# APLIKASI OTOMATISASI PEMBAGIAN BEBAN PADA SISTEM OPERASI LINUX

**LAPORAN TUGAS AKHIR**

**Diajukan Sebagai Salah Satu Syarat**

**Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika**


**Oleh :**

Nama : Fietyata Yudha

NIM : 07 523 027

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**YOGYAKARTA**

**2011**

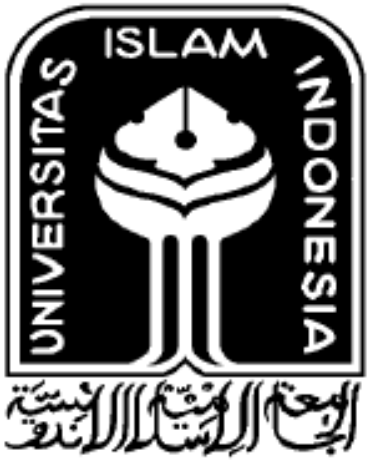
**HALAMAN JUDUL**

# APLIKASI OTOMATISASI PEMBAGIAN BEBAN PADA SISTEM OPERASI LINUX

**LAPORAN TUGAS AKHIR**

**Diajukan Sebagai Salah Satu Syarat**

**Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika**




**Oleh :**

Nama : Fietyata Yudha

NIM : 07 523 027

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**YOGYAKARTA**

**2011**

**LEMBAR PENGESAHAN PEMBIMBING**

**APLIKASI OTOMATISASI PEMBAGIAN BEBAN PADA SISTEM OPERASI LINUX**

**Oleh :**

Nama : Fietyata Yudha

NIM : 07 523 027

Yogyakarta, 20 Agustus 2011

Pembimbing

(R. Teduh Dirgahayu, ST, M. Sc, Ph. D)



ii

**LEMBAR PENGESAHAN PENGUJI**

**APLIKASI OTOMATISASI PEMBAGIAN BEBAN PADA SISTEM OPERASI LINUX**

Oleh :

Nama : Fietyata Yudha

NIM : 07 523 027

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat

untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika

Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta 22 Agustus 2011

Tim Penguji :

R. Teduh Dirgahayu, ST, M. Sc, Ph. D
Ketua

Syarif Hidayat, S. Kom, M. IT
Anggota 1

Hendrik, S. T, M. Eng
Anggota 2

Mengetahui,

Ketua Jurusan Teknik Informatika

Universitas Islam Indonesia

(Yudi Prayudi, S.Si, M.Kom)



iii

**LEMBAR PERNYATAAN KEASLIAN**
**TUGAS AKHIR**

Saya yang bertanda tangan di bawah ini :

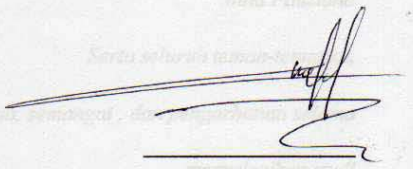
Nama : Fietyata Yudha

NIM : 07 523 027

Menyatakan bahwa seluruh komponen dan isi dalam laporan Tugas Akhir ini adalah hasil karya sendiri. Apabila kemudian hari terbukti bahwa ada beberapa bagian dari karya ini yang bukan hasil karya sendiri, maka saya siap menanggung resiko dan konsekuensinya.

Demikian pernyataan ini dibuat, supaya dapat dipergunakan dengan sebagaimana mestinya.

Yogyakarta, 2 Agustus 2011



Fietyata Yudha



iv

## HALAMAN PERSEMBAHAN

***Kupersembahkan Tugas Akhir***

***Dengan setulusnya untuk***

*Ibu dan bapakku tercinta,*

*Yayuk Marliana dan Lalu Makruf*

*Adikku,*

*Hasinadara Pramadanti*

*Dan seseorang yang selalu mendukung ku saat susah dan senang,*

*Inna Fauziana*

*Serta seluruh teman-temanku,*

*Yang selalu memberikan Doa, semangat , dan pengorbanan selama*

*menyelsaikan studi*

## HALAMAN MOTTO

*“Sesungguhnya sesudah kesulitan itu ada kemudahan”*

*( Q.S Al Insyirah : 6 )*

*"Hanya mereka yang berani gagal dapat meraih keberhasilan."*
*-- John F. Kennedy. --*

*"Banyak kegagalan dalam hidup ini dikarenakan orang-orang tidak menyadari betapa dekatnya mereka dengan keberhasilan saat mereka menyerah."*
*-- Thomas Alva Edison –*

*"Keberhasilan tidak diukur dengan apa yang anda raih, namun kegagalan yang telah anda hadapi, dan keberanian yang membuat anda tetap berjuang melawan rintangan yang datang bertubi-tubi."*
*-- Orison Swett Marden --*

## KATA PENGANTAR

بسم الله الرحمن الرحيم

*Assalamualaikum, Wr. Wb.*

Alhamdulillah, puji syukur bagi Allah SWT atas segala n**ikmat dan karunianya, sehingga laporan Tugas Akhir yang berjudul "Aplikasi otomatisasi pembagian beban Pada Sisem Operasi Linux"** dapat terselesaikan. Shalawat serta salam semoga selalu tercurah kepada junjungan kita Nabi Muhammad SAW, serta kerabat dan pengikutnya.

Laporan tugas akhir ini disusun sbagai salah saatu sarat untuk menempuh gelar sarjana jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Laporan ini dibuat sebagai sarana untuk mempraktekkan ilmu yang telah didapat selama menuntut ilmu di jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Dalam penyusunan laporan ini tidak lepas dari dukungan berbagai pihak, oleh karena itu pada kesempatan ini penulis dengan segala kerendahan hati ingin mengucapkan termima kasih kepada :

1. Allah SWT Tuhan bagi seluruh alam atas karunia-Nya dan nikmat-Nya yang sehingga penulis diberikan kesehatan, ketabahan, kemudahan selama mengerjakan tugas akhir ini.
2. Bapak DR. Ir. Lalu Makrup, M.T dan ibu Yayuk Marliana yang selalu memberikan semangat dan dukungan atas terselesaikannya tugas akhir ini.
3. Bapak Gumbolo Hadi Susanto, M. Sc selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia serta seluruh jajarannya.
4. Bapak Yudhi Prayudi, S. Si, M. Kom selaku ketua Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia yang telah memberikan keudahan sehingga dapat terselesaikannya tugas akhir ini.

5. Bapak Teduh Dirgahayu, S. T, M. Sc, Ph. D yang telah membimbing penulis sehingga dapat terselesaikannya tugas akhir yang dibuat.
6. Ibu Izzati Muhimmah, S. T, M. Sc, Ph. D selaku kepala Laboratorium Komputasi dan Sistem Cerdas atas dukungannya selama ini.
7. Bapak Hamid, S. T, M. Eng
8. Dosen-Dosen Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
9. Bapak Ir. Lalu Gafar Ismail, terima kasih atas dukungannya.
10. Bapak Moch. Fajri Salim atas motivasinya.
11. Seseorang yang selalu mendukungku dalam susah dan senang, Inna Fauziana, yang memerikan semangat dan motivasi.
12. Bapak Azmiansyah, S. T serta seluruh crew PT. Rabiha Pilar Informatika Mas Kope, Ancha, Karjok, Novi Miechan, Rizal, Mas Aziz.
13. Seluruh manajemen CV. Imedia Informatika.
14. Teman – teman di Laboratorium Komputasi dan Sistem Cerdas Arief, Mbak Zha, Kisti, Tukul, Robby, Ardhy, Ifa, Yogie, Andhika, Indra, Ari, Tia, Rifqi, Haqi, termia kasih atas dukunganya.
15. Teman-teman di Laboratorium Informatika Terpadu, Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.
16. Teman – teman di “Bajz Company” Dion, Doni_apple, Lupex, Cepe, Apeng, Surya, Susi, Bagus atas canda dan tawanya.
17. Teman-teman “include”
18. Teman-teman KKN Unit 96 Mami Tia yang sudah menjadi ibu, Oqi, Astiti, Wulan, Mas Topek, Lui, Widia dan teman dari unit lain Satyo.
19. Teman-teman kontrakan Candi Mamed, Suhu Adit, Aan terimakasih atas bimbingannya terutama buat Suhu Adit.
20. Teman – teman dikost “Red Top” yang selalu mendukung atas terselaesaikannya lapora ini. Devi, Oliph, Mbak Nel, dkk saya ucapkan terimakasih.
21. Teman-teman angkatan 2005-2010 Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

22. Bapak Onno W. Purbo atas tutorial-tutorialnya.
23. Dan seluruh pihak yang telah mendukung yang tidak dapat kami ucapkan satu per satu kami ucapkan terima kasih.

