KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER PALCOMTECH PALEMBANG

SKRIPSI

DESAIN DAN IMPLEMENTASI PENGATURAN *HTB-TOOLS* DAN *CRONJOB* MENGGUNAKAN *SHELL*

Diajukan Oleh :

JOHNSON

011080265

Untuk Memenuhi Sebagian Dari Syarat-syarat

Guna Mencapai Gelar Sarjana Komputer

PALEMBANG

2012

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
PALCOMTECH PALEMBANG

HALAMAN PENGESAHAN PEMBIMBING SKRIPSI

NAMA	**:**	**Johnson**
NOMOR POKOK	**:**	**011080265**
PROGRAM STUDI	**:**	**Teknik Informatika**
JENJANG PENDIDIKAN	**:**	**Strata Satu (S1)**
KONSENTRASI	**:**	**Jaringan Komputer**
JUDUL SKRIPSI	**:**	**Desain dan Implementasi pengaturan *HTB-Tools* dan *Cronjob* menggunakan *Shell***

Tanggal : 4 Agustus 2012

Pembimbing :

Mengetahui,

Ketua,

Rudi Sutomo, S.Kom, M.Si

NIDN : 0222057501

Rudi Sutomo, S.Kom, M.Si

NIP : 028.PCT.08

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
PALCOMTECH PALEMBANG

HALAMAN PENGESAHAN PENGUJI

NAMA : **Johnson**

NOMOR POKOK : **011080265**

PROGRAM STUDI : **Teknik Informatika**

JENJANG PENDIDIKAN : **Strata Satu (S1)**

KONSENTRASI : **Jaringan Komputer**

JUDUL SKRIPSI : **Desain dan Implementasi pengaturan *HTB-Tools* dan *Cronjob* menggunakan *Shell***

Tanggal	**: 22 September 2012**	**Tanggal**	**: 22 September 2012**
Penguji 1	**:**	**Penguji 2**	**:**
Febrianty, SE, M.Si		**Atin triwahyuni,S.T.,M.Eng.**	
NIDN : 0013028001		**NIDN : 0215028002**	

Menyetujui,

Ketua,

Rudi Sutomo, S.Kom., M.Si.

NIP : 028.PCT.08

MOTTO :

KEMALASAN adalah penyebab utama

KEGAGALAN dalam berkarya.

Karya tulis ini kepersembahkan kepada :

- *Kedua Orangtuaku tercinta*
- *Saudara-saudaraku tersayang*
- *Para Pendidik yang kuhormati*
- *Teman-temanku yang selalu memberikan semangat*

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, karena berkat rahmat dan berkah yang diberikan-Nya penulis dapat menyelesaikan penyusunan dan penulisan Skripsi ini yang berjudul "Desain dan Implementasi Pengaturan *HTB-Tools* dan *Cronjob* menggunakan *Shell*" yang merupakan salah satu syarat guna mencapai gelar Sarjana Komputer dengan program studi Teknik Informatika pada Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) PalComTech Palembang.

Dalam penyusunan Skripsi ini, penulis menyadari sepenuhnya bahwa penulis mendapatkan banyak bantuan dan dukungan dari berbagai pihak, baik dari pihak akademik, keluarga, maupun sahabat. Oleh karena itu penulis mengucapkan banyak terima kasih yang tulus atas segala bimbingan, nasehat-nasehat, dan motivasi yang diberikan baik secara langsung atau tidak langsung dalam penulisan Skripsi ini terutama kepada Rudi Sutomo, S.Kom, M.Si, selaku ketua STMIK PalComTech, D. Tri Octafian, S.Kom, M.Kom, selaku Kaprodi Teknik Informatika dan sebagai Dosen Pembimbing yang telah meluangkan waktu dan pikiran dalam membantu penulisan Skripsi ini.

Kritik dan saran yang sifatnya membangun juga diharapkan oleh penulis, dan selepas dari segala kekurangan yang ada semoga Skripsi ini dapat memberikan manfaat bagi kita semua.

Palembang, _____20__

Penulis

DAFTAR ISI

Nama Halaman **Hal**

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRACT

These days, the use of computer and computer network is a necessity. That's why bandwidth management holds an important role for performance improvement in office activities.

HTB-tools is a software in Linux operating system, that can provide dynamic bandwidth sharing. To run HTB-tools services, can be done by running HTB-tools commands from Linux terminal. In addition, in Linux operating system, there's also an utility that is called cron daemon, which is used to run a command line or a program at a certain time repeatedly according to the determined schedule. If the use of HTB-tools and cron daemon are combined, we can create a bandwidth management that is not only dynamic but also scheduled.

Based on the idea, the author attempt to develop a software that can simplify the user in combining HTB-tools and cron daemon using shell programming. By using this program, the author hope that users can perform a more effective and efficient bandwidth management.

Keywords*: *Bandwidth, HTB-tools, Cron, Crontab, Shell

ABSTRAK

Penggunaan komputer dan jaringan komputer pada perkantoran zaman ini sangat dibutuhkan. Oleh karena itu, manajemen *bandwidth* memegang sebuah peran penting dalam peningkatan kinerja didalam kegiatan perkantoran.

HTB-tools adalah sebuah perangkat lunak didalam sistem operasi *Linux*, yang bisa melakukan pembagian *bandwidth* secara dinamis. Menjalankan servis dari *HTB-tools* bisa dilakukan dengan cara menjalankan perintahnya melalui terminal. Selain itu, didalam sistem operasi *Linux* juga terdapat sebuah utilitas bernama *cron daemon* yang berfungsi untuk menjalankan sebuah baris perintah program pada waktu tertentu dan secara berulang sesuai dengan jadwal yang ditentukan. Jika keduanya, *HTB-tools* dan *cron daemon* digabungkan, maka bisa diciptakan sebuah sistem manajemen *bandwidth* yang selain dinamis juga terjadwal.

Berdasarkan ide tersebut, penulis berusaha membangun sebuah perangkat lunak yang bisa mempermudah pengguna dalam menggabungkan *HTB-tools* dan *cron daemon* dengan menggunakan pemrograman *shell*. Dengan program ini, penulis berharap pengguna bisa melakukan sebuah manajemen *bandwidth* yang lebih efektif dan efisien.

Kata Kunci : *Bandwidth, HTB-tools, Cron, Crontab, Shell*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penggunaan komputer pada zaman modern ini menjadi hal yang sangat dibutuhkan dalam kegiatan perkantoran. Hal ini dikarenakan pemrosesan dan penyimpanan berkas dan data dibuat menjadi lebih mudah, lebih cepat, dan lebih hemat dengan menggunakan sistem komputer. Selain penggunaan komputer secara individual, penggunaan jaringan komputer dan internet juga berperan penting dalam pengiriman berkas dan data dari sebuah kantor ke kantor lain secara cepat dan efisien.

Dalam penggunaan jaringan komputer, pembagian *bandwidth* merupakan hal yang sangat penting. Biasanya, bila sebuah jaringan komputer tidak diterapkan pembagian *bandwidth* bisa terjadi pengkonsumsian jalur transmisi data yang tidak adil. Artinya satu komputer klien bisa menggunakan seluruh kapasitas *bandwidth* jaringan tanpa membaginya ke komputer klien lain yang membutuhkan, sehingga terjadi kemacetan pada transmisi data atau berkas.

Pembagian besar *bandwidth* kepada masing-masing komputer klien bisa dilakukan dengan dua metode, yakni statis dan dinamis. Sebelum metode pembagian secara dinamis digunakan, besar *bandwidth* yang sudah ditetapkan untuk masing-masing komputer klien secara statis tidak dapat berubah-ubah dengan otomatis. Jadi, bila terdapat komputer klien yang tidak digunakan, *bandwidth* yang telah dialokasikan kepadanya akan

tersia-siakan. Tetapi, setelah ditemukannya pembagian dengan menggunakan metode dinamis, *bandwidth* yang telah dialokasikan kepada komputer klien yang sedang tidak digunakan bisa dipinjamkan kepada komputer klien yang sedang membutuhkan. Dan terdapat sebuah program pada sistem operasi *Linux* yang dapat melakukan pembagian *bandwidth* secara dinamis, yakni *HTB-tools*.

Pengalokasian besar *bandwidth* secara dinamis tentu saja sangat membantu dalam pembagian *bandwidth* agar bisa digunakan secara efisien, tetapi itu saja tidak cukup. Terkadang kebutuhan pemakaian *bandwidth* oleh komputer-komputer klien dalam sebuah jaringan komputer bisa berubah pada kurun waktu tertentu. Sehingga, selain dibutuhkannya pembagian *bandwidth* dengan metode dinamis, juga diperlukan pembagian *bandwitdh* secara terjadwal. Didalam sistem operasi *Linux* juga ada program yang bisa menjalankan program lain sesuai dengan waktu yang telah ditentukan, program tersebut adalah *Crontab*.

Dikarenakan untuk membuat kedua program tersebut bekerja sama cukup rumit, dan keduanya dijalankan didalam terminal *Linux,* maka penulis mengusulkan ide mengenai sebuah aplikasi yang mempermudah dan mempercepat pengguna untuk melakukan pengaturan kedua program tersebut. Sehingga didapatkanlah karya tulis sebagai skripsi dengan judul "Desain dan Implementasi pengaturan *HTB-Tools* dan *Cronjob* menggunakan *Shell*".

1.2 Rumusan Masalah

Berdasarkan uraian dalam latar belakang masalah, rumusan masalah yang diangkat adalah sebagai berikut : “Bagaimana cara membangun aplikasi yang bisa mempermudah pengguna untuk membuat *HTB-Tools* dan *Crontab* bekerja sama dalam penanganan pembagian *bandwidth* jaringan komputer?”.

