menyimpan halaman *web*site atau homepage. Dalam melakukan permintaan suatu halaman pada suatu situs *web*, *browser* melakukan koneksi ke suatu *server* dengan protocol HTTP. *Server* akan menanggapi koneksi tersebut dengan mengirimkan isi file yang diminta dan memutuskan koneksi tersebut. *Browser* kemudian mengolah informasi yang didapat dari *server*. Pada bagian *server*, *browser* yang berbeda dapat melakukan koneksi pada *server* yang sama untuk memperoleh informasi yang sama. Data ini mempunyai format yang standar, disebut dengan format SGML (Standart General Markup Language). Data yang berupa format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemapuan *browser* tersebut. *Web server* yang terkenal diantaranya adalah Apache. *Web server* merupakan software yang menjadi tulang punggung dari *World Wide Web* (WWW) [5].

B. Load Balancing

Load balancing adalah proses pendistribusian beban terhadap sebuah servis yang ada pada sekumpulan *server* atau perangkat jaringan ketika ada permintaan dari pemakai [7].

Secara umum, jika ada pengguna mengirimkan permintaan HTTP untuk sebuah alamat *web*, permintaan diarahkan ke *web server* yang ditentukan oleh *Domain Name System* (DNS) [7].

Semua permintaan akan ditangani oleh mesin ini. Dalam *Load balancing*, layanan dapat didistribusikan dengan mengirim permintaan berikutnya ke *server* yang berbeda. Sehingga dengan *Load balancing*, permintaan untuk layanan pada sebuah *server* dapat tersebar di hampir semua jumlah *server* yang disediakan [2].

Load balancing pada *web server* berada di layer 4 yaitu untuk mendistribusikan permintaan ke *server* pada lapisan *transport*, seperti TCP dan UDP [2].

Dalam membuat sebuah *Load balancing* dapat menggunakan beberapa metode yaitu :

1. *Balance*, solusi yang simpel dalam membuat sebuah *Load balancing* namun tidak dapat menjaga *session* pengguna [2].
2. Eddie Mission, merupakan penyempurnaan dari *balance* dengan dapat menjaga *session* dari pengguna [2].
3. Enhanced DNS *Server*, pada metode ini harus memasang eddie mission terlebih dahulu dalam membuat sebuah *Load balancing* [2].
4. LVS (Linux *Virtual Server*), perangkat lunak yang dapat menangani *Load balancing* dengan beberapa metode [2].

C. Algoritma Penjadwalan pada Load Balancing

Linux *director* perlu menerapkan algoritma jadwal tertentu pada *Load balancing* [3]. Algoritma penjadwalan (*sceduling algorithm*) *Load balancing* yang digunakan antara lain sebagai berikut :

1. *Round Robin* adalah algoritma yang berfungsi untuk mengarahkan koneksi jaringan pada sebuah model *Round Robin*. Penjadwalan ini memberlakukan semua *real server* sama menurut jumlah koneksi dan waktu respon. Kelemahannya adalah membutuhkan peramalan beban (*load*) yang akurat untuk setiap request agar algoritma ini bisa berjalan dengan efektif [3].
2. *Least Connection* adalah algoritma untuk mengarahkan koneksi baru ke *real server* dengan koneksi aktif yang paling sedikit. Penjadwalan ini merupakan penjadwalan yang dinamik, karena memerlukan perhitungan koneksi aktif untuk masing-masing *real server* secara dinamik [3].

D. HAProxy

HAProxy adalah sebuah software gratis yang sangat cepat dan handal dalam memberikan ketersediaan yang tinggi terhadap *server*, *load balancing*, dan proxy untuk TCP dan aplikasi berbasis HTTP. HAProxy sangat cocok diterapkan pada *server* yang memiliki trafik yang sangat tinggi. Selama bertahun-tahun telah menjadi standar de-facto untuk aplikasi opensource *load balancer* dan telah tersebar ke berbagai macam distro linux dan digunakan secara default pada platform cloud [1].