Kami mengetahui bahwa dalam pembuatan laporan ini banyak sekali kekurangan dan kesalahan, untuk itu kami mengucapkan permohonan maaf yang sebesar-besarya. Diharapkan kritik dan saran yang membangun untuk penyempurnaan di masa mendatang.

Akhir kata semoga laporan ini dapat berguna bagi kita semua, Amin.

*Wassalamualaikum Wr. Wb*

Yogyakarta, Juni 2011

Penulis

# SARI

Server gateway adalah suatu perangkat penting yang digunakan untuk menghubungkan satu atau lebih jaringan komputer. Dalam dunia jaringan komputer gateway biasanya digunakan untuk menghubungkan jaringan lokal ke jaringan Internet. Koneksi Internet yang dimiliki belum tentu hanya menggunakan satu jalur koneksi ke Internet namun bisa menggunakan dua atau lebih koneksi. Pada situasi ini dibutuhkan sebuah perangkat lunak yang dapat melakukan penggabungan koneksi tersebut secara otomatis saat dibutuhkan. Perangkat lunak yang dibuat menggunakan bahasa pemrograman shell dan python yang didukung oleh semua sistem operasi berbasis Linux yang merupakan sistem operasi freeware. Perangkat lunak yang dibuat sudah dapat menjalankan fungsi pembagian beban. Setelah penggunaan aplikasi pembagian beban di gateway, komputer gateway sudah dapat mebagi paket data yang lewat melalui kedua interface outbond.

Kata Kunci :

Koneksi, Server, Gateway, Linux, Python, Shell

## TAKARIR

| | |
|---|---|
| *DNS* | Domain Name Sistem, Nama D omain sebagai pengganti IP address |
| *Fail-over* | Kemampuan untuk mengubah sistem ke sistem lain secara otomatis, jika salah satu sistem mengalami kegagalan |
| *Gateway* | Komputer yang berfungsi sebagai pintu jaringan untuk keluar ke Internet |
| *IP* | Internet Protocol, aturan yang digunakan untuk komunikasi jaringan Internet |
| *Network Manager* | Aplikasi untuk melakukan pengaturan ip address, subnetmask, Gateway, dan DNS pada sistem operasi |
| *NIC* | Network Interface Card, kartu jaringan untuk menghubungkan kompter ke jaringan |
| *Redundancy* | Duplikasi |
| *Router* | Perangkat yang berfungsi untuk melakukan penjaluran terhadap paket data yang akan lewat |
| *Server* | Komputer yang bertugas memberikan layanan ke komputer klien |

## DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Peningkatan jumlah pengguna Internet saat ini harus diimbangi dengan kecepatan koneksi yang memadai. Sebuah komputer *gateway* belum tentu hanya menggunakan satu koneksi saja untuk memenuhi tuntutan kebutuhan yanga ada. Makin banyak koneksi membutuhkannya manajemen koneksi yang handal.

Saat ini aplikasi manajemen koneksi yang ada sudah dapat melayani koneksi lebih dari satu koneksi, pembagian beban kerja dapat dilakukan terhadap beberapa piranti, baik itu pembagian kerja prosesor, *harddisk*, memori, dan lain-lain. Sedangkan pembagian beban dalam manajemen koneksi adalah pembagian beban kerja dilakukan terhadap beberapa koneksi yang sudah dibuat, sehingga layanan yang diberikan lebih handal dan lebih besar kekuatan *bandwidth* dalam melayani client.

Sistem Operasi Linux sudah dapat melakukan pembagian beban terhadap koneksi yang dimiliki sebuah server, tetapi pembagian beban baru bisa dilakukan secara manual dengan menggunakan aplikasi *ip route* dan waktu untuk melakukan pembagian beban masih diatur oleh Administrator. Aplikasi tersebut belum bisa melakukan pembagian beban secara otomatis. Linux dipilih karena sistem operasi ini bersifat bebas dan gratis, dan sudah mendukung load balancing tanpa memasang aplikasi tambahan.

Hingga saat ini, belum ada aplikasi untuk melakukan pembagian beban secara otomatis.

## 1.2 Rumusan Masalah

Bagaimana membuat aplikasi pada sistem operasi Linux sehingga suatu *router gateway* mampu melakukan pembagian beban koneksi secara otomatis.

## 1.3 Tujuan Penelitian

Tujuan penelitian ini adalah membuat sebuah aplikasi pada sistem operasi Linux untuk melakukan pembagian beban secara otomatis.

## 1.4 Batasan Masalah

Batasan masalah tugas akhir adalah :

1. *Aplikasi* ini hanya menangani pembagian beban koneksi, tidak melayani pembagian beban perangkat lain.
2. Batasan lain :
   a. Untuk pengecekan *latency* koneksi menggunakan aplikasi *ping.*
   b. perangkat *Network Interface Card (NIC)* yang berbasikan koneksi kabel.
   c. Bahasa pemrograman yang digunakan adalah bahasa *shell* dan *python.*
   d. Implementasi dilakukan pada sistem operasi Ubuntu Server.
   e. Pengujian dilakukan dengan metode virtualisasi.
   f. *Bandwidth* dari kedua koneksi sama besar, karena Destinasi ping ada di Internet.

## 1.5 Manfaat Penelitian

Hasil penelitian tugas akhir ini diharapkan dapat memudahkan bagi administrator jaringan untuk melakukan pembagian beban kerja dari koneksi Internet pada jaringan yang memiliki lebih dari satu koneksi ke Internet.

## 1.6 Metodologi Penelitian

Penulis memperoleh data melalui studi tentang pembuatan program dengan bahasa python, bahasa shell, dan tentang pembagian beban pada

jaringan baik dari halaman web dan dari berbagai literatur untuk menggali informasi yang berhubungan dengan penelitian yang dibuat.

Perancangan dilakukan dengan membuat diagram alur (*Flow Chart*) untuk mengetahui alur program yang akan dibuat. Dari diagram alur tersebut dapat terlihat proses-proses yang harus dibuat di dalam program.

Implementasi dilakukan dengan mengubah flowchart yang sudah dibuat kedalam bentuk bahasa pemrograman. Bahasa pemrograman yang dipilih adalah bahasa pemrograman yang mendukung untuk eksekusi perintah-perintah shell Linux, karena aplikasi yang yang dibuat berbasiskan shell Linux.

Aplikasi yang sudah dibuat akan di implementasikan pada suatu komputer *router gateway* berbasis sistem operasi Linux dengan sudah terpasang lebih dari dua *Network Interface Card*. Aplikasi dijalankan dengan melakukan instalasi terlebih dahulu dan aplikasi dijalankan. Pegujian dilakukan dengan membebani koneksi sehinggan latency dari koneksi tersebut menjadi besar dan membutuhkan koneksi tambahan.

## 1.7 Sistematika

Sistematika penulisan dan garis besar isi laporan ini adalah sebagai berikut :

Bab I membahas tentang latar belakang pembuatan aplikasi ini, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi, dan sistematika penelitian.

Bab II membahas tentang teori-teori yang berhubungan dengan penelitian ini meliputi : Bahasa Python, pemrograman shell, teori tentang pembagian beban serta algoritmanya juga bagaimana mengkonfigurasikan pembagian beban koneksi di sistem operasi Linux.

Bab III membahas tentang analisis kebutuhan aplikasi tersebut, yang meliputi: analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, kemudian dilanjutkan dengan melakukan desain aplikasi.

Bab IV membahas tentang implementasi perancangan yang sudah dibuat dan dilanjutkan dengan pengujian aplikasi yang sudah dibuat. Pengujian dilakukan dengan melakukan pembebanan terhadap koneksi sehingga latency ke Internet menjadi berat dan membutuhkan pembagian beban koneksi.

Bab V menyajikan kesimpulan dari aplikasi yang sudah dibuat dan saran untuk perbaikan dan pengembangan aplikasi tersebut selanjutnya

# BAB II
# LANDASAN TEORI

## 2.1 Pembagian beban

### 2.1.1 Pengertian

Pembagian beban (*load balancing)* adalah teknik untuk mendistribusikan beban kerja dari secara merata kepada perangkat-perangkat komputer, untuk pemanfaatan sumber daya secara optimal, mendapatkan hasil keluaran yang maksimal, meminimalkan waktu tanggap, dan menhindari *overload*. [ANO11]

### 2.1.2 Fungsi dari Server Pembagian beban

Server pembagian beban ini memiliki banyak fungsi, antara lain [DEN08] :

a. Menerima *traffic* dari sebuah jaringan untuk diteruskan ke jaringan Internet
b. Melakukan pembagian lalu lintas paket menjadi permintaan individual.
c. Memantau server dengan meyakinkan server bahwa server tersebut dapat menangani *traffic.*

### 2.1.3 Algoritma Pembagian beban

Algoritma adalah urutan logis pengambilan keputusan untuk memecahkan masalah. Algoritma dibutuhkan oleh komputer untuk menyelesaikan perintah-perintah yang diberikan oleh pengguna komputer tersebut.