1.3 Batasan Masalah

Dalam penulisan skripsi ini, penulis membatasi masalah yang akan dibahas hanya pada pembuatan dan penggunaan aplikasi pengaturan *HTB-Tools* dan *Crontab* dengan menggunakan *shell* didalam sistem operasi *Linux*, dimana fungsi-fungsi yang dimiliki aplikasi ini meliputi :

1) Fungsi yang membantu pengguna dalam membuat berkas konfigurasi *HTB-tools* yang nantinya akan dijadikan sebagai dasar kebijakan dalam pembagian *bandwidth* jaringan.
2) Fungsi yang membantu pengguna dalam memasukkan jadwal pengaktifasian *HTB-tools* pada *interface* tertentu dengan berkas konfigurasi yang ditentukan oleh pengguna kedalam tabel *cron daemon*.
3) Fungsi yang melakukan pengaktifan berkas konfigurasi yang dibuat pengguna menggunakan program yang dikembangkan penulis, menjadi berkas konfigurasi *HTB-tools* sebagai dasar pembagian *bandwidth* jaringan.

1.4 Tujuan Penelitian

Adapun tujuan dari penulisan skripsi ini adalah untuk menghasilkan sebuah aplikasi yang berfungsi untuk mempermudah penggabungan dari *HTB-tools* dan *crond* dimana *HTB-tools* berfungsi untuk melakukan manajemen *bandwidth* dan *crond* berfungsi untuk melakukan pengaktifan pekerjaan didalam sistem sesuai waktu yang sudah ditetapkan.

1.5 Manfaat Penelitian

Adapun manfaat dari penulisan skripsi ini sebagai berikut :

1.5.1 Bagi Peneliti

1) Dapat menerapkan ilmu yang telah dipelajari pada perkuliahan agar bermanfaat bagi masyarakat luas.
2) Dapat mengetahui lebih dalam tentan pemrograman *shell*, sehingga bisa mengembangkan ilmu dan potensi diri yang dimiliki.

1.5.2 Bagi Akademik

Dapat dijadikan sebagai referensi bagi mahasiswa lainnya untuk dijadikan sebagai bahan perbandingan dalam menyusun penelitian yang dapat dikembangkan lebih lanjut.

1.5.3 Bagi Pengguna

1) Dapat mempermudah pengaturan *HTB-tools* untuk manajemen *bandwidth* jaringan komputer.

2) Dapat mempermudah dalam pembuatan sebuah manajemen *bandwidth* secara dinamis dan terjadwal dengan menggunakan *cron* dan *HTB-tools*.

1.6 Sistematika Penulisan

Untuk memperoleh gambaran yang jelas didalam penyusunan skripsi ini, secara garis besar penulis membaginya kedalam enam bab, yaitu :

BAB I PENDAHULUAN

Didalam bab ini akan dijelaskan latar belakang pemilihan judul, perumusan masalah, batasan masalah, tujuan dan manfaat yang diperoleh dan sistematika penulisan.

BAB II PERANGKAT LUNAK YANG DIKEMBANGKAN

Didalam bab ini akan diberikan penjelasan mengenai perangkat lunak yang dikembangkan.

BAB III TINJAUAN PUSTAKA

Didalam bab ini akan ditampilkan pustaka yang dijadikan tinjauan penulis dalam menyelesaikan karya tulis ini.

BAB IV METODE PENELITIAN

Didalam bab ini akan dijelaskan metode penelitian yang digunakan penulis untuk mendapatkan data dalam pengembangan dan penyelesaian karya tulis ini.

BAB V HASIL DAN PEMBAHASAN

Didalam bab ini akan dibeberkan hal mengenai pembuatan dan penggunaan perangkat lunak yang dikembangkan.

BAB VI PENUTUP

Didalam bab ini akan dijelaskan kesimpulan akhir dan saran yang bisa diberikan penulis saat mengerjakan dan menyelesaikan karya tulis ini.

BAB II

PERANGKAT LUNAK YANG DIKEMBANGKAN

2.1 Fenomena Perangkat Lunak yang Dikembangkan

HTB-Tools, adalah sebuah perangkat lunak yang berjalan pada sistem operasi *Linux*, yang berfungsi untuk melakukan pembagian *bandwidth* jaringan komputer dengan menggunakan metode pembagian dinamis, dimana *bandwidth* jaringan yang dialokasikan ke sebuah komputer klien bisa dipinjamkan kepada komputer lainnya jika komputer klien tersebut tidak sedang digunakan *bandwidth* jaringannya. Dan untuk membuat kebijakan pengalokasian *bandwidth HTB-Tools* bisa berubah pada kurun waktu tertentu, diperlukan sebuah program bantu yang bernama *cron daemon*. *Cron daemon* sendiri merupakan salah satu utilitas yang ada didalam sistem operasi *Linux*, yang fungsinya untuk menjalankan pekerjaan-pekerjaan secara otomatis pada waktu yang telah ditentukan pengguna.

Perangkat lunak yang akan dibuat didalam skripsi ini akan bekerja sebagai sebuah perangkat lunak yang mempermudah dan mempercepat pengguna dalam membuat berkas konfigurasi *HTB-tools* jika dibandingkan dengan membuat berkas tersebut dengan menggunakan *text editor*. Selain membuat berkas konfigurasi *HTB-tools*, perangkat lunak ini juga memiliki fungsi yang digunakan untuk mempermudah pengguna dalam menentukan

kapan pengaplikasian berkas konfigurasi tersebut kedalam jaringan akan dilakukan.

Prosedur penggunaan dan proses berjalannya perangkat lunak ini terdiri dari beberapa tahap. Tahap pertama, pengguna akan menentukan apa isi dari peraturan pembagian *bandwidth* jaringan, yang meliputi banyaknya kelas jaringan komputer, besarnya *bandwidth* yang dialokasikan kepada kelas tersebut, banyaknya komputer atau jaringan komputer yang ada dalam satu kelas jaringan komputer, dan besarnya *bandwidth* yang ditentukan untuk masing-masing komputer atau jaringan komputer. Kemudian tahap kedua, setelah file konfigurasi (berkas peraturan) yang dibuat telah selesai dan disimpan, pengguna harus menentukan waktu tepatnya file konfigurasi yang telah dibuat tersebut akan diimplementasikan pada *network interface* yang juga harus ditentukan oleh pengguna. Langkah yang terakhir, setelah waktu yang ditentukan tercapai, berkas peraturan yang telah ditentukan waktu pengaktifannya, yang tersimpan didalam direktori utama kumpulan berkas konfigurasi, akan dikirimkan kedalam direktori penyimpanan berkas konfigurasi yang digunakan oleh *HTB-Tools* sebagai dasar peraturan pembagian *bandwidth* jaringan, dan diaktifkan.

BAB III

TINJAUAN PUSTAKA

3.1 Teori Pendukung

3.1.1 Jaringan Komputer

Jaringan Komputer adalah kumpulan interkoneksi dari komputer-komputer yang otonom (Sofana, 2008 : 4).

Jaringan komputer dapat dibedakan berdasarkan cakupan geografisnya. Ada tiga katagori utama jaringan komputer yaitu :

a. LAN (*Local Area Network*)

Lokal Area Network adalah jaringan lokal yang dibuat pada area tertutup. Misalkan dalam satu gedung atau dalam satu ruangan. Seringkali jaringan lokal disebut juga jaringan *private*. LAN biasa digunakan untuk jaringan kecil yang menggunakan *resource* bersama-sama, seperti penggunaan *printer* secara bersama, penggunaan media penyimpanan secara bersama. (Sofana, 2008 : 4)

b. MAN (*Metropolitan Area Network*)

Metropolitan Area Network menggunakan metode yang sama dengan LAN namun daerah cakupannya lebih luas Daerah cakupan MAN bisa satu RW, beberapa kantor yang berada dalam komplek yang sama, satu kota, bahkan

satu provinsi. Dapat dikatakan MAN merupakan pengembangan dari LAN (Sofana, 2008 : 4).

c. WAN (*Wide Area Network*)

Wide Area Network cakupannya lebih luas daripada MAN. Cakupan WAN meliputi satu kawasan, satu Negara, satu pulau, bahkan satu benua. Metode yang digunakan WAN hamper sama dengan LAN dan MAN (Sofana, 2008 : 4).

3.1.2 Protokol Jaringan

Protokol pada suatu jaringan merupakan aturan dalam melakukan pengiriman data berupa blok-blok data dari sebuah *node* jaringan ke *node* jaringan yang lain (Mulyanta, 2008 : 5).

3.1.2.1 *Internet Protocol* (IP)

IP *address* yaitu sistem pengalamatan di *network* yang direpresentasikan dengan sederetan angka berupa kombinasi 4 deret bilangan antara 0 sampai dengan 255 yang masing-masing dipisahkan oleh tanda titik (.), mulai dari 0.0.0.1 hingga 255.255.255.255

IP *address* digunakan sebagai alamat dalam hubungan antar *host* di *internet* sehingga merupakan sebuah sistem komunikasi yang *universal* karena merupakan metode pengalamatan yang telah diterima di seluruh dunia. Dengan menentukan IP *address*

berarti kita telah memberikan identitas yang *universal* bagi setiap interadce komputer. Jika suatu komputer memiliki lebih dari satu *interface* (misalkan menggunakan dua ethernet) maka kita harus memberi dua IP *address* untuk komputer tersebut masing-masing untuk setiap *interface*nya. (Sopandi, 2006 : 57)

3.1.2.2 Format Penulisan IP *Address*

IP *address* terdiri dari bilangan biner 32 bit yang dipisahkan oleh tanda titik setiap 8 bitnya. Tiap 8 bit ini disebut sebagai *oktet*. Bentuk IP *address* dapat dituliskan sebagai berikut :

xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx

Jadi IP *address* ini mempunyai *range* dari 00000000.00000000.00000000.00000000 sampai 11111111.11111111.11111111.11111111. Notasi IP *address* dengan bilangan biner seperti ini susah untuk digunakan, sehingga sering ditulis dalam 4 bilangan desimal yang masing-masing dipisahkan oleh 4 buah titik yang lebih dikenal dengan “notasi desimal bertitik”. Setiap bilangan desimal merupakan nilai dari satu oktet IP *address* (Sopandi, 2006 :57).