HAProxy dikenal sangat handal dan dapat berjalan pada beberapa sistem operasi antara lain Linux 2.4, Linux 2.6/3.x, Solaris 8/9, Solaris 10, FreeBSD 4.10-10, OpenBSD 3.1 dan AIX 5.1-5.3. HAProxy dapat berjalan dengan baik pada sistem operasi tersebut dan memberikan kinerja yang tinggi seperti transfer data yang cepat, trafik forwarding hingga 40 Gbps.

HAProxy dipilih karena memiliki kemampuan mengontrol trafik dari masing-masing *request* data dari *client*, bukan hanya berdasarkan jumlah koneksi yang masuk, selain itu HaProxy dipilih karena terdapat fitur untuk menampilkan statistik dari penggunaan masing-masing *web server* yang dikontrolnya [1].

E. Network Address Translation (NAT)

Network Address Translation (NAT) yaitu sebuah metode yang memanipulasi alamat ip dan nomor *port* baik berasal sumber maupun tujuannya. Alamat ip publik disamarkan untuk digunakan oleh alamat ip privat agar bisa berhubungan dengan *internet*. Linux *director* dan *real server* dihubungkan oleh switch. *Real server* biasanya menjalankan layanan yang sama dan mempunyai layanan yang sama [1].

Cara kerja *Network Address Translation* NAT adalah sebagai berikut :

1. Ketika *user* mengakses layanan yang disediakan oleh *server*. Maka paket yang dikirimkan akan sampai ke *director* [1].
2. *Director* lalu akan memilih *server* berdasarkan penjadwalan dan menulis ulang alamat tujuan paket ke alamat IP *real server* lalu meneruskannya ke *real server* [1].
3. Ketika korespondensi *server* terhadap proses permintaan dan balasan sesuai dengan permintaan, maka dikirimkan ke *director* [1].
4. *Director* menulis ulang alamat dari paket balasan ke alamat IP dari *real server* dan kemudian mengembalikan paket ke *user* [1].

F. Secure Shell (SSH)

Secure Shell (SSH) merupakan protokol jaringan yang dapat mengakomodasi transfer data antara dua buah komputer melalui jalur komunikasi yang aman. Keamanan yang disediakan oleh protocol SSH adalah melalui fasilitas enkripsi yang mampu membuat integritas dan kerahasiaan data tetap

terjaga. SSH menggunakn kriptograpi public key untuk melakukan autentifikasi komputer remote dan untuk menginginkan komputer remote untuk mengautentikasi *user* jika diperlukan. *Server* SSH yang biasa digunakan adalah OpenSSH yang secara default akan melakukan listening di *port* 22 [8].

G. *Protokol HTTP*

HTTP adalah tipe protokol meminta/menjawab, dimana *client* akan membuka koneksi ke suatu *server* dan kemudian mengirimkan permintaan menggunakan format spesifik. *Server* akan memberi tanggapan kemudian akan menutup koneksi. HTTP memiliki kemampuan untuk mentransfer halaman web, gambar dan media lainnya yang dipergunakan oleh aplikasi Web [4].

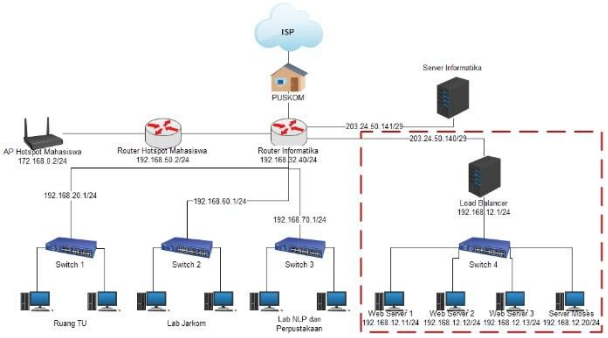
H. *Web Browser*

Browser adalah sebuah program yang dirancang untuk mengambil informasi-informasi dari *server* komputer pada suatu jaringan *internet* maupun intranet. *Web Browser* adalah suatu program yang digunakan untuk menjelajahi dunia *internet* atau untuk mencari informasi tentang suatu halaman web yang tersimpan di komputer. Cara kerja *web browser* adalah pada saat kita mengetikkan suatu alamat pada browser maka data akan dilewatkan oleh suatu protocol HTTP melewati *port* 80 pada *server* [8].