Dalam pembagian beban terdapat beberapa algoritma untuk melakukan pembagian beban, antara lain :

a. DNS Load Balancing

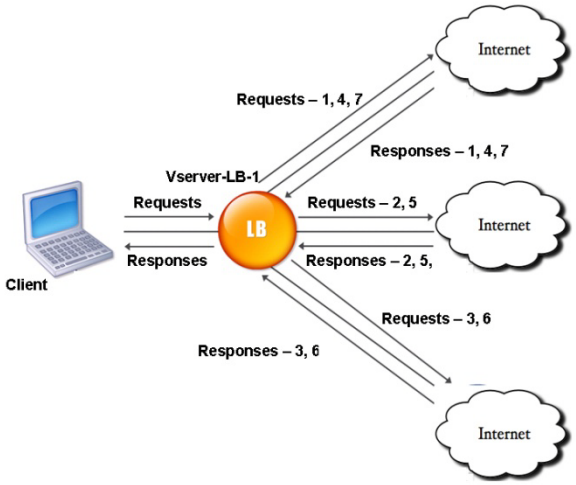
   Metode pembagian beban ini yang menggunakan beberapa alamat IP (*Internet Protocol*) untuk satu nama domain.

b. Random Allocation

   Algoritma ini membagi beban kerja secara acak , jadi tidak ada pola yang jelas dalam pembagian beban. Pada implementasi algoritma ini memungkinkan satu perangkat mendapatkan banyak tugas dan perangkat yang lain menganggur. Namun pada umumnya setiiap perangkat mendapatkan pembagian tugas

c. Round Robin

   Algoritma Round Robin adalah algoritma yang paling umum digunakan untuk Pembagian beban. Algoritma ini lalulintas paket data secara rotasi, sehingga pembagian kerja dari perangkat merata. Algoritma ini menggunakan metode FCFS (*First Come First Serve*). Pada metode FCFS proses yang pertama kali datang akan di layani terlebih dahulu kemudian akan melayani proses berikutnya. Pada Algoritma Round Robin paket data yang datang lebih dulu akan dilayani dengan menggunakan jalur yang pertama, paket berikutnya akan dilayani menggunakan jalur berikutnya, seperti terlihat pada gambar 2.1.




**Gambar 2. 1** Round Robin

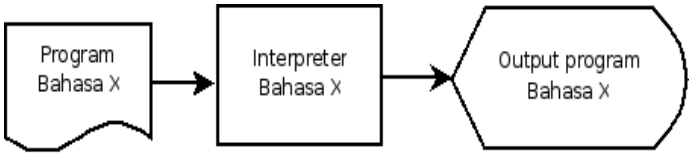
Pada gambar 2.1 menunjukan saat klien mengirimkan paket pertama, paket tersebut akan dilayani oleh jalur yang pertama, saat paket kedua dikirim maka server load balancing akan melewatkan ke jalur kedua, saat dari klien ada pengiriman paket ketiga akan dilewatan jalar yang ke tiga. Proses pengiriman akan dilakukan secara rotasi antara jalur 1, jalur 2 dan jalur 3, jika jalur masih dalam kondisi digunakan maka paket akan di kirimkan lewat jalur berikutnya.

## 2.2 Linux

Linux adalah sebuah sistem operasi yang dikembangkan oleh Linus Benedict Torvalds. Linux merupakan sistem operasi yang disebarluaskan dibawah lisensi GPL (*General Public License*). Linux berrsifat multiuser, multitasking, multiconsole, freeware, dan free source. Sistem operasi Linux dapat berjalan di berbagai platform komputer.[ANO11]

### 2.2.1 Python

Python adalah bahasa pemrograman tingkat tinggi seperti Pascal, Perl, Java, dan sebagainya. Python merupakan bahasa pemrograman tipe interpreted karena pemrograman python langsung dieksekusi oleh interpreter tanpa melalui kompilasi terlebih dahulu.[EMA04] Interpreter langsung mengartikan baris program, sedangkan kompilasi proses dimana baris program diubah ke bahasa yang dapat dimengerti oleh komputer terlebih dahulu sebelum program dijalankan. Dalam proses eksekusi bahasa python program tidak akan di kompilasi tetapi akan langsung diartikan ole terminal seperti telihat pada gambar 2.2.




**Gambar 2. 2** Langkah Interpreter[YOH11]

Python dapat dijalankan dengan 2 cara (mode) :

a. Mode command-line

Pada mode ini statement langsung ditulis ke dalam interpereter dan interpreter akan langsung menampilkan hasil keluaran dari statement tersebut. [EMA04]

b. Mode script

Pada mode script, seluruh statement dituliskan dalam sebuah file berekstensi *.py*. Interpreter akan mengeksekusi seluruh isi file, isi file tersebut dibaca urut dari atas ke bawah.[EMA04]

### 2.2.2 Shell

Shell adalah program penerjemah perintah yang menjembatani pengguna dengan sistem operasi. Pada umumnya shell menyediakan prompt sebagai antarmuka. Pengguna mengetikkan perintah-perintah pada shell di prompt. [SYA10] Perintah dapat berupa perintah-perintah internal shell, ataupun perintah eksternal untuk mengeksekusi suatu program. Table 2.1 menunjukkan jenis-jenis shell yang ada di dalam sistem operasi.

| **Nama Shell** | **Developer** | **Keterangan** |
|---|---|---|
| sh (Bourne Shell) | | Shell Sederhana, belum dilengkapi fitur seperti auto-completion |
| CSH (C Shell) | Bill Joy | Biasa disertakan bersama keluarga BSD |
| KSH (Korn Shell) | David Korn | Pengambangan dari Bourne Shell dan C Shell |
| BASH (Bourne Againt Shell) | Brian Fox dan Chet Ramey | Shell yang banyak digunakan di linux |
| TCSH | | Kompatibel dengan C Shell yang sudah dilengkapi dengan fitur command-line-completion dan command-line editing |
| ZSH (Z Shell) | | kompatible dengan C Shell, Korn Shell, dan Bourne Shell |

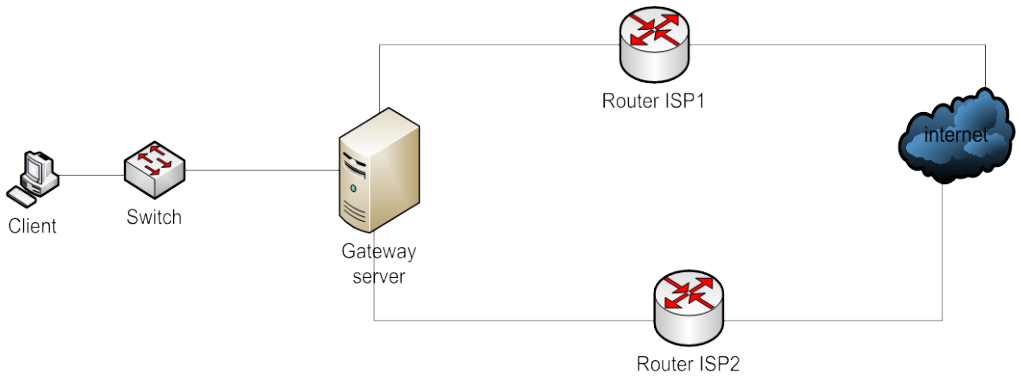
**Tabel 2. 1** Jenis-jenis Shell

Prinsip kerja shell adalah membaca dan mengartikan apa yang dimasukkan pengguna melalui input standart yaitu keyboard. Jika perintah tersebut dimengerti oleh shell makan shell akan menampilkan hasil eksekusi perintah yang dimasukkan, jika tidak maka shell akan memunculkan pesan error.

Pada aplikasi yang dibangun shell script digunakan untuk menjalankan perintah-perintah pembagian beban yang ada di sistem operasi Linux. Selain untuk menjalankan perintah pembagian beban shell script juga digunakan pada file daemon dan juga pada file-file konfigurasi pembagian beban.

### 2.2.3 Pembagian beban di Linux

Dalam sistem operasi Linux pembagian beban merupakan proses menjalankan routing paket data ke beberapa jalur koneksi yang ada, jalur koneksi yang ada lebih dari 1 jalur seperti ditunjukkan pada gambar 2.3. Secara umum, kerja pembagian beban dapat ditunjukan pada gambar 2.4 sebagai berikut.