3.1.2.3 Pembagian Kelas IP *Address* V.4

a. Kelas A

Bit pertama IP *address* kelas A adalah 0, dengan panjang net ID 8 bit dan panjang *host* ID 24 bit. Jadi byte pertama IP *address* kelas A mempunyai *range* dari 0-127. Jadi pada kelas A terdapat 127 *network* dengan tiap *network* dapat menampung sekitar 16 juta *host* (255x255x255) (Sopandi, 2006 : 58).

b. Kelas B

Dua bit IP *address* kelas B selalu diset 10 sehingga byte pertamanya selalu bernilai antara 128-191. Network ID adalah 16 bit pertama dan 16 bit sisanya adalah *host* ID sehingga kalau ada komputer mempunyai IP *address* 167.205.26.161, *network* ID = 167.205 dan *host* ID = 26.161. Pada. IP *address* kelas B ini mempunyai *range* IP dari 128.0.xxx.xxx sampai 191.155.xxx.xxx, yakni berjumlah 65.255 *network* dengan jumlah *host* tiap *network* 255 x 255 *host* atau sekitar 65 ribu *host* (Sopandi, 2006 : 61).

c. Kelas C

IP *address* kelas C mulanya digunakan untuk jaringan berukuran kecil seperti LAN. Tiga bit pertama IP *address* kelas C selalu diset 111. Network ID terdiri dari 24 bit dan *host* ID 8 bit sisanya sehingga dapat

terbentuk sekitar 2 juta *network* dengan masing-masing *network* memiliki 256 *host*. (Sopandi, 2006 : 64)

d. Kelas D

IP *address* kelas D digunakan untuk keperluan multicasting. 4 bit pertama IP *address* kelas D selalu diset 1110 sehingga byte pertamanya berkisar antara 224-247, sedangkan bit-bit berikutnya diatur sesuai keperluan *multicast group* yang menggunakan IP *address* ini. Dalam multicasting tidak dikenal istilah *network* ID dan *host* ID (Sopandi, 2006 : 66).

e. Kelas E

IP *address* kelas E tidak diperuntukkan untuk keperluan umum. 4 bit pertama IP *address* kelas ini diset 1111 sehingga byte pertamanya berkisar antara 248-255 (Sopandi, 2006 : 68).

3.1.3 Komponen Jaringan

3.1.3.1 NIC (*Network Interface Card*)

Network Interface Card (NIC) berfungsi sebagai Interface Fisik atau penghubung antar komputer menggunakan kabel jaringan ke peralatan penghubung. *Card* dipasang pada *slot* tambahan yang terdapat di masing-masing komputer. Setelah *card* terpasang, pasangkan kabel jaringan ke *port* yang terdapat pada

NIC agar komputer yang satu dengan yang lain dapat terhubung secara fisik (Sofana, 2008 : 66).

3.1.3.2 HUB atau *Switch*

HUB atau *Switch* digunakan untuk menghubungkan setiap *node* dalam jaringan LAN. Peralatan ini sering digunakan pada topologi *star* dan *extended star*. Perbedaan antara HUB dan *Switch* adalah kecepatan *transfer* datanya (Sofana, 2008 : 68).

3.1.3.3 *Bridge*

Bridge adalah peralatan jaringan yang digunakan untuk memperluas atau memecah jaringan. *Bridge* berfungsi untuk menghubungkan dan menggabungkan media jaringan yang tidak sama seperti kabel *unshielded twisted pair* (UTP) dan kabel *fiber-optic*, dan untuk menggabungkan arsitektur jaringan yang berbeda seperti *Token Ring* dan *Ethernet* (Sofana, 2008 : 6).

3.1.3.4 *Modem*

Modem adalah singkatan dari *modulator* dan *demulator*. *Modem* berfungsi untuk mengubah sinyal analog (sinyal suara) menjadi sinyal digital. Pada saat

sebuah komputer mengirimkan data ke *internet*, modem akan mengubah sinyal digital dari komputer menjadi sinyal suara, sehingga sinyal tersebut bisa dilewatkan melalui kabel telepon (Sunarto, : 83).

3.1.4 Tipe Jaringan komputer

3.1.4.1 Jaringan *peer to peer*

Pada jaringan *peer to peer* setiap komputer yang terhubung pada jaringan dapat berkomunukasi dengan komputer-komputer yang lain secara langsung tanpa melalui komputer perantara (Sofana, 2008 : 6).

3.1.4.2 Jaringan *Client-Server*

Berbeda dengan jaringan *peer to peer*, pada jaringan *client-server* terdapat Sebuah komputer yang berfungsi sebagai *client*. Sesuai dengan namanya maka komputer *server* berfungsi dan bertugas melayani seluruh komputer yang terdapat dalam jaringan tersebut sedangkan *client* atau *workstation*, yaitu komputer dimana pengguna jaringan dapat mengakses dan memanfaatkan pelayanan yang diberikan oleh komputer *server* (Sofana, 2008 : 6).

3.1.5 *Bandwidth*

Dalam sistem analog, *bandwidth* adalah perbedaan antara frekuensi tertinggi dan terendah yang dapat dihasilkan sebuah alat. Frekuensi diukur dalam satuan putaran per detik atau Hertz (Hz). Dalam teknologi digital seperti komputer dan jaringan komputer, *bandwidth* adalah ukuran transmisi data dalam bits perdetik(bps), ribuan bits perdetik (kbps), atau jutaan bits perdetik(Mbps) (Andrew, 2009 : 864).

3.1.6 *Traffic Control*

Traffic control adalah istilah yang digunakan untuk menjelaskan *packet queueing subsystem* dalam jaringan. *Traffic control* terdiri dari beberapa fungsi, yaitu: klasifikasi, pengaturan, penjadwalan dan *shaping*. Klasifikasi adalah mekanisme yang mengidentifikasikan paket dan menempatkan mereka pada kelas-kelas. Pengaturan adalah mekanisme yang mengatur besar paket data yang mengalir dalam suatu klasifikasi khusus. Penjadwalan adalah proses pengambilan keputusan di mana paket akan *ordered* atau *re-ordered* untuk transmisi. *Shaping* adalah proses yang menentukan kapan pengiriman paket akan ditunda dan dimulai untuk menghasilkan kecepatan aliran data yang mantap (stabil) dan dapat diprediksi. Karakteristik-karakteristik dari *traffic control* system ini dapat dikombinasikan untuk menyediakan *bandwidth* pada aliran data atau membatasi besar *bandwidth* yang tersedia

pada aliran data (contoh: *bandwidth* jaringan). Menyimpan data di cara yang tidak akan mengembang terhadap waktu, dan memberikan grafik berguna dengan mengolah data untuk menyelenggarakan data tertentu. RRD dapat digunakan dalam *shell script* maupun *perl script* pada linux atau melalui *frontend* yang berhubungan dengan jaringan dan membuat tampilan yang *userfriendly* (Andrew, 2009 : 865).

3.1.7 *Cron Daemon*

Cron daemon, *crond*, yang sudah dipaketkan dengan distro-distro *Linux* kebanyakan, berfungsi untuk melakukan pengontrolan dalam penjadwalan pekerjaan yang sering terjadi atau digunakan di dalam sistem. Ketika dimulai pada mode *multi-user*, *crond* mengecek direktori /var/spool/cron/crontabs dan /etc/cron.d dan berkas /etc/crontab untuk pekerjaan yang telah dijadwalkan. Kemudian *crond* dipanggil setiap menit, melakukan pekerjaan yang tercatat harus dilakukan pada menit tersebut, memberi tahu hasilnya kepada pemilik jadwalnya, kemudian menunggu menit berikutnya. (S.Keller, Michael. 1999)

Cron didalam *Linux* dapat digunakan untuk menjadwalkan pekerjaan yang berulang dalam interval yang sama. Kata *crontab* berasal dari *"chron table"*, atau tabel waktu. Dimana pengguna membuat tabel dalam format yang ditentukan, yang berisi perintah dan waktu pengeksekusian perintah tersebut. Perintah yang

dituliskan pada tabel dapat berupa program apapun yang bisa dijalankan, misalnya sebuah perintah didalam */usr/bin* atau sebuah program *shell*. Pengguna dapat menggunakan sebuah perintah untuk membuat, mengubah, atau mendaftarkan tabel, dan sistem *cron daemon (crond)* membaca tabel dan mengeksekusi perintah perintah tersebut pada waktu yang telah ditentukan. (Raithel, John. 1996)

Untuk membuat *crontab*, seorang pengguna bisa menggunakan perintah *crontab –e*. Perintah ini akan membuat sebuah berkas *crontab* yang akan dibaca *crond* untuk mencari tugas yang harus diselesaikan. Perintah *crontab* kemudian akan mencari variabel *visual environment*, kemudian mencari variabel *editor environment*. Dan kemudian menggunakan editor teks yang terdapat pada variabel-variabel tersebut untuk digunakan sebagai alat bantu dalam mengubah berkas *crontab*. (S. Keller, Michael. 1999)

Didalam berkas crontab, setiap masukan terdiri atas enam bagian yang terpisah satu sama lain oleh spasi. Dimulai dari masukan menit penjalanan perintah dari angka 0 s/d 59 sebagai bagian pertama, jam dari angka 0 s/d 23 sebagai bagian kedua, hari dalam satu bulan dari angka 1 s/d 31 sebagai bagian ketiga, bulan dalam satu tahun dari angka 1 s/d 12 sebagai bagian keempat, hari dalam satu minggu dari angka 0 s/d 6 dimana angka 0 dianggap

sebagai hari minggu sebagai bagian kelima, dan perintah yang akan dijalankan pada bagian keenam.