III. PERANCANGAN SISTEM

A. *Perancangan Pengembangan Arsitektur Jaringan*

Pada tahap ini dilakukan pengembangan arsitektur jaringan yang telah ada pada jaringan Prodi Teknik Informatika untuk menambahkan *load balancer* dan beberapa *backend web server*. Pengembangan arsitektur jaringan yang akan diterapkan pada jaringan Prodi Teknik Informatika dijelaskan pada Gambar 1 berikut.



Gambar 1 Pengembangan Arsitektur Jaringan

B. *Pembuatan Sistem*

Pembuatan sistem yang dilakukan antara lain konfigurasi IP Address, update dan *upgrade* Kernel Linux, konfigurasi *Network Address Translation* (NAT), konfigurasi SSH *server*, konfigurasi FTP *server*, instalasi paket pendukung *load balancer*, konfigurasi HAProxy, konfigurasi *web server*, instalasi PHP, serta konfigurasi moses *server* dan integrasi moses *server* dan *web server*.

C. *Implementasi Pengembangan Arsitektur Jaringan*

Pengembangan arsitektur jaringan yang dilakukan yaitu dengan meletakkan posisi *load balancer* dan beberapa *backend web server* yang terhubung ke routerboard utama melalui *port ethernet* 9 yang memiliki IP Public 203.24.50.140 dan netmask 255.255.255.248. Untuk lebih jelasnya pengembangan arsitektur jaringan mengacu pada Gambar 1.

D. *Pengujian Sistem*

1. *Pengujian Ketersediaan*

Pengujian ketersediaan adalah pengujian terhadap ketersediaan layanan dari *web server*. Hal ini dilakukan dengan cara mengakses *web server* dalam keadaan *down* baik itu *web server* tunggal tanpa *load balancing* dan *web server* dengan *load balancing*. Lalu akan diuji dengan cara mengakses *web server* melalui *web browser*. Pengujian ini dilakukan untuk membuktikan apakah *web server* masih dapat memberikan layanan atau tidak walaupun dalam keadaan *down* [1].

Tabel 1
Pengujian Ketersediaan *Web Server*

Jenis *Web Server*	Keterangan	Hasil Pengujian
Web Server Tunggal	*Web server* dalam keadaan *down*	Tidak dapat diakses
Web Server dengan *Load Balancing*	1 dari 3 *web server* dalam keadaan *down*	Dapat diakses
Web Server dengan *Load Balancing*	2 dari 3 *web server* dalam keadaan *down*	Dapat diakses
Web Server dengan *Load Balancing*	3 dari 3 *web server* dalam keadaan *down*	Tidak dapat diakses

2. *Pengujian Throughput*

Dalam pengujian ini dilakukan pengukuran *throughput* pada *web server* tunggal tanpa *load balancing* dan *web server* dengan *load balancing*. *Throughput* digunakan untuk mengetahui kemampuan *web server* dalam memberikan layanan secara benar terhadap *request* yang datang secara bersamaan [1]. Pengujian ini menggunakan simulasi mengirimkan koneksi secara bersamaan sebanyak 500 hingga 2500 *request*, dengan hasil pengamatan sebagai berikut.

Tabel 2
Pengujian *Throughput Web Server*

Jumlah *Request*	Lama Waktu Tes *Server* Tunggal (detik)	Lama Waktu Tes *Round Robin* (detik)	Lama Waktu Tes *Least Connection* (detik)	*Throughput Server* Tunggal (MB/detik)	*Throughput Round Robin* (MB/detik)	*Throughput Least Connection* (MB/detik)
500	3,22	2,79	3,3	0,18	0,21	0,17
1000	6,9	8,66	9,03	0,17	0,13	0,13
1500	9,32	8,3	9,3	0,18	0,21	0,18
2000	12,11	12,04	12,57	0,19	0,19	0,18
2500	15,05	13,73	13,72	0,19	0,21	0,21
Rata-rata	**9,32**	**9,104**	**9,584**	**0,182**	**0,19**	**0,174**

3. *Pengujian Waktu Respon*

Dalam pengujian ini dilakukan pengukuran waktu respon pada *web server* tunggal tanpa *load balancing* dan *web server*

dengan *load balancing*. Pengujian ini dilakukan untuk mengetahui waktu respon terhadap *request* yang datang [1]. Pengujian ini menggunakan simulasi mengirimkan koneksi secara bersamaan sebanyak 500 hingga 2500 *request*, dengan hasil pengamatan sebagai berikut.