**Gambar 2. 3** Multiple Connection Server

Pembagian beban dilakukan dengan memanipulasi tabel routing yang ada. Manipulasi table routing pada sistem operasi Linux menggunakan perintah ip route.

```
ip route add default scope global
nexthop via $P1 dev $IF1 weight 1 \
nexthop via $P2 dev $IF2 weight 1 \
nexthop via $P3 dev IF3
```

**Gambar 2. 4** perintah ip route untuk pembagian beban

Keterangan :

P1 = IP address dari gateway inet1

P2 = IP address dari gateway inet2

P3 = IP address dari gateway inet3

IF1 = interface card 1

IF2 = interface card 2

IF3 = interface card 3

Perintah pada gambar 2.5 akan menggabungkan jalur, semua permintaan paket yang berada di network local akan di jalurkan ke IP gateway masing-masing interface, sedangkan parameter weight di beri nilai 1 berfungsi agar semua paket yang lewat memiliki prioritas sama melalui interface $IF1, $IF2, maupun $IF3. Jadi jika interface $IF1 sedang melakukan layanan permintaan paket jika ada permintaan paket berikutnya akan ditangan oleh interface $IF2. Tetapi disini tetap memperhatikan cache dari paket tersebut, jika paket tersebut pernah diminta melalui interface $IF1 maka pada permintaan berikutnya paket tetap akan melalui interface yang sama yaitu melalui interface $IF1.

# BAB III
# METODOLOGI

## 3.1 Analisis Kebutuhan Perangkat Lunak

### 3.1.1 Metode Analisis

Tahap analisis digunakan untuk mengetahui kebutuhan dalam pembuatan aplikasi yang akan dibangun. Metode yang digunakan adalah metode analisi terstruktur dengan metode diagram alir. Metode ini memiliki alur yang jelas sehingga aplikasi yang dibuat adalah aplikasi yang detail.

### 3.1.2 Hasil Analisis

#### 3.1.2.1 Kebutuhan Masukan

Berdasarkan analisis yang dilakukan kebutuhan masukan dari aplikasi yang akan dibuat antara lain :

1. Jumlah Interface yang ada.
2. IP Address router ISP.
3. IP Broadcast router ISP.
4. IP Gateway router ISP.
5. Interface yang terkoneksi ke klien.

#### 3.1.2.2 Kebutuhan Proses

Proses-proses yang dilakukan dalam library :

1. Proses pembuatan file konfigurasi.
2. Proses pembuatan file daemon.
3. Pengambilan data jumlah Ethernet.
4. Pengambilan data ping time.
5. Penghitungan rata-rata dari ping time.
6. Proses manipulasi routing table.

a. Proses pengisian rt_table.
b. Proses flushing routing table.
c. Proses penambahan prioritas.
d. Proses delete routing table default.
e. Proses pembuatan routing table baru.
f. Proses pembagian beban.

#### 3.1.2.3 Kebutuhan Keluaran

Keluaran aplikasi adalah perubahan tabel *routing*.

#### 3.1.2.4 Perangkat Lunak Yang Dibutuhkan

Perangkat lunak yang dibutuhkan untuk membuat dan menjalankan library ini adalah sebagai berikut :

1. GNS3

   GNS3 atau *Graphical Network Simulator* merupakan aplikasi emulator untuk jaringan yang kompleks. GNS3 dilengakapi dengan beberapa fitur antara lain :

   - Dynamips, yang merupakan inti dari GNS3 yang memampukan aplikasi ini dapat mengemulasikan Cisco IOS.
   - Dynagen, merupakan *text-based frontend* untuk Dynamips
   - Qemu, merupakan aplikasi untuk melakukan virtualisasi

2. VMware

   VMware adalah aplikasi untuk melakukan virtualisasi terhadap sebuah komputer. VMware memvirtualisasikan perangkat keras yang ada di dalam sebuah komputer. Dengan VMware sebuah komputer dapat menjalankan beberapa sistem operasi sekaligus secara bersamaan.

3. Ubuntu Server

   Ubuntu Server digunakan untuk melakukan pengujian perangkat lunak yang dibangun. Karena perangkat lunak yang dibangun adalah perangkat lunak yang berbasiskan bahasa python dan sistem operasi linux. Versi Ubuntu Server yang digunakan adalah Ubuntu Server 10.04

4. Ubuntu Desktop

   Ubuntu Desktop digunakan untuk membangun perangkat lunak yang dibuat, karena perangkat lunak yang akan dibuat berbasis Linux. Versi ubuntu desktop yang digunakan adalah Ubuntu 10.04

5. Stany's Phyton Editor

   Stany's Phyton Editor atau SPE adalah sebua IDE (*integrated development environtment*) untuk bahasa pemrograman phyton. SPE dapat berjalan di berbagai platform sistem operasi. Editor ini mempunyai fitur auto-complete layaknya editor seperti Netbeans.

6. Python 2.6.6

   Python merupakan interpreter dari bahasa pemrograman Python.

### 3.2 Perangkat Keras yang Dibutuhkan

Dalam melakukan pembuatan aplikasi, diperlukan perangkat keras untuk melakukan uji coba aplikasi. Perangkat keras yang dibutuhkan antara lain :

1. Komputer

Komputer digunakan untuk melakukan uji coba aplikasi yang sudah dibuat. Dengan membuat 2 buah mesin virtual berbasis OS ubuntu server dan ubuntu desktop sebagai mesin klien. Pada virtualisasi jaringan menggunakan emulator GNS3 dengan 3 buah router cisco berbasis IOS 7200.

## 3.3 Perancangan Perangkat Lunak

### 3.3.1 Metode Perancangan

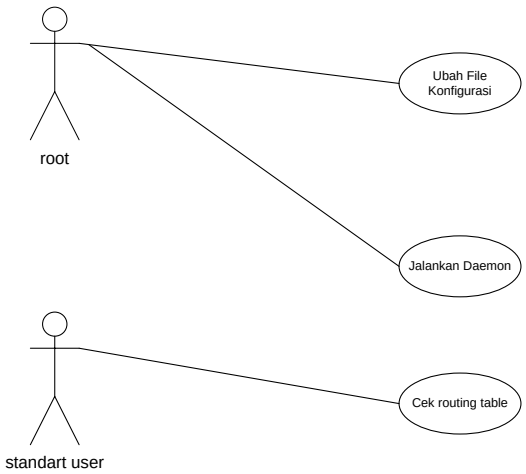
Metode perancangan yang digunakan adalah Flowchart dan UML ( Unified Modeling Language). UML yang digunakan yaitu use case diagram yang digunakan untuk menggambarkan peran user terhadap aplikasi yang dibuat.

### 3.3.2 Hasil Perancangan

#### 3.3.2.1 UML

**a. Use Case Diagram**

Use case diagram merupakan bagian dari diagram UML. Use case memodelkan sistem berdasarkan perspektif pengguna perangkat lunak. Diagram use case terdiri dari use case itu sendiri dan actor. Actor merepresentasikan pengguna dari perangkat lunak tersebut. Dalam aplikasi yang dibuat, use case memodelkan bagaimana hak-hak user biasa dan super user (root) dalam mengakases aplikasi yang dibuat, seperti digambarkan pada gambar 3.1.