Saat pertama, mungkin format ini terlihat adanya konflik dikarenakan terdapat dua bagian yang berisikan variabel hari, hari dalam satu bulan dan hari dalam satu minggu, tetapi hal ini sebenarnya agar dapat melakukan algoritma penjadwalan yang berbeda. Contohnya, saat pengguna ingin perintahnya dijalankan setiap hari selasa atau setiap tanggal 15 setiap bulannya. Berikan nilai asterik (*) pada bagian hari yang tidak digunakan. Kedua bagian hari ini bisa digunakan secara bersamaan misalnya setiap tanggal 15 yang merupakan hari selasa setiap bulannya.

Masukan yang berupa jarak didalam *crontab* ditentukan dengan menggunakan tanda pisah (-). Jika ingin menentukan masukan dimulai dari tanggal 8 s/d 15 dalam satu bulan, masukkan 8-15 pada bagian ke-3. Masukkan yang tidak berturut-turut dalam salah satu bagian dipisahkan oleh tanda koma (,) , jadi 8,15 didalam bagian ke-3 berarti hanya tanggal 8 dan tanggal 15. Untuk menentukan masukan yang berarti tidak ada penghentian atau penjalanan perintah yang terus menerus, bisa menggunakan tanda asterik (*), misal untuk menentukan “setiap hari” berikan masukan asterik (*) pada bagian ke-3. Berikut contoh masukan didalam *crontab*:

```
12  4  *     *  *  /usr/local/bin/backup
```

```
5   3  10-15 4  *  echo "taxes due" | mail jones
```

(Sumber : Diolah Sendiri)

Gambar 3.1 Contoh *Entry crontab*

Baris pertama berfungsi untuk menjalankan script backup setiap pagi pada jam 4:12, dan baris kedua berfungsi untuk mengirimkan pesan kepengguna “jones” selama enam hari sebagai pengingat bahwa pajak harus dibayar. Pada umumnya, pengeksekusian perintah oleh *crontab* sebaiknya dilakukan diluar dari waktu sibuk untuk mengurangi penggunaan sistem pada jam sibuk. Jika standar keluaran dan standar error nya tidak ditentukan, hasilnya akan diberitahukan kepada pemilik dari berkas *crontab* tersebut. Pada contoh diatas, jika pengguna “jones” tidak ditemukan, pemilik berkas *crontab* inilah yang akan diberikan keluaran berupa “taxes due” dan pesan errornya.

Setelah berkas *crontab* telah disimpan dan pengguna sudah keluar dari editor, maka akan dibuat sebuah berkas *crontab* untuk pengguna didalam direktori. Untuk melihat berkas-berkas *crontab* yang ada, pengguna bisa menggunakan perintah *crontab –l* dan untuk menghapus berkas *crontab –d*. *Superuser* bisa menggunakan perintah *crontab –d username* untuk menghapus berkas *crontab* yang dimiliki oleh pengguna (*username* sebagai contoh). (Raithel, John. 1996)

3.1.8 *Hierarchical Token Bucket*

Penjadwalan paket *Hierarchical Token Bucket* (HTB), adalah mekanisme yang menyediakan kemampuan pengaturan pembagian *bandwidth* dan berguna untuk melakukan pengaturan aktifitas pengiriman paket data didalam jalur jaringan komputer (*traffic*).

Fungsi pengontrolan *traffic*, seperti yang diimplementasikan oleh *Alexey Kuznetsov*, terdiri dari empat komponen: *queuing discipline*, *quality of service*, *filtering*, dan *policing*.

Queuing discipline merupakan mekanisme-mekanisme perangkat lunak yang menentukan algoritma yang digunakan untuk menghadapi antrian paket-paket IP. Setiap alat jaringan dikaitkan dengan sebuah *queuing discipline*, dan sebuah *queuing discipline* khusus menggunakan algoritma FIFO untuk mengontrol antrian paket. Paket-paket disimpan dalam urutan saat diterima dan diteruskan secepat perangkat yang terkait dengan antrian tersebut dapat mengirimkannya. HTB menggunakan algoritma *Token Bucket Filter* (TBF) dalam mengontrol antrian paket untuk setiap kelas yang didefinisikan didalam *class of service* yang terkait dengannya.

Sebuah *class of service* mendefinisikan kebijakan pengaturan, seperti *bandwidth* maksimal atau *burst* maksimal, dan menggunakan *queuing discipline* untuk menjalankan peraturan-

peraturan tersebut. Sebuah *queuing discipline* dan kelas (*class of service*) terikat bersama. Peraturan yang didefinisikan oleh sebuah kelas harus terhubung dengan sebuah antrian yang telah didefinisikan. Dalam kebanyakan kasus, setiap kelas memiliki sebuah *queue discipline*, tetapi juga terdapat kemungkinan beberapa kelas saling berbagi antrian (*queue discipline*) yang sama.

Filters menentukan peraturan-peraturan yang digunakan oleh *queuing discipline*. Kemudian *queuing discipline* menggunakan peraturan-peraturan tersebut untuk memutuskan ke kelas mana paket-paket akan diteruskan. Setiap *filter* memiliki prioritas yang ditetapkan. *Filter* diurutkan berdasarkan prioritas yang mereka miliki. Ketika sebuah *queuing discipline* mendapatkan sebuah paket yang akan diteruskan, ia akan mencoba mencocokan paket tersebut pada salah satu *filter* yang sudah ditentukan. Pencarian kecocokan diselesaikan menggunakan setiap *filter* yang ada didalam daftar, dimulai dari *filter* yang memiliki prioritas yang paling tinggi. Setiap *class of service* atau *queuing discipline* bisa memiliki satu *filter* yang dihubungkan dengannya atau lebih.

Komponen *policing* berfungsi untuk memastikan tidak ada jalur paket yang melebihi kapasitas *bandwidth* yang telah ditentukan. Keputusan *policing* dibuat berdasarkan *filter* dan peraturan yang ditetapkan perkelas. (Benita, Yaron. 1999)

3.1.8.1 *HTB-TOOLS*

Perangkat lunak manajemen *bandwidth HTB-tools* adalah sebuah perangkat lunak dengan beberapa *tools* yang mempermudah proses pengalokasian *bandwidth* yang sulit, untuk jalur *upload* dan *download*, menggunakan fasilitas kernel *HTB* di *Linux*, yang diajukan oleh *Spirlea*, *Subredu*, dan *Stanimir*.

Perangkat lunak ini bisa membuat dan mengecek berkas konfigurasi dan juga menyediakan pengawasan jalur paket secara *real-time* untuk setiap klien.

Fitur-fitur yang dimiliki oleh *HTB-tools* meliputi :

1) Pembatasan *bandwidth* pada jaringan publik, menggunakan dua berkas konfigurasi untuk jalur *upload* dan *download*.
2) Pembatasan *bandwidth* pada jaringan lokal, menggunakan satu berkas konfigurasi.
3) *Match mark*
4) *Match mark* pada u32
5) Pembatasan pada jaringan eksternal

HTB-tools sendiri terdiri dari :

1) *q_parser*, yang berfungsi untuk membaca sebuah berkas konfigurasi dan membuat sebuah *setting*-an HTB.

2) *q_checkcfg*, yang berfungsi untuk mengecek berkas konfigurasi.

3) *q_show*, untuk menampilkan status jalur paket data klien yang ditentukan dalam berkas konfigurasi secara *real-time* pada terminal.

4) *q_show.php*, untuk menampilkan status jalur paket data klien yang ditentukan dalam berkas konfigurasi melalui halaman *web*.

5) *wHTB-tools_cfg_gen*, untuk membuat berkas konfigurasi dari sebuah halaman *web* (hanya di *HTB-tools 0.3.0*).

6) *htbgen*, membuat berkas konfigurasi dari *bash shell*.

3.2. Hasil Penelitian Terdahulu

3.2.1 Konfigurasi *HTB-TOOLS* v0.3.0

Berdasarkan hasil pada Laporan Praktek Kerja Lapangan yang berjudul “Implementasi pembagian *bandwidth* koneksi jaringan internet menggunakan algoritma *Hierarchical Token Bucket* (HTB) pada CV. AN Mandiri” yang ditulis oleh peneliti sebelumnya, yang membahas tentang pengimplementasian HTB-

Tools-0.3.0, konfigurasi khusus untuk *HTB-tools* terdiri dari 3 tahap, yakni tahap instalasi, pembuatan berkas konfigurasi, dan pengeksekusian *HTB-tools*.

Tahap instalasi terdiri dari pemindahan berkas-berkas bagian dari aplikasi *HTB-tools*. Berkas yang dipindahkan berupa *htb, htbgen, q_checkcfg, q_parser,* dan *q_show* kedalam direktori */sbin,* direktori yang bernama *etc* dipindahkan kedalam direktori */etc,* dan berkas *rc.htb.new* diubah namanya menjadi *rc.htb* dan dipindahkan kedalam direktori */etc/init.d*.

Hal berikutnya yang harus dilakukan adalah membuat berkas konfigurasi yang akan digunakan HTB-tools sebagai dasar pengaturan manajemen *bandwidth* jaringan. Berkas yang dibuat, namanya disesuaikan dengan jaringan komputer yang terhubung dengan *network interface* yang akan diatur, misalnya untuk mengatur jaringan komputer yang terhubung dengan *eth1* dari sistem, berkas diberikan nama *eth1-qos.cfg,* dan untuk *eth2* diberikan nama *eth2-qos.cfg*.