Tabel 3
Pengujian Waktu Respon *Web Server*

Jumlah *Request*	**Lama Waktu Tes *Server* Tunggal (detik)**	**Lama Waktu Tes *Round Robin* (detik)**	**Lama Waktu Tes *Least Connection* (detik)**	**Waktu Respon *Server* Tunggal (detik)**	**Waktu Respon *Round Robin* (detik)**	**Waktu Respon *Least Connection* (detik)**
500	3,22	2,79	3,3	0,29	0,26	0,25
1000	6,9	8,66	9,03	0,28	0,26	0,25
1500	9,32	8,3	9,3	0,28	0,26	0,27
2000	12,11	12,04	12,57	0,28	0,26	0,26
2500	15,05	13,73	13,72	0,29	0,26	0,26
Rata-rata	**9,32**	**9,104**	**9,584**	**0,284**	**0,26**	**0,258**

E. Analisis Hasil Pengujian

Rincian hasil analisis pengujian *web server* yang telah dilakukan adalah sebagai berikut:

1. Berdasarkan hasil pengujian ketersediaan pada *web server* yang *down* didapat hasil bahwa pada *web server* tunggal tidak dapat memberikan ketersediaan layanan kepada *client* yang melakukan *request*, sedangkan *web server* dengan *load balancing* masih dapat memberikan ketersediaan layanan kepada *client* walaupun terdapat beberapa *web server* yang *down*. Namun *web server* dengan *load balancing* tidak dapat memberikan ketersedian layanan apabila seluruh *backend web server* mengalami *down*.
2. Berdasarkan hasil pengujian *throughput* didapat hasil untuk *web server* tunggal memiliki rata-rata *throughput* 0,182 MB/detik, sedangkan untuk *web server load balancing* dengan algoritma *round robin* memiliki *throughput* paling besar yaitu 0,19 MB/detik dan *web server load balancing* dengan algoritma *least connection* memiliki *throughput* 0,174 MB/detik. Hal ini membuktikan bahwa *web server* dengan *load balancing* dapat meningkatkan *throughput* dibandingkan *web server* tunggal.
3. Berdasarkan hasil pengujian waktu respon didapat hasil untuk *web server* tunggal memiliki rata-rata waktu respon terlama dibandingkan dengan *web server* dengan *load balancing* yaitu dengan waktu respon 0,284 detik, sedangkan untuk *web server load balancing* dengan algoritma *round robin* memiliki waktu respon 0,26 detik dan *web server load balancing* dengan algoritma *least connection* memiliki waktu respon tercepat yaitu 0,258 detik. Hal ini membuktikan bahwa *web server* dengan *load balancing* dapat meningkatkan waktu dibandingkan *web server* tunggal.
4. Berdasarkan hasil pengujian *throughput* dan waktu respon yang telah dilakukan pada *web server* dengan metode *load balancing* maka didapat hasil sebagai berikut:
   a. Pada pengujian *throughput*, *web server* dengan algoritma *round robin* memiliki *throughput* paling besar dibandingkan *web server* dengan algortima *least connection* yaitu 0,19 MB/detik. Sedangkan *web server* dengan algoritma *least connection* hanya memilki *throughput* sebesar 0,174 MB/detik.
   b. Pada pengujian waktu respon, *web server* dengan algoritma *round robin* memiliki hasil yang lebih stabil dan konstan pada setiap jumlah HTTP *request* yang diberikan yaitu 0,26 detik. Sedangkan *web server* dengan algoritma *least connection* memiliki hasil yang berbeda-beda pada setiap jumlah HTTP *request* yang diberikan, namun *web server* dengan algoritma *least connection* memiliki rata-rata waktu respon lebih baik dibandingkan *web server* dengan algoritma *round robin* yaitu 0,258 detik.