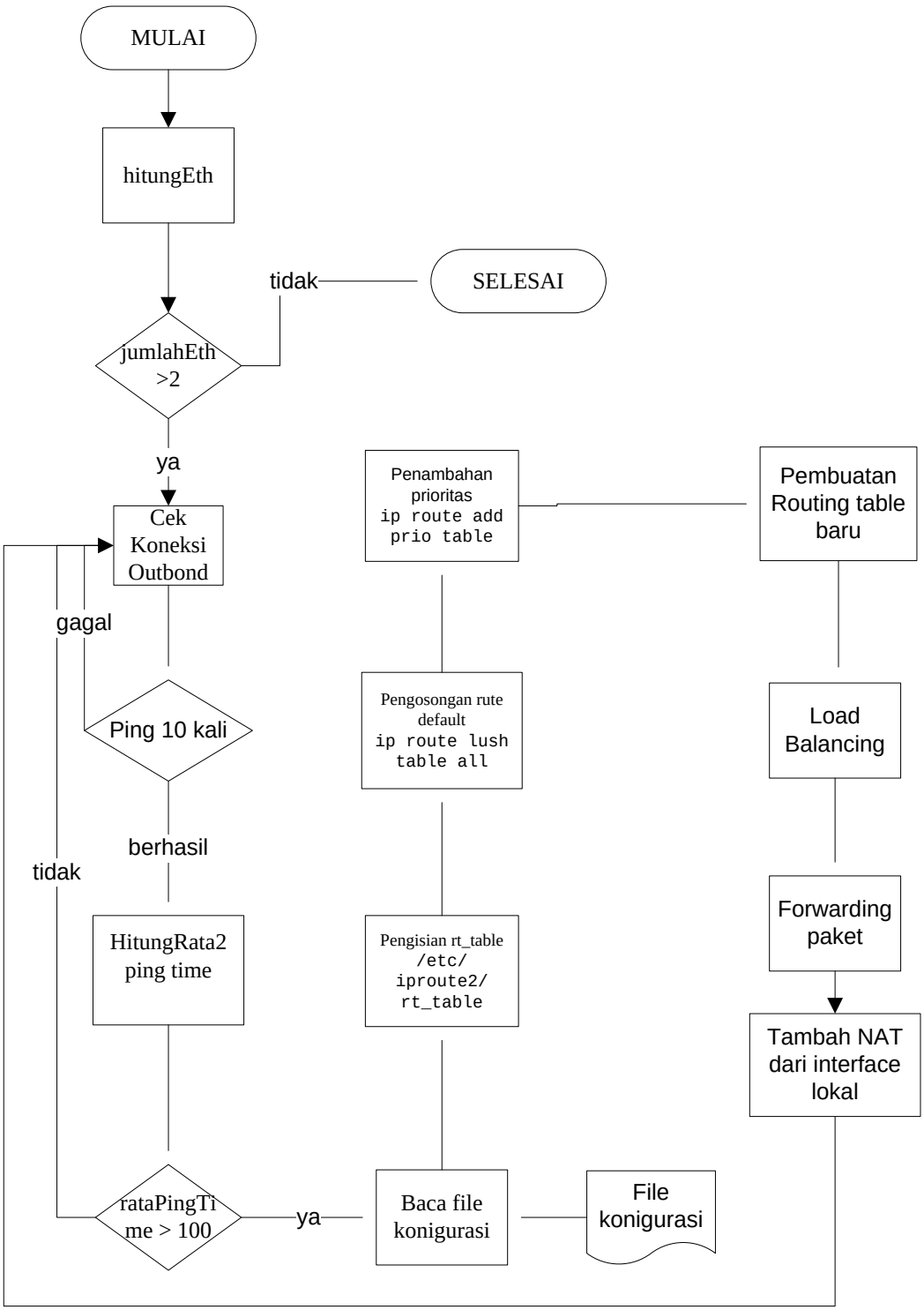

**Gambar 3. 1** *Use Case Diagram*

Gambar 3.1 root memiliki peran dalam mengubah file konfigurasi, menjalankan file daemon, dan mengecek tabel routing. Sedangkan standart user hanya dapat melihat tabe routing saja.

#### 3.3.2.2 Flowchart

Flowchart merupakan suatu diagram alir untuk menggambarkan aliran proses atau algoritma dari suatu perangkat lunak. Diagram alir ini menggambarkan bagaimana menyelsaikan suatu masalah. Pada pembuatan aplikasi ini flowchart digunakan untuk mengetahui aliran proses dari aplikasi yang dibuat. sehingga terlihat urutan proses-proses yang terjadi.

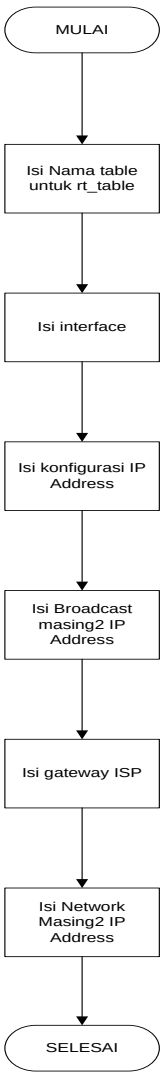
Seperti terlihat pada gambar 3.2 diagram dimulai dengan terminator untuk memulai proses. Dalam diagram alir tersebut terdapat beberapa bentuk bujursangkar untuk menggambarkan proses yang terjadi, belah ketupat untuk meggam barkan proses pengambilan keputusan.



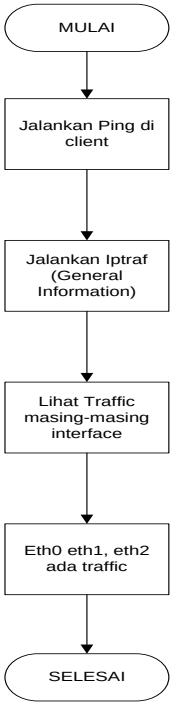

**Gambar 3. 2** *Flowchart*

### 3.4 Rencana Pengujian

Pengujian akan dilakukan dengan mencoba menggunakan satu koneksi kemudian dilihat kualitas koneksinya dengan mengukur nilai latensi. Latensi dinilai buruk jika nilainya di diatas 20 milisecond, pada flowchart digunakan koneksi yang sebenarnya sehingga latensinya menggunaan nilai 100. Lalu pengujian dilakukan dengan menjalankan aplikasi pembagian beban dan dilihat kualitas koneksinya dengan mengukur latensinya juga. Flowchart pengisian file konfigurasi dapat dilihat di gambar 3.3, dan flowchart pengujian dapat dilihat di gambar 3.4.




**Gambar 3. 3** Flowchart Pengisian File Konfigurasi



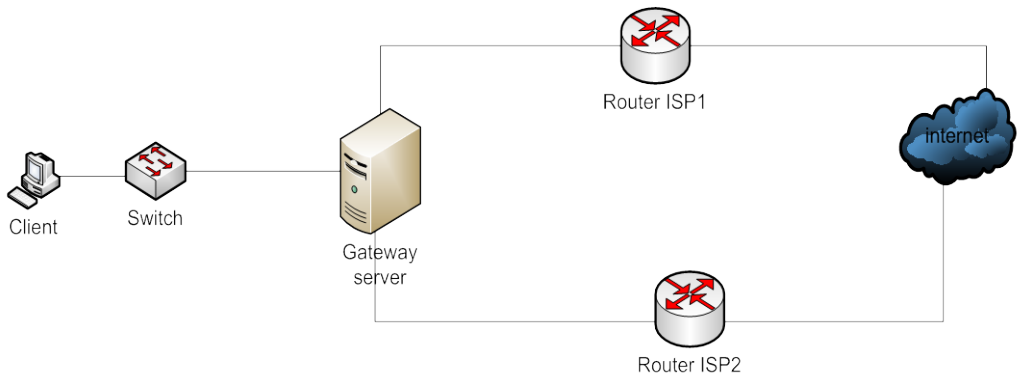

**Gambar 3. 4** Flowchart Pengujian

# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1 Implementasi

Implementasi dilakukan untuk memastikan perangkat lunak yang dibuat dapat bekerja dengan baik sesuai dengan tujuan dibuatnya perangkat lunak tersebut. Dalam proses implementasi ini dilakukan pembangunan perangkat lunak. Perangkat lunak yang dibuat merupakan kombinasi dari 2 bahasa pemrograman yaitu Shell dan python.

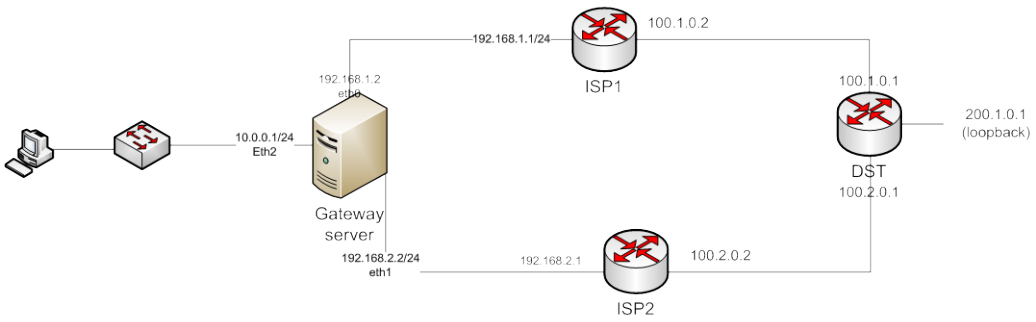
Implementasi yang paling ideal adalah menggunakan 2 koneksi dari ISP yang berbeda sekaligus seperti terlihat di gambar 4.1, karena jika menggunakan koneksi dari ISP yang sama maka tidak akan terjadi perubahan kualitas.




**Gambar 4. 1** Arsitektur Ideal

Implementasi yang dilakukan dalam pengerjaan tugas akhir ini menggunakan konsep virtualisasi. Implementasi dilakukan dengan melakukan pembangunan jaringan ISP secara virtual dengan menggunakan GNS3. Jaringan virtual ISP dibangun untuk mensimulasikan koneksi dari gateway ke ISP seperti pada koneksi sesungguhnya. Sedangkan untuk pembuatan server

pembagian beban dan klien digunakan VMware. Arsitektur yang dibangun untuk melakukan pengujian dapat dilihat pada gambar 4.2




**Gambar 4. 2** Arsitektur Implementasi

Pada gambar 4.2 diatas klien dengan ipaddress 10.0.0.2/24 terhubung ke gateway server dengan ipaddres 10.0.0.1/24 melalui interface eth2. Gateway terhubung ke 2 buah ISP melalui interface eth0 dan eth1 dengan ipaddress masing-mmasing adalah 192.168.1.2/24 dan 192.168.2.2/24.