Untuk format isi dari berkas konfigurasi, terdiri dari dua bagian utama, yaitu bagian kelas klien dan bagian klien. Informasi yang dibutuhkan untuk kelas klien terdiri dari nama kelas yang tidak boleh ditulis menggunakan simbol dan tidak diawali dengan angka, nilai *bandwidth* minimum dengan satuan kbits, nilai *bandwidth* maksimum dengan satuan kbits, nilai *burst*, tingkat prioritas pengiriman data untuk kelas ini (0 s/d 7), dan klien-klien

yang berada didalam kelas klien. Dan untuk informasi yang perlu dimasukkan untuk klien berupa nama klien yang tidak boleh menggunakan simbol dan tidak diawali dengan angka, nilai *bandwidth* minimum dengan satuan kbits, nilai *bandwidth* maksimum dengan satuan kbits, besarnya nilai *burst*, tingkatan prioritas (0 s/d 7), *IP address* yang digunakan oleh klien, dan *port* yang dibatasi. Untuk contoh format penulisannya sebagai berikut

```
class class_1 {
      bandwidth 192;
      limit 256;
      burst 2;
      priority 1;

      client client_1 {
            bandwidth 128;
            limit 192;
            burst 2;
            priority 1;
            dst {
                  192.168.1.0/24;
            };
      };
      client client_2 {
            bandwidth 64;
            limit 64;
            burst 2;
            priority 1;
            dst {
                  192.168.2.2/32 80;
            };
      };
};
```

(Sumber : Diolah Sendiri)

Gambar 3.2 Contoh Berkas Konfigurasi *HTB-Tools*

Pada berkas konfigurasi tersebut, *class_1* diberikan jaminan *bandwidth* sebesar 192 kbits, dan bisa meminjam *bandwidth* yang tidak digunakan oleh kelas lain semaksimalnya sampai *bandwidth class_1* mencapai 256 kbits. Nilai *burst* menentukan besarny data

yang bisa dikirimkan sekaligus dalam satu pengiriman, bila diberikan nilai 0, *HTB-tools* akan menghitung sendiri nilai besar *burst*. Nilai prioritas (*priority*) menentukan tingkatan prioritas transmisi data satu kelas atau klien jika jalur transmisi data sudah mulai padat. Untuk variabel-variabel klien, dimulai dari variabel *bandwidth* sampai variabel *priority*, fungsinya sama dengan variabel yang dimiliki oleh kelas klien, variabel yang memiliki fungsi berbeda terdapat pada *dst*. Variabel *dst* memiliki fungsi untuk menerima nilai berupa *IP address* sebuah klien beserta nilai *port* yang akan dibatasi. Bila kliennya berupa jaringan komputer, untuk penulisanny menggunakan h*ost address* disertai *netmask* yang dipisahkan dengan *slash* (/), dan bila kliennya berupa komputer individu, dituliskan *ip address* disertai nilai 32 sebagai petunjuk bahwa klien merupakan komputer individu, dan bukan jaringan komputer, yang dipisahkan dengan *slash* (/). Setelah penentuan *IP* klien, *port* yang ingin dibatasi transmisi datanya bisa dilakukan dengan cara memberikan nilai *port* tersebut setelah *IP* klien dan sebelum tanda titik koma, dibatasi dengan spasi.

Pengeksekusian *HTB-tools* dilakukan dengan cara menjalankan berkas *rc.htb* yang terletak pada direktori */etc/init.d* dengan diberikan parameter *start_eth1* untuk mejalankannya di *interface eth1*. Berikut contoh perintahnya.

```
root@router:/home#/etc/init.d/rc.htb start_eth1
```

Ada juga perintah lainnya, antara lain *stop_eth* yang berfungsi untuk menghentikan *HTB-tools* pada *interface* tertentu, *show_eth* yang berfungsi untuk melihat aktifitas transmisi data pada *interface* tertentu, *stop* yang berfungsi untuk menghentikan *HTB-tools* pada semua *interface* jaringan, dan *start* untuk mengaktifkan *HTB-tools* pada semua *interface* jaringan.

BAB IV

METODE PENELITIAN

4.1 Waktu Penelitian

Peneliti memulai riset pada tanggal 24 Desember 2011, yang kemudian dilanjutkan dengan desain dan pengembangan aplikasi pada bulan Februari tahun 2012. Penulisan skripsi ini dimulai pada tanggal 10 April 2012 dan dijadwalkan akan selesai pada akhir bulan Juli tahun 2012.

4.2 Jenis Data yang Digunakan

Adapun jenis data yang digunakan dalam penyusunan skripsi ini, yaitu :

1) Data Primer

Menurut Kuncoro (2003:157), data primer dapat didefinisikan sebagai data yang dikumpulkan dari sumber-sumber asli untuk tujuan tertentu.

Peneliti dalam melakukan obeservasi terhadap kinerja *HTB-tools* ketika isi berkas-berkas konfigurasinya diubah. Selain itu juga dilakukan observasi terhadap kinerja *crond*. Kedua data ini

digunakan peneliti untuk memahami prilaku kinerja *HTB-tools* dan *crond*

2) Data Sekunder

Menurut Kuncoro (2003:148), data sekunder adalah data yang telah dikumpulkan oleh pihak lain.

Dalam penulisan skripsi ini, penulis mendapatkan data sekunder berupa bentuk dan struktur penulisan berkas konfigurasi yang akan digunakan *HTB-tools* dan tabel *cron*. Dimana kedua data ini akan digunakan sebagai dasar peneliti dalam membangun fungsi untuk mempermudah pengguna dalam pengaturan *HTB-Tools* dan *crond*.

4.3 Teknik Pengumpulan Data

Dalam proses penyusunan skripsi ini, teknik pengumpulan data yang digunakan, yaitu :

1) Tinjauan Pustaka

Tinjauan pustaka meliputi identifikasi, lokasi, dan analisis dari dokumen yang berisi informasi yang berhubungan dengan permasalahan penelitian secara sistematis (Kuncoro, 2003:34).

Studi pustaka yang dilakukan oleh penulis meliputi membaca artikel yang bersangkutan dengan masalah yang diteliti,

membaca hal mengenai baris perintah yang bisa digunakan oleh penulis dalam mengembangkan aplikasi, dan melakukan studi mengenai algoritma pemrograman yang bisa digunakan.

2) Observasi

Menurut Sutarbi (2004:133), observasi adalah pengamatan langsung objek yang akan dibahas yang akan dilakukan penulis atau penulis untuk mengumpulkan data.

Observasi yang dilakukan penulis meliputi melakukan pengamatan terhadap bagaimana perubahan yang dilakukan *HTB-tools* dan *crond* ketika berkas yang digunakan masing-masing aplikasi diubah.

4.4 Jenis Penelitian

Menurut Hermawan (2005:17), penelitian eksploratif adalah penelitian yang dilakukan apabila penelitian sebelumnya masih jarang. Tujuannya adalah untuk melihat pola, gagasan, atau merumuskan hipotesis bukan untuk menguji hipotesis.

Penulis menggolongkan penelitiannya sebagai tugas akhir kedalam golongan jenis penelitian eksploratif, dikarenakan penelitian ini bertujuan untuk membuat sesuatu yang baru dan untuk melihat pola kerja serta hasil dari program atau aplikasi yang dilaporkan didalam skripsi ini.

4.5 Alat dan Teknik Pengembangan Sistem

4.5.1 Alat Pengembangan Sistem

Didalam membangun sistem aplikasi yang dilaporkan didalam laporan ini, peneliti menggunakan :

1) *Hardware*

   Processor : *Intel Core 2 Duo* (2.40GHz)

   Memory : 3072MB

2) *Software*

   Sistem Operasi : *Linux Ubuntu*, versi *kernel 2.6.32-39-generic*

   Program : *Linux Shell* dan *Dialog v1.1*

4.5.2 Model Data

Berikut adalah data berkas konfigurasi *HTB-tools* dan berkas *crontab* yang akan dibuat, disimpan, dan digunakan dalam pengembangan aplikasi :

1) Berkas */etc/crontab* yang akan digunakan sebagai basis penjadwalan pengaktifan berkas konfigurasi *HTB-tools*.