   Dengan mengamati dua faktor diatas maka didapat rekomendasi *web server* yang cocok untuk diterapkan pada Laboratorium Teknik Informatika yaitu *web server* dengan algortima *round robin* karena memiliki *throughput* paling besar dan memiliki waktu respon yang stabil pada setiap jumlah HTTP *request* yang diberikan.

IV. KESIMPULAN/RINGKASAN

Berdasarkan hasil implementasi dan hasil analisis pengujian terhadap *web server* tunggal dan *web server* dengan metode *load balancing* dapat disimpulkan bahwa:

1. Berdasarkan pengujian ketersediaan *web server* dengan *load balancing* dapat memberikan ketersediaan layanan lebih baik dibandingkan *web server* tunggal.
2. Berdasarkan pengujian *throughput web server load balancing* dengan algoritma *round robin* memiliki *throughput* paling baik yaitu 0,19 MB/detik dibandingkan dengan *web server* tunggal dengan *throughput* 0,182 MB/detik dan *web server load balancing* dengan algoritma *least connection* yang memiliki *throughput* paling kecil yaitu 0,174 MB/detik.
3. Berdasarkan pengujian waktu respon *web server load balancing* dengan algoritma *least connection* memiliki waktu respon tercepat yaitu 0,258 detik diikuti oleh *web server load balancing* dengan algoritma *round robin* memiliki waktu respon 0,26 detik, sedangkan *web server* tunggal memiliki waktu respon paling lama yaitu 0,284 detik.
4. Setelah dilakukan pengujian serta analisis hasil pengujian pada *web server* yang telah diterapkan pada Laboratorium Teknik Informatika, maka dapat diberikan rekomendasi *web server* yang cocok untuk diterapkan yaitu *web server load balancing* dengan algortima *round robin* karena memiliki *throughput* paling besar dan memiliki waktu respon yang stabil pada setiap jumlah HTTP *request* yang diberikan.

DAFTAR PUSTAKA

[1] Asyanto, Budi. 2011. Perancangan dan Pembuatan *Load Balancing* Pada Clustering *Web Server* Menggunakan LVS (Studi Kasus *Web Server* LEMIGAS). Program Studi Teknik Informatika, Universitas Islam Negeri Syarif Hidayatullah. Jakarta

[2] Bookman, Charles. 2003. Linux Clustering: Building and Maintaining Linux Clusters. New Riders Publishing. USA
[3] Jefry Alvonsius Rabu, Joko Puttrtadi, dan Willy S. Raharjo. Implementasi *Load Balancing Web Server* Menggunakan Metode LVS-NAT. Universitas Kristen Duta Wacana. Yogyakarta
[4] Kusumo, Ario Suryo. 2004. Buku Latihan Visual Basic .Net. Elex Media Komputindo. Jakarta
[5] Margono, Eko Adriansyah, dan Harry Asibrah. Analisis dan Perancangan *Load Balancing* Pada *Web Server* Berbasis Cloud Pada Kantor DPRD Kota Palembang. Jurusan Teknik Informatika, STMIK PalComTech
[6] Mumpuni, Joko I. Dan Wardono, Adisuryo. 2006. Meningkatkan Kemampuan Jaringan Komputer dengan PC Cloning System Menjadikan PC x86 berkemampuan PC iP4. Yogyakarta : CV. Andi Offset.
[7] Rijayana, Iwan. 2005. Teknologi *Load Balancing* Untuk Mengatasi Beban *Server*. Yogyakarta : Jurnal SNATI
[8] Sahala, Aldo. 2014. Konsep dan Implementasi Jaringan dengan Linux Ubuntu. Andi Yogyakarta. Yogyakarta.
[9] Strauss, Judy, Adel El-Ansary, Raymond Frost. 2003. E-Marketing third edition. New Jersey : Prentice Hall.