### 4.1.1 Implementasi Proses

#### 4.1.1.1 File Konfigurasi

Dalam sistem operasi Linux ataupun Unix file konfigurasi adalah jantung dari suatu perangkat lunak. Semua hal yang dapat diubah oleh penggunan dapat dilakukan di file konfigurasi.

```
#!/bin/sh
#ini adalah file konfigurasi dari pembagian beban, disini
diletakan konfigurasi interface dan IP Address dari jaringan
yang akan di pembagian beban bagian ini akan diisikan ke file
/etc/iproute2/rt_tables

#!/bin/sh
```

```
#ini adalah file konfigurasi dari pembagian beban, disini diletakan konfigurasi interface dan IP Address dari jaringan yang akan di pembagian beban

#konfigurasi rt_table
RT_main="main"
RT_net1="inet1"
RT_net2="inet2"
RT_Internet="Internet"
```

Pada bagian file konfigurasi seperti dibawah ini diisikan interface apa saja yang ada di dalam server, ditambahkan dengan interface loopback yaitu localhost yang disingkat dengan lo.

```
#konfigurasi interface
#Disini diisikan interface yang ada didalam Gatewa, IFlocalnet
#diisi dengan interface yang terhubung ke jaringan local,
#IFinet1 diisikan interface yang terhubung ke ISP 1, IFinet2
#diisikan interface yang terhubung ke ISP2
IFlocalhost="lo"
IFlocalnet="eth2"
IFinet1="eth0"
IFinet2="eth1"

#konfigurasi IP Address
#disini diisikan masing-masing IP Address dari masing-masing interface
IPlocalhost="127.0.0.1/8"
IPLocalNet="10.0.100.1/24"
IPinet1="192.168.1.2/24"
IPinet2="192.168.2.2/24"
IPnet1="192.168.1.2"
IPnet2="192.168.2.2"
```

Pada konfigurasi IP Address diatas ada beberapa yang harus di konfigurasi, yaitu IP localhost 127.0.0.1, IP localnet yaitu IP yang menuju ke

jaringan lokal, IPinet dan IPnet memliliki perbedaan yaitu jika Ipinet disertakan prefix dari Ip tersebut sedangkan Ipnet hanya menyimpan IP address saja.

```
#konfigurasi Broadcast masing2 ethernet
BRDlocalhost="127.0.0.255"
BRDlocalNet="10.0.100.255"
BRDinet1="192.168.1.255"
BRDinet2="192.168.2.255"

#konfigurasi gateway inet interface
GWinet1="192.168.1.1"
GWinet2="192.168.2.1"
```

#### 4.1.1.2 File Daemon

File daemon merupakan file yang berfungsi untuk menjalankan perangkat lunak dalam sistem operasi Linux. Daemon akan membaca proses yang berjalan dalam sistem operasi dengan melihat PID (Proccess ID). PID merupakan nomor unik yang diberikan sistem operasi kepada proses yang sedang berjalan.

```
#!/bin/bash -e
#
# /etc/rc.d/init.d
#
#

#. /etc/init.d/fucntion
hostname=`whoami`

#if [$hostname="root"]; then

  start(){
        echo "Starting Load Balancing"
        sh /etc/loadBalancer/loadBalancing
        echo "Load Balance started successfull"
  }

  restart(){
        echo "restarting Load Balancing"
        PID_sh=`pidof sh loadBalancing`
        PID_py=`pidof python`
```

```
        kill $PID_sh
        kill $PID_py
        echo "Starting Load Balancing"
        sh /etc/loadBalancer/loadBalancing
}

stop (){
        echo "Stoping Load Balancing"
        PID_shell=`pidof sh load Balancing`
        PID_python=`pidof python`
        kill $PID_shell
        kill $PID_python
}

case $1 in
start)
        start
        ;;
restart)
        restart
        ;;
stop)
        stop
        ;;
reloaad)
        restart
        ;;
esac

#else
# echo "this file must run under root priviledge"
#fi
```

#### 4.1.1.3 Pengambilan Data Jumlah Ethernet

Data jumlah Ethernet diperlukan untuk melakukan pembagian beban karena pembagian beban dapat dilakukan dengan memiliki 2 interface Ethernet untuk koneksi keluar (outbond).

```
def hitungEth():
    count=0
    for line in os.popen("/sbin/ifconfig"):

        if  line.find('eth')>-1://mencari  baris  dengan  string
eth
```

```
            eth=line.split()[0]//memecah baris menjadi string
eth saja
            count+=1

    print "Jumlah ethernet Card = ", count
```

Penghitungan awal di inisialisasikan bahwa jumlah ethernet card adalah 0, kemudian dengan perintah os.popen akan membuka aplikasi ifconfig, kemudian hasil keluaran dari ifconfig akan dicari baris yang mengandung kata eth dengan perintah line.find. Kedua skrip tersebut merupakan perintah bawaan pada pustaka di dalam bahasa pemrograman python.

#### 4.1.1.4 Pengambilan Data Ping Time

Data ping time diambil dengan membuat sebuah program dengan bahasa pemrograman python. Ping dilakukan sebayak 10 kali ping untuk melihat perubahan data yang terjadi dalam 10 kali ping. Nilai 10 diisikan kedalam parameter count yang ada dialam fungsi ping_host.

```
__author__="cah-mendem"
__date__ ="$Apr 10, 2011 10:43:50 PM$"
#originally taken from g-loaded.com
import os
import sys
import socket
import struct
import select
import time

# berdasarkan pemantauan di /usr/include/Linux/icmp.h nilai
ICMP_ECHO_REQUEST
ICMP_ECHO_REQUEST = 8
```

Kode di bawah ini adalah kode untuk melakukan pengecekan terhadap nilai denga ping.c yang ada pada sistem operasi yang terpasang pada server.

```
def checksum(source_string):


    jum = 0
    count = (len(source_string)/2)*2 #menghitung panjang suatu
objek
    counter = 0
    while counter<count:
        thisVal  =  ord(source_string[counter  +  1])*256  +
ord(source_string[counter])
        jum = jum + thisVal
        jum = jum & 0xffffffff # Necessary?
        counter = counter + 2

    if count<len(source_string):
        jum = jum + ord(source_string[len(source_string) - 1])
        jum = jum & 0xffffffff # Necessary?

    jum = (jum >> 16)  +  (jum & 0xffff)
    jum = jum + (jum >> 16)
    reply = ~jum
    reply = reply & 0xffff

    # Swap bytes.
    reply = reply >> 8 | (reply << 8 & 0xff00)

    return reply



def receive(new_socket, ID, timeout):

    timeLeft = timeout
    while True:
```

```
        startedSelect = time.time()
        whatReady   =   select.select([new_socket],   [],   [],
timeLeft)
        howLongInSelect = (time.time() - startedSelect)
        if whatReady[0] == []: # pengecekan Timeout
            return

        timeReceived = time.time()
        recPacket, addr = new_socket.recvfrom(1024)
        icmpHeader = recPacket[20:28]
        type,    code,    checksum,    packetID,    sequence    =
struct.unpack("bbHHh", icmpHeader)
        if packetID == ID:
            bytesInDouble = struct.calcsize("d")
            timeSent  =  struct.unpack("d",  recPacket[28:28  +
bytesInDouble])[0]
            return timeReceived - timeSent

        timeLeft = timeLeft - howLongInSelect
        if timeLeft <= 0:
            return


def send(new_socket, dest_addr, ID):

    dest_addr  =  socket.gethostbyname(dest_addr)

    # Header is type (8), code (8), checksum (16), id (16),
sequence (16)
    ceksum = 0

    # Make a dummy heder with a 0 checksum.
    header = struct.pack("bbHHh", ICMP_ECHO_REQUEST, 0, ceksum,
ID, 1)
    bytesInDouble = struct.calcsize("d")
    data = (192 - bytesInDouble) * "Q"
    data = struct.pack("d", time.time()) + data
```

```
    # Calculate the checksum on the data and the dummy header.
    ceksum = checksum(header + data)

    # Now that we have the right checksum, we put that in. It's
just easier
    # to make up a new header than to stuff it into the dummy.
    header   =   struct.pack("bbHHh",   ICMP_ECHO_REQUEST,   0,
socket.htons(ceksum), ID, 1)
    packet = header + data
    new_socket.sendto(packet,  (dest_addr,  1))  #  Don't  know
about the 1


def ping_one(dest_addr, timeout):

    icmp = socket.getprotobyname("icmp")
    try:
        new_socket       =       socket.socket(socket.AF_INET,
socket.SOCK_RAW, icmp)
    except socket.error, (errno, msg):
        if errno == 1:
            # Operation not permitted
            msg = (
                " harus dijalankan dengan root priviledge"
            )
            raise socket.error(msg)
        raise # raise the original error

    ID = os.getpid() & 0xFFFF

    send(new_socket, dest_addr, ID)
    delay = receive(new_socket, ID, timeout)

    new_socket.close()
    return delay
```