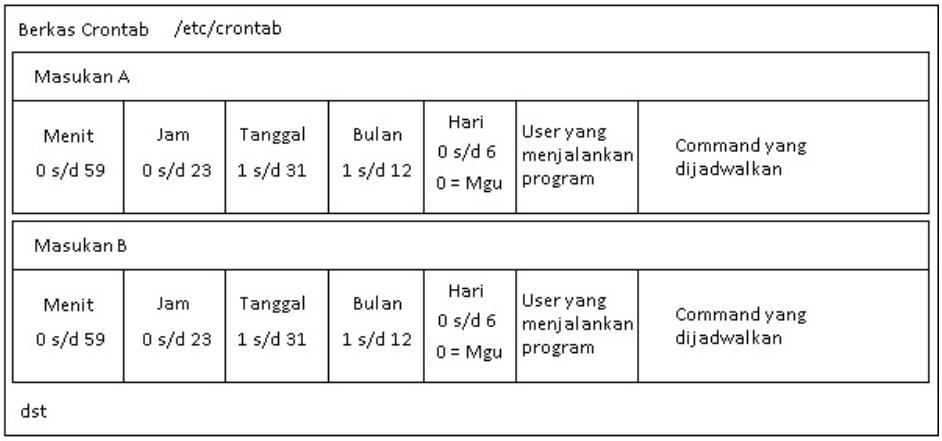
```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user    command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

(Sumber : Diolah Sendiri)

Gambar 4.1 Berkas */etc/crontab*

Dari gambar berkas diatas, bentuk struktur penulisannya jika digambarkan maka bisa dilihat sebagai berikut.



(Sumber : Diolah Sendiri)

Gambar 4.2 Struktur Penulisan pada */etc/crontab*

2) Berkas konfigurasi HTB-tools yang akan digunakan oleh HTB-tools sebagai dasar pembagian *bandwidth* jaringan komputer. Gambar 4.3 adalah contoh berkas *HTB-tools*, dimana pembagian akan dilakukan kepada jaringan yang terhubung dengan *interface eth1*.

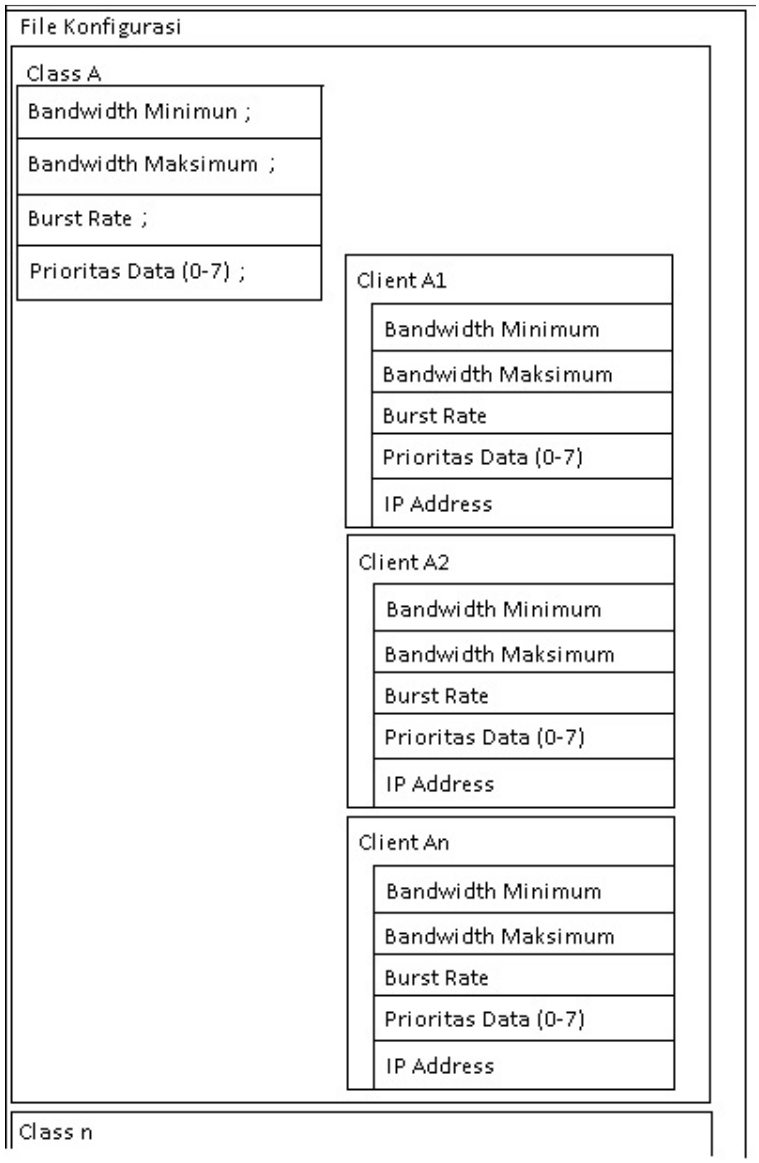
```
################
# eth1-qos.cfg #
################
# for how to configure and use see docs/HowTo/