Dibawah ini adalah kode untuk melakukan ping ke server. Server yang di simulasikan menjadi sebuah router loopback dengan nama R1 seperti pada gambar 4.2 dengan ipaddress loopback 200.1.0.1, dengan ping yang dilakukan adalah sebanyak 10 kali, dan timeout adalah 2 detik, jika diatas 2 detik gateway tidak mendapatkan balasan, server tujuan atau jaringan dianggap mengalami gangguan.

```
def ping_host(dest_addr, timeout = 2, count = 10, rata_delay =
0):
    rata2_delay=0
    for i in xrange(count):

        print "ping %s ..." % dest_addr,
        try:
            delay  =  ping_one(dest_addr, timeout)

        except socket.gaierror, e:
            print "ping fail (socket error: '%s')" % e[1]
            break

        if delay  ==  None:
            print  "ping  fail.  (timeout  within  %ssec.)"  %
timeout
        else:
            delay  =  delay * 1000
            print "ping in %0.4fms" % delay
            rata2_delay+=delay

    print
    rata_delay=rata2_delay/10
    print "Nilai time total = %s" % rata2_delay
    print "nilai rata-rata adalah %s" % rata_delay


def loadBalance():
    jumLahEth = hitungEth()
```

```
    latencyMax = 30
    if jumLahEth >= 2:
        ping_host("200.1.0.1")
        if rata_delay > latencyMax :
            print "delay = ",rata_delay

subprocess.call(['/etc/loadBalancer/loadBalancingScript'])#call
sh file
            loadBalance()
        else :

           #
subprocess.call(['/etc/loadBalancer/unLoadBalace'])#call     sh
file
            loadBalance()
    else :
        print "jumlah eth belum memenuhi ketentuan"
if __name__ == '__main__':
    loadBalance()
```

Kode diatas adalah kode untuk memanggil aplikasi pembagian beban yang berbentuk shell. File itu dipanggil setelah melakukan penghitungan jumlah interface ethernet.

#### 4.1.1.5 Penghitungan Rata-rata Ping Time

Setelah penghitungan ping time, dilakukan penghitungan rata-rata dari ping time yang ada.

```
def ping_host(dest_addr, timeout = 2, count = 10, rata_delay =
0):
    rata2_delay=0
    for i in xrange(count):

        print "ping %s ..." % dest_addr,
        try:
```

```
            delay  =  ping_one(dest_addr, timeout)

        except socket.gaierror, e:
            print "ping fail (socket error: '%s')" % e[1]
            break

        if delay  ==  None:
            print  "ping  fail.  (timeout  within  %ssec.)"  %
timeout
        else:
            delay  =  delay * 1000
            print "ping in %0.4fms" % delay
            rata2_delay+=delay

    print
    rata_delay=rata2_delay/10
    print "Nilai time total = %s" % rata2_delay
    print "nilai rata-rata adalah %s" % rata_delay
```

Nilai rata-rata ping time pada awalnya di inisialisasikan dengan nilai 0, setelah aplikasi melakukan ping terhadap server tujuan dalam hal server yang digunakan adalah router R1 maka jumlah time dari ping tersebut akan dijumlahkan dan dicari nilai rata-ratanya.

#### 4.1.1.6 Manipulasi Routing Table

**a. Pengisian rt_table**

Rt_tables merupakan file konfigurasi dari iproute2. Dalam file rt_tables diisikan nama-nama dari tabel routing yang dibuat. File rt_tables terletak di file /et/iproute2/rt_tables. File ini berfungsi menyimpan nama nama dari tabel routing yang akan dibuat.

```
#!/bin/bash
```

```
cp /etc/iproute2/rt_tables /etc/iproute2/rt_tables.backup
. ./loadBalance-conf


echo 120 $RT_net1 >> /etc/iproute2/rt_tables
echo 121 $RT_net2 >> /etc/iproute2/rt_tables
echo 123 $RT_Internet >> /etc/iproute2/rt_tables
```

**b. Flushing routing table**

Kode dibawah ini adalah kode untuk melakukan flush terhadap table yang ada. Flush adalah fungsi untuk pengosongan konfigurasi yang sudah dibuat. Pada kode dibawah ini flush dilakukan terhadap tabel pada ip route.

```
ip route flush table $RT_net1
ip route flush table $RT_net2
ip route flush table $RT_Internet
ip addr flush $IFlocalhost
ip addr flush $IFinet1
ip addr flush $IFinet2
```

**c. Penambahan prioritas**

kode ini berfungsi untuk menambahkan prioritas pada setiap routing tabel yang tercantum pada file rt_table. Sehingga pada saat menjalankan routing terhadap paket yang lewat.

```
ip rule add prio 10 table $RT_main

ip rule add prio 20 table $RT_net1

ip rule add prio 30 table $RT_net2

ip rule add prio 40 table $RT_Internet
```

**d. Pembuatan routing table baru**

Disini dilakukan penambahan routing table baru dengan perintah ip route. Pertama dilakukan penambahan aturan dengan kode ip rule bahwa setiap permintaan dari $IPinet akan dimasukkan ke dalam tabel $RT_net, selanjutnya menambahkan routing dengan kode ip route sehingga seluruh permintaan dari interface pertama akan melalui Gateway $GWinet.

```
ip rule add prio 20 from $IPinet1 table $RT_net1
ip route add default via $GWinet1 dev $IFinet1 src $IPnet1
proto static table $RT_net1
ip route append prohibit default table $RT_net1 metric 1 proto
static
```

```
ip rule add prio 30 from $IPinet2 table $RT_net2
ip route add default via $GWinet2 dev $IFinet2 src $IPnet2
proto static table $RT_net2
ip route append prohibit default table $RT_net2 metric 5 proto
static
```

**e. Pembagian beban**

Pembagian beban dilakukan dengan menggunakan bahasa shell. Dengan menambahkan routing ke kedua Gateway dari ISP yang ada. Route ditambahkan dengan fungsi nexthop dan diberikan nilai weight 1,.

```
ip rule add prio 40 table $RT_Internet
#menambahkan rute baru untuk melakukan pembagian beban
ip route add default proto static table $RT_Internet \
nexthop via $GWinet1 dev $IFinet1 weight 1 \
nexthop via $GWinet2 dev $IFinet2 weight 1
```

### f. Forwarding

Forwarding merupakan proses untuk meneruskan paket dari jaringan yang berbeda. Biasanya forwarding dilakukan pada perangkat-perangkat router. Pada sistem operasi Linux forwarding dilakukan dengan memberikan nilai 1 ataupun 0 pada file `/proc/sys/net/ipv4/ip_forward` nilai diberikan dengan dengan menggunakan perintah echo pada bahasa Shell. Nilai 1 berarti true sehingga seluruh paket yang lewat akan di teruskan ke jaringan diatasnya, sedangkan nilai 0 berarti false. Secara default nilai ip_forward bernilai 0.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

### g. NAT ke Interface Outbond

Dalam sistem operasi Linux NAT dilakukan dengan memanipulasi iptables. Iptables akan ditambahkan isinnya agar seluruh permintaan dari interface lokal akan di NAT kedalam kedua interfacae yang di terhubung ke Internet.

```
iptables -F
iptables -P INPUT DROP
iptables -A INPUT -i $IFlocalNet -j ACCEPT
iptables -A INPUT -i $IFinet1 -p tcp -s 0/0 --dport 25 -j
ACCEPT
iptables -A INPUT -i $IFinet2 -p tcp -s 0/0 --dport 25 -j
ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
iptables -A INPUT -p tcp -i $IFinet2 -j REJECT --reject-with
tcp-reset
iptables -A INPUT -p tcp -i $IFinet1 -j REJECT --reject-with
tcp-reset
```

```
iptables -A INPUT -p udp -i $IFinet2 -j REJECT --reject-with
icmp-port-unreachable
iptables -A INPUT -p udp -i $IFinet1 -j REJECT --reject-with
icmp-port-unreachable
```

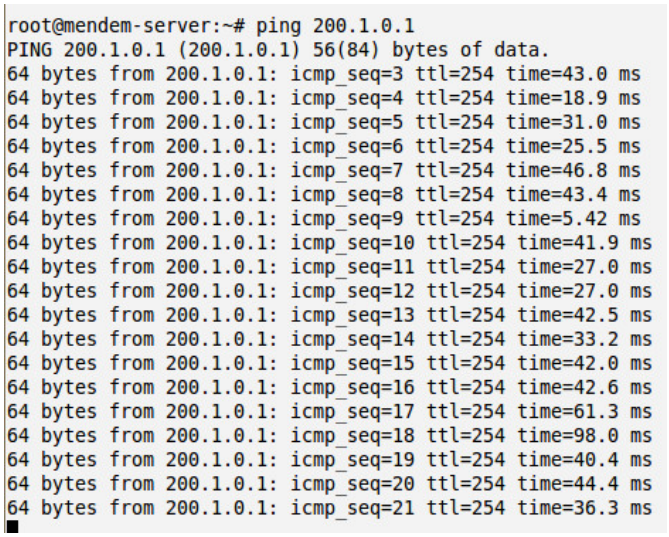
Setelah melakukan flush dan drop seluruh interface, maka akan ditambahkan konfigurasi baru untuk melewatkan paket ke interface yang terhubung ke Internet. Script Shell dari POSTROUTING seperti dibawah ini.

```
iptables -t nat -A POSTROUTING -o $IFinet1 -j SNAT --to $IPnet1
iptables -t nat -A POSTROUTING -o $IFinet2 -j SNAT --to $IPnet2
```

## 4.2 Hasil Implementasi

### 4.2.1 Pengujian Normal

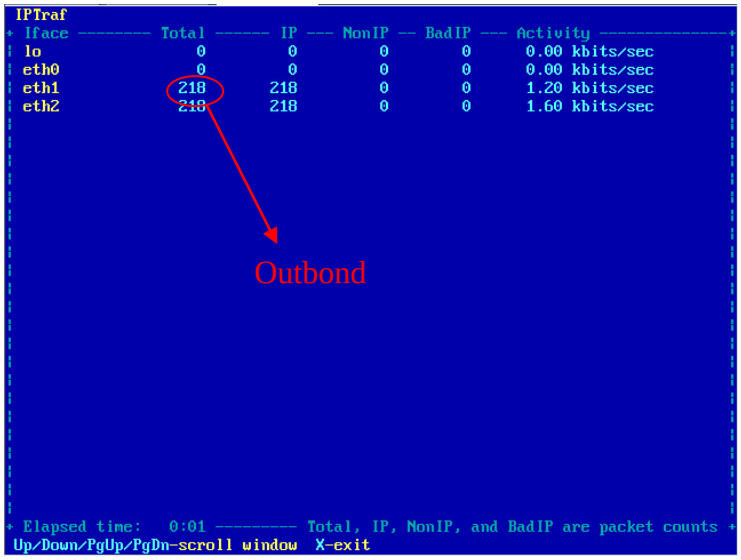
Pengujian normal dilakukan dengan menjalankan aplikasi ping secara normal pada terminal sehingga terlihat aliran paket yang terjadi ketika belum terjadi pembagian beban atau tidak ada aplikasi pembagian beban di server. Proses menjalankan aplikasi ping dilakukan melalui komputer klien yang ada di VMware dan juga menjalankan aplikasi ping di server yang juga ada di dalam virtual machine. Hasil dari pengujian yang dilakukan dilihat melalui aplikasi IPTraf. Pada aplikasi IPTraf akan telihat hanya 1 interface saja yang mengalami penambahan paket ang lewat, sedangkan interface yang lain tidak mengalami peambahan (tetap bernilai 0). Flowchart pengujian dapat dilihat di gambar 3.4




**Gambar 4. 3** Ping Normal

Gambar 4.14 menunjukkan proses ping biasa ke ip loopback yaitu 200.1.0.1. terlihat nilai *latency*. Pada bagian *time*.

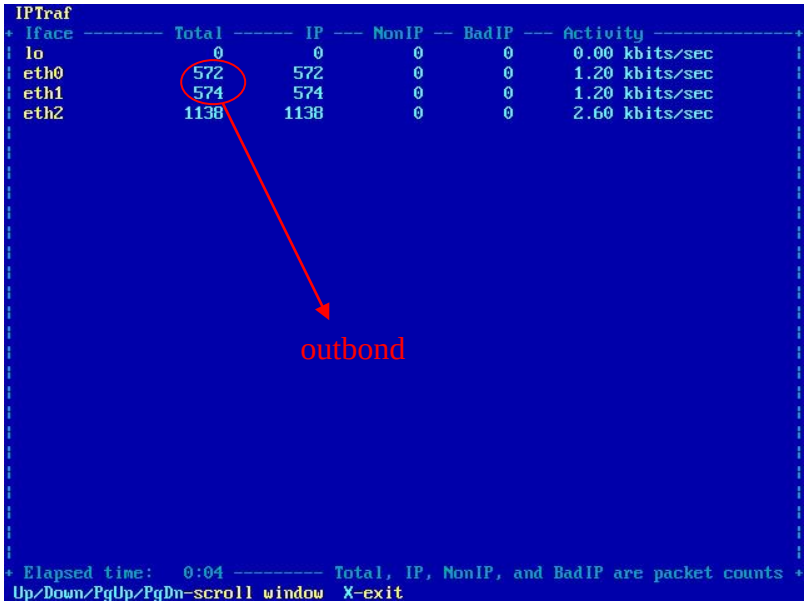
Pada pengujian dengan IPTraf digunakan menu general stasistic. Akan muncul 4 buah interface yang dapat di capture, yaitu lo untuk interface *loopback*, eht0 untuk *interface* eth0 yang menjadi koneksi ke ISP1, eth1 untuk *interface* eth1 yang menjadi koneksi ke ISP2, dan eth2 untuk *interface* eth2 yang menjadi koneksi ke jaringan lokal. (Lihat gambar 4.2)




**Gambar 4. 4** Capture IPTraf Sebelum Load Balancing

Pada gambar 4.15 menunjukkan pembagian beban belum dilakukan terlihat hanya eth1 dan eth2 saja yang mengalami perubahan nilai, karena pembagian beban belum dilakukan sehingga hanya 1 *interface* saja yang mengalami perubahan *traffic*.

Pengujian kedua dilakukan dengan aplikasi pembagian beban sudah terpasang dan pembagian beban terjadi terlihat bahwa kedua interface mengalami penambahan jumlah paket yang lewat.

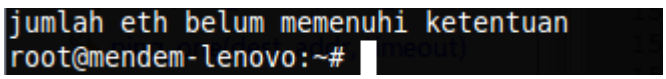



**Gambar 4. 5** Capture IPTraf setelah Load Balancing

Pada gambar 4.16 dapat terlihat keduah interface outbond sudah mengalami perubahan traffic yaitu eth0 dan eth1.

### 4.2.2 Pengujian Tidak Normal

Pada saat dilakukan pengujian tidak normal yang pertama kali dilihat adalah jumlah Ethernet yang terpasang pada komputer. Jika jumlah Ethernet card lebih besar sama dengan 3 maka load balancing bisa dilakukan namun jika jumlah Ethernet card kurang dari 3 maka hasil keluaran akan seperti gambar 4.17 di bawah ini.




**Gambar 4. 6** Pengujian Jumlah Ethernet Kurang Dari ketetuan

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Setelah melakukan serangkaian pengujian dalam pembuatan perangkat lunak ini, maka dapat disimpulkan :

1. Network Administrator dapat melakukan Pembagian beban koneksi secara otomatis
2. File konfigurasi dapat diubah-ubah sehingga perangkat lunak dapat digunakan di berbagai server berbasis Linux

## 5.2 Saran

Hasil penelitian tugas akhir ini masih tentunya masih jauh dari sempurna oleh karena itu untuk pengembangan perangkat luna dimasa mendatang ditambahkan hal-hal sebagai berikut :

1. Pembuatan perangkat lunak menggunakan bahasa pemrograman python tanpa ada campuran dari bahasa pemrograman lain, agar dihasilkan program yang beroriantasi objek.
2. Pembuatan aplikasi yang dapat menangani kapasitas beban koneksi yang tidak sama.
3. Pembuatan file konfigurasi yang dapat dikelola menggunakan GUI ( Graphical User Interface)

# DAFTAR PUSTAKA


[DEN08] Denny. *Pengenalan Server Load Balancing.* http://dennycharter.wordpress.com/2008/07/25/pengenalan-server-load-balancing/ di akses tanggal : 24 Febuari 2011

[ANO11] Anonymous. *Load Balancing (Computing).* http://en.wikipedia.org/wiki/Load_balancing_(computing) di akses tanggal : 24 Febuari 2011

[ANO11] Anonymous. *Linux.* http://en.wikipedia.org/wiki/Linux di akses tanggal 24 Febuari 2011

[ONN07] Purbo, Onno W. *Membuat Router Load Balancing Menggunakan Linux Ubuntu.* http ://portal.cbn.net.id/cbprtl/cybertech/ detail.aspx? x=Tech+Talk&y=cybertech|0|0|3|6 di akses tanggal : 15 Maret 2011

[YOH11] Yohanes. *Interpreter.* http://yohan.es/compiler/interpreter.png di akses tanggal : 24 Febuari 2011

[SYAM10] M, Syamsudin. 2010. *60 Menit Belajar Script Shell.60 Menit Belajar Script Shell.* Yogyakarta : Penerbit Andi

[EMA04] Utami, Ema & Raharjo, Suwanto. 2004. *Logika Algoritma dan Implementasinya dalam Bahasa Python di GNU/Linux.* Yogyakarta : Penerbit Andi.