class class_1 {
      bandwidth 192;
      limit 256;
      burst 2;
      priority 1;
               client client_1 {
                  bandwidth 48;
                  limit 64;
                  burst 2;
                  priority 1;
                     dst {
                        192.168.1.0/24; };
                 };
               client client_2 {
                  bandwidth 48;
                  limit 64;
                  burst 2;
                  priority 1;
                     dst {
                        192.168.2.0/24; };
                 };
               client client_3 {
                  bandwidth 48;
                  limit 64;
                  burst 2;
                  priority 1;
                     dst {
                        192.168.3.0/24; };
                 };
               client client_4 {
                  bandwidth 48;
                  limit 64;
                  burst 2;
                  priority 1;
                     dst {
                        192.168.4.0/24; };
                 };
      };
class default { bandwidth 8; };
```

(Sumber : Diolah Sendiri)

Gambar 4.3 Berkas Konfigurasi *HTB-tools*

Dari berkas diatas, bisa digambarkan struktur penulisan dari berkas konfigurasi *HTB-tools* sebagai berikut.



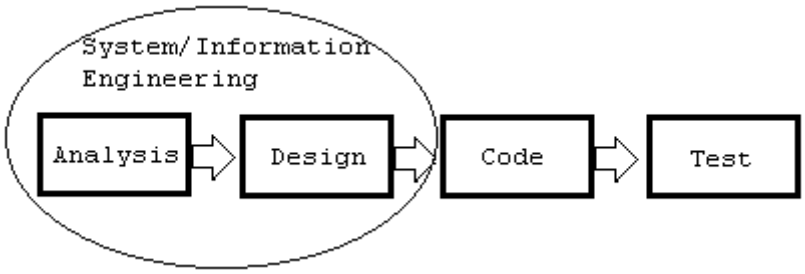
(Sumber : Diolah sendiri)

Gambar 4.4 Struktur Penulisan Berkas Konfigurasi *HTB-tools*

4.5.3 Teknik Pengembangan Sistem

Peneliti, dalam mengembangkan aplikasi yang dilaporkan pada skripsi ini menggunakan teknik SDLC atau *Waterfall*. Dimana pembuatan aplikasi terbagi kedalam empat bagian dan harus

dijalani secara berurut, dimulai dari tahap analisis, desain, pengkodean, dan berakhir pada penerapan.



(Sumber : Fatta, 2007)

Gambar 4.5 Model SDLC menurut Pressman

1) Pada tahap analisis, peneliti mengumpulkan informasi yang dibutuhkan untuk membangun sistem aplikasi, dengan menggunakan teknik pengumpulan data observasi dan studi pustaka.
2) Pada tahap desain, peneliti menggunakan data dan informasi yang telah dikumpulkan sebagai dasar dalam perancangan sistem aplikasi yang akan dibuat.
3) Pada tahap pengkodean dan pengetesan, peneliti menulis kode masing-masing fungsi sesuai dengan bentuk desain yang telah ditentukan sebelumnya dan melakukan pengetesan untuk melihat apakah kode yang ditulis bisa dijalankan sesuai dengan keinginan atau tidak.

4) Pada tahap penerapan, peneliti menerapkan seluruh sistem aplikasi kedalam komputer lain sebagai simulasi untuk melihat kinerja sistem aplikasi.

BAB V

HASIL DAN PEMBAHASAN

5.1 Tahap Analisis

Tahap analisis dibagi kedalam dua bagian, yakni:

1) Analisis kelemahan sistem sebelumnya
2) Analisis kebutuhan sistem yang baru

5.1.1 Analisis Kelemahan Sistem Lama

Berdasarkan pada observasi yang sebelumnya dilakukan pada penggunaan *HTB-tools*, terdapat beberapa kelemahan yang ditemukan oleh penulis, yakni :

1) Pembuatan berkas konfigurasi HTB-tools harus dibuat secara manual menggunakan *text editor*, sehingga menyebabkan lambatnya dalam pembuatan berkas konfigurasi, dan tingkat kesalahan dalam pembuatan berkas tersebut oleh pengguna sangat tinggi, dikarenakan berkas konfigurasi tersebut dibuat secara manual.
2) Berkas yang dibuat dan disimpan kedalam direktori penyimpanan berkas konfigurasi utama *HTB-tools* harus disesuaikan dengan nama *interface* jaringan yang ingin

diimplementasikan, sehingga membuat pengguna sulit untuk mengingat berkas-berkas konfigurasi yang akan dan telah dipakai.

3) Untuk menjalankan servis *HTB-tools*, pengguna terlebih dahulu harus mengetikkan perintahnya didalam terminal *Linux*. Hal ini mengakibatkan tingkat fleksibilitas yang rendah, dimana servis tidak bisa dijalankan pada saat pengguna tidak mengontrol komputer secara langsung.

5.1.2 Analisis Kebutuhan Sistem yang Baru

Kebutuhan sistem yang diperlukan dibagi kedalam 2 macam, yakni : kebutuhan fungsional dan kebutuhan nonfungsional.

5.1.2.1 Kebutuhan Fungsional

1) Sistem memiliki fungsi yang bertugas sebagai menu, dimana pengguna dengan hanya memilih salah satu pilihan didalam menu, bisa menjalankan fungsi lainnya.

2) Sistem memiliki fungsi yang mempermudah pengguna dalam pembuatan berkas konfigurasi *HTB-tools*:

   a. Pengguna bisa membuat berkas konfigurasi yang baru dengan menggunakan fungsi ini.

   b. Pengguna hanya perlu memasukkan data mengenai nama berkas, *maxdefault bandwidth*, banyaknya

kelas didalam berkas, nama kelas, besarny *bandwidth* terkecil dan terbesar tiap kelas, tingkat prioritas data kelas, banyaknya klien yang dimiliki tiap kelas, identitas klien, besarnya *bandwidth* terkecil dan terbesar tiap klien, tingkat prioritas data tiap klien, dan *IP address* dan *netmask* tiap klien.

c. Fungsi harus bisa melakukan konversi penulisan *IP address* dan *netmask*-nya kedalam bentuk penulisan *CIDR* pada saat menyimpannya kedalam berkas konfigurasi.

d. Fungsi harus memiliki kemampuan untuk menyimpan berkas yang belum diselesaikan oleh penggunanya, dan bisa dilanjutkan kembali.

3) Sistem memiliki fungsi yang dapat membantu penggunanya dalam memasukkan jadwal yang diinginkan oleh pengguna kedalam tabel *cron* (*crontab*). Dimana pengguna hanya perlu menginputkan berkas konfigurasi yang ingin digunakan, interface jaringan yang ingin diimplementasikan peraturan, dan kapan peraturan akan diaktifkan.

4) Sistem memiliki fungsi yang dapat melakukan pengaktifan berkas konfigurasi yang ditentukan pada

jadwal yang diinginkan telah tiba, dimana kegiatan yang dilakukan oleh fungsi:

a. Menduplikasikan berkas konfigurasi yang dipilih kedalam direktori penyimpanan berkas konfigurasi utama *HTB-tools*. Dilakukan duplikasi dan bukan memindahkan agar berkas konfigurasi tersebut masih dapat digunakan lagi kedepannya.
b. Berkas konfigurasi yang dipilih, diduplikasikan kedalam direktori utama tersebut dengan nama sesuai *interface* jaringan yang ingin diimplementasikan peraturan baru. Hal ini dikarenakan ketentuan dari HTB-tools yang mengharuskan berkas yang digunakan harus dinamakan sesuai dengan nama *interface* jaringan yang ingin digunakan.
c. Setiap kegiatan aktifasi berkas harus dicatatkan kedalam *logfile*, agar pengguna dapat mengetahui aktifitas apa saja yang telah dilakukan oleh sistem.

5) Sistem memiliki fungsi yang dapat membantu penggunanya dalam melakukan instalasi perangkat lunak ini.

5.1.2.2 Kebutuhan Nonfungsional

1) Operasional

Sistem operasi yang digunakan berupa *Linux Ubuntu kernel v2.6* dengan infrastruktur untuk prosesor *Intel*.

2) Keamanan

Sistem hanya bisa dijalankan dengan identitas *root*. Jika menggunakan identitas lainnya, sistem harus dihentikan.

3) Informasi

a. Sistem harus bisa menginformasikan kepada pengguna bila terdapat *input* dari pengguna yang tidak *valid*.
b. Sistem harus bisa menginformasikan kegiatan penukaran berkas konfigurasi yang telah dilakukan.

5.2 Desain dan Rancangan Fungsi dalam Program

Proses jalannya sistem ini secara garis besar bisa diurutkan seperti berikut :

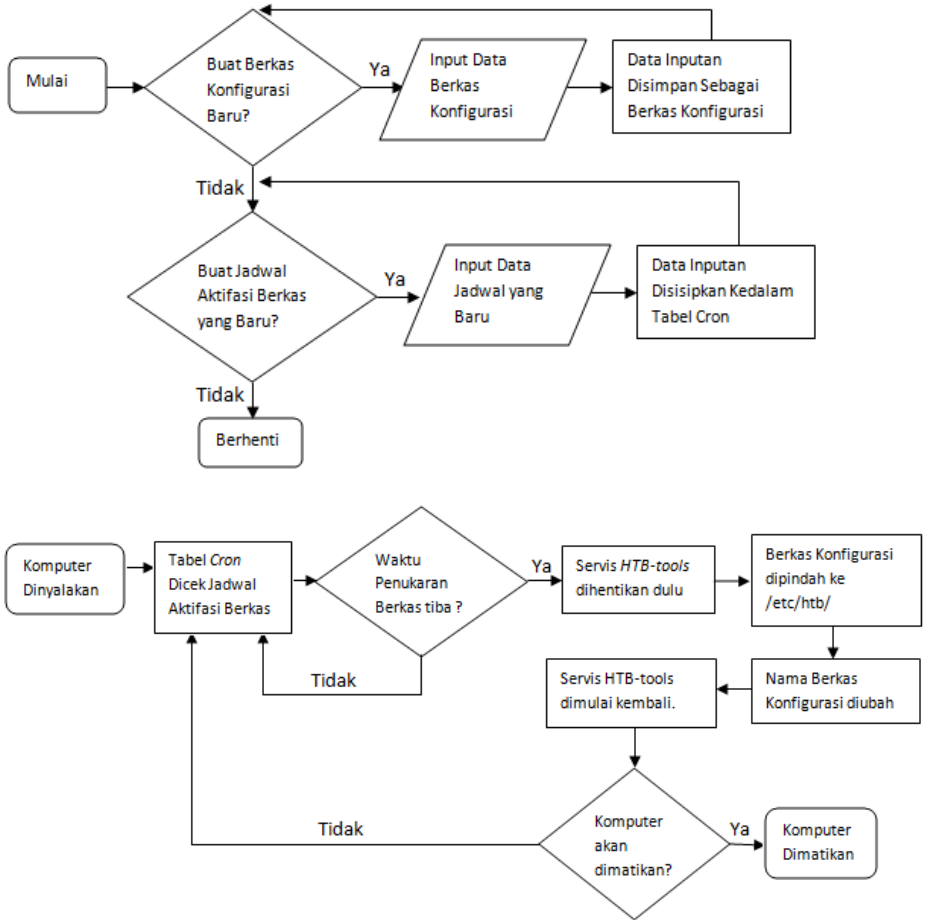
1) Pengguna membuat berkas konfigurasi *HTB-tools* yang baru, jika ingin.
2) Berkas konfigurasi tersebut disimpan.
3) Pengguna menetapkan jadwal pengaktifan berkas konfigurasi.
4) Sistem menunggu sampai waktu yang ditentukan tiba.

5) Saat jadwal pengaktifan berkas baru tiba, sesuai dengan perintah yang dimasukkan kedalam tabel *cron*, servis *HTB-tools* dihentikan sementara.

6) Berkas konfigurasi yang baru dipindahkan kedalam direktori penyimpanan berkas utama.

7) Servis *HTB-tools* diaktifkan kembali dengan peraturan yang baru.

Jika langkah-langkah tersebut digambarkan menjadi diagram alir, bisa dilihat pada Gambar 5.21.



Gambar 5.21 Alur Kerja Sistem Secara Garis Besar

Pada Gambar 5.21, terdapat dua diagram alir, hal ini dikarenakan proses pembuatan berkas konfigurasi dan penetapan jadwal tidak berhubungan dengan proses penukaran berkas konfigurasi, dimana satu proses dimulai pada saat sistem perangkat lunak dijalankan, dan yang satu lagi dimulai pada saat komputer dinyalakan. Proses penukaran berkas konfigurasi tidak berhubungan langsung dengan pengguna.

Setelah dikaji secara garis besar, untuk lebih mendetilnya, jika disesuaikan dengan fungsi-fungsi yang butuh untuk dibuat seperti pada tahap analisis kebutuhan sistem yang baru, dimana sistem harus memiliki beberapa fungsi, maka fungsi-fungsi yang harus ada, dibagi menjadi:

1) Fungsi yang bertugas sebagai menu utama.
2) Fungsi pembuatan berkas konfigurasi *HTB-tools*.
3) Fungsi penetapan jadwal penukaran berkas konfigurasi *HTB-tools*.
4) Fungsi penukaran berkas konfigurasi *HTB-tools*.
5) Fungsi instalasi sistem.

Untuk fungsi yang berperan membantu dalam pembuatan berkas konfigurasi *HTB-tools*, terbagi menjadi 4 bagian, yakni : fungsi *createrule, continuecr8, ubahipadd* dan *editrule*. Fungsi *createrule* berguna untuk bantuan pembuatan berkas konfigurasi dari awal berkas tersebut tidak ada sampai dengan selesai (atau tidak selesai). Fungsi *continuecr8* digunakan untuk melanjutkan pembuatan berkas konfigurasi yang sebelumnya tidak selesai. Fungsi *ubahipadd*, yang dipanggil pada saat fungsi *createrule* dan *continuecr8* membutuhkan pengubahan bentuk penulisan data inputan dari

pengguna berupa *IP address* dan *netmask* kedalam *format* penulisan *CIDR*. Dan fungsi *editrule* yang merupakan fungsi tambahan, berguna untuk membantu pengguna menemukan dimana letak berkas-berkas konfigurasi yang dibuat disimpan, dan mengubah isi dari berkas konfigurasi yang dipilih.

Fungsi penetapan jadwal, dibagi menjadi 2 bagian, yakni : fungsi *settime* dan *edittimeset*. Fungsi *settime* digunakan untuk mendaftarkan jadwal penukaran berkas konfigurasi pada *network interface* yang ditentukan oleh pengguna kedalam tabel *cron*. Dan fungsi edittimeset yang merupakan fungsi tambahan, berguna untuk mengakses tabel *cron* yang merupakan isi dari berkas */etc/crontab*.

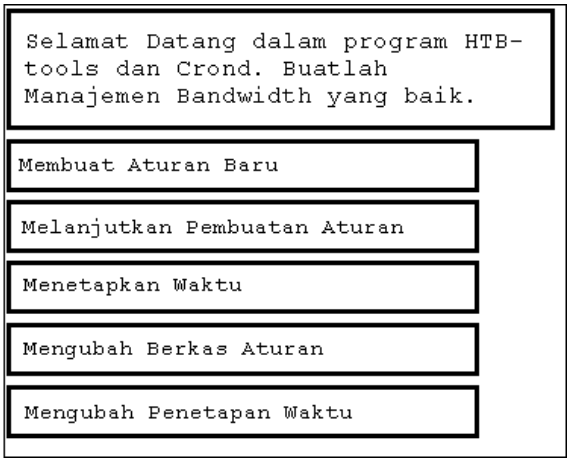
Agar pengguna dalam melakukan *input* data tidak melakukan kesalahan, maka pada setiap fungsi yang berhubungan langsung dengan inputan dari pengguna akan diintegrasikan prosedur validasi data didalamnya. Fungsi-fungsi tersebut antara lain : *createrule, continuecr8, editrule,* dan *settime*.

5.2.1 Fungsi *menu*

Fungsi *menu*, digunakan sebagai indeks utama dari semua fungsi yang dibuat, untuk memberikan kemudahan bagi pengguna dalam menjalankan fungsi-fungsi yang lainnya. Pada saat dijalankan, fungsi ini memberikan 4 pilihan, yakni : "Membuat Aturan", "Menetapkan Waktu", "Mengubah Aturan", dan

"Mengubah Penetapan Waktu". Dan jika terdapat berkas konfigurasi yang sebelumnya belum selesai dibuat, menu akan menampilkan pilihan baru dibawah pilihan "Membuat Aturan", dengan nama pilihan "Lanjutkan Pembuatan Berkas (nama berkas)".

Alur kerja fungsi ini, pada saat dijalankan, identitas pengguna diperiksa apakah menggunakan *root* atau tidak, jika bukan *root*, maka aplikasi akan dihentikan, dan jika menggunakan *root*, menu utama akan ditampilkan.



Gambar 5.22 Desain Tampilan Fungsi *menu*

Kemudian, berdasarkan pilihan dari pengguna, fungsi yang dipilih akan dijalankan.

1) Pilihan "Membuat Aturan" akan menjalankan fungsi *createrule*.
2) Pilihan "Lanjutkan Pembuatan Berkas ..." akan menjalankan fungsi *continuecr8*.

3) Pilihan "Menetapkan Waktu" akan menjalankan fungsi *settime*.
4) Pilihan "Mengubah Aturan" akan menjalankan fungsi *editrule*.
5) Pilihan "Mengubah Penetapan Waktu" akan menjalankan fungsi *edittimeset*.

Setelah fungsi yang dipilih oleh pengguna selesai dijalankan, fungsi *menu* akan dipanggil kembali.

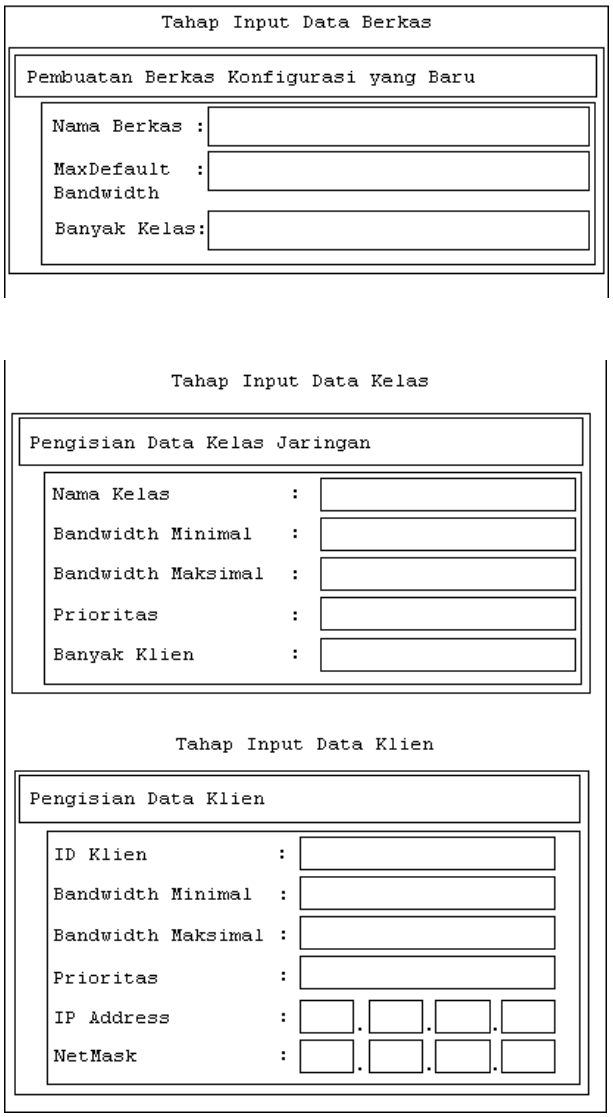
5.2.2 Fungsi *createrule*

Fungsi yang bekerja sebagai antarmuka dengan pengguna untuk melakukan pembuatan berkas konfigurasi ini, memiliki 3 tahapan *input*, yakni :

1) Tahap *input* data berkas, dimana data yang diinputkan adalah nama berkas konfigurasi, besar maksimal *default bandwidth*, dan banyaknya kelas yang ingin dibuat.
2) Tahap *input* data kelas, dimana data yang diinputkan adalah nama kelas, besar alokasi *bandwidth* minimum, besar alokasi *bandwidth* maksimum, tingkat prioritas data kelas, dan banyaknya klien didalam kelas ini.
3) Tahap *input* data klien, dimana data yang diinptkan adalah nama atau id klien, besar alokasi *bandwidth* minimum, besar

alokasi *bandwidth* maksimum, tingkat prioritas data klien, *IP address* klien, dan *subnetmask* klien.

Pada saat fungsi *createrule* dimulai, tahap *input* data berkas dilakukan, kemudian dilanjutkan pada *input* data kelas, dan dilanjutkan ketahap *input* data klien. Setelah selesai menginput data klien, jika jumlah klien yang harus diinputkan datanya belum habis, maka *input* data klien akan dilakukan lagi. Kemudian, jika jumlah data klien yang harus diinputkan sudah habis, akan dikembalikan ketahap *input* data kelas jika masih ada data kelas yang harus diinputkan. Jika jumlah data kelas dan jumlah data klien yang harus diinputkan sudah habis, maka proses pembuatan berkas konfigurasi sudah selesai. Dan jika pada saat sedang melakukan penginputan data, pengguna ingin menghentikan proses tersebut dengan menggunakan tombol *Esc* atau memilih "*Cancel*", pengguna akan ditanyakan apakah ingin menyimpan proses pembuatan berkas konfigurasi dan dilanjutkan nanti. Jika pengguna tidak ingin melanjutkan pembuatan berkas, maka berkas konfigurasi tersebut akan dihapus, dan jika pengguna ingin melanjutkan pembuatannya, maka sebuah data - data inputan pengguna yang meliputi nama berkas, jumlah kelas yang belum diinputkan, nama kelas yang terakhir diinputkan, dan jumlah klien yang belum diinputkan akan disimpan kedalam berkas */tmp/HTBCron/progress*. Berkas tersebut akan digunakan oleh fungsi *continuecr8* sebagai dasar untuk melanjutkan pembuatan berkas konfigurasi yang tertinggal.



Gambar 5.23 Desain Tampilan *Input* Fungsi *createrule*

Gambar 5.23 menggambarkan bagaimana tampilan fungsi ini didepan layar pengguna. Masing-masing tahapan *input* akan keluar secara berurut setelah pengguna selesai menginputkan data pada setiap tahap dengan benar. Alur kerja fungsi *createrule*, jika diurutkan secara keseluruhan menjadi :

1) Pengguna menginputkan nama berkas, *maxdefault bandwidth*, dan banyaknya jumlah kelas. Jika pengguna

mengkonfirmasikan inputannya, dilanjutkan ke langkah ke-2, jika pengguna menekan tombol *Esc* atau memilih "*Cancel*", fungsi *createrule* dihentikan, dan kembali ke menu utama.

2) Prosedur validasi dipanggil untuk mengecek apakah ada *field* yang kosong, apakah nama berkas mengandung simbol atau spasi, *maxdefault bandwidth* bukan angka, dan banyak jumlah kelas bukan angka. Jika iya, maka pesan kesalahan dimunculkan, dan diulang ke langkah ke-1, dan jika tidak, maka akan dilakukan pemeriksaan apakah sudah ada berkas dengan nama tersebut, jika ada, pengguna akan ditanyakan apakah ingin dilakukan penimpaan berkas atau tidak, jika tidak, fungsi *createrule* akan dihentikan dan jika ingin menimpa atau berkas dengan nama tersebut tidak ada, proses akan dilanjutkan ke tahap berikutnya.
3) *Maxdefault bandwidth* dituliskan kedalam berkas konfigurasi *HTB-tools* dengan nama sesuai dengan isi dari variabel nama berkas, dan berkas disimpan didalam direktori */bin/HTBCron/rules/* . Kemudian proses *input* data dilanjutkan ke langkah ke-4.
4) Pengguna menginputkan nama kelas, alokasi *bandwidth* minimal, alokasi *bandwidth* maksimal, tingkap prioritas data, dan jumlah klien yang dimiliki kelas ini. Jika pengguna mengkonfirmasikan inputannya, proses dilanjutkan ke

langkah ke-5, dan jika pengguna menekan tombol Esc atau memilih "Cancel", maka dilanjutkan ke langkah ke-6.

5) Prosedur validasi dipanggil untuk memeriksa apakah ada *field* yang kosong, apakah nama kelas memiliki simbol atau spasi, apakah *bandwidth* minimal, *bandwidth* maksimal, tingkat prioritas data, dan jumlah klien bukan angka. Jika iya, pesan kesalahan dimunculkan, dan dikembalikan ke langkah ke-4, jika tidak, akan dilanjutkan ke langkah ke-7.
6) Pengguna akan ditanyakan untuk menyimpan data berkas atau tidak, jika tidak, berkas konfigurasi *HTB-tools* dengan namanya sesuai dengan isi dari variabel nama berkas akan dihapus, kemudian fungsi *createrule* dihentikan dan kembali ke menu utama. Jika ingin menyimpan dan melanjutkannya nanti, nama berkas yang sedang dibuat, jumlah kelas yang belum diinput datanya, nama kelas yang datany diinputkan terakhir kali, dan jumlah klien yang belum diinputkan datanya disimpan kedalam berkas */tmp/HTBCron/progress*, fungsi *createrule* dihentikan, dan kembali ke menu utama.
7) Nama kelas, *bandwidth* minimal, *bandwidth* maksimal, dan prioritas data dituliskan kedalam berkas konfigurasi *HTB-tools* dengan nama sesuai dengan nama berkas, yang tersimpan didalam direktori */bin/HTBCron/rules/* , dan jumlah kelas dikurangi 1. Kemudian proses dilanjutkan ke langkah ke-8.

8) Pengguna menginputkan nama atau identitas klien, alokasi *bandwidth* minimum, alokasi *bandwidth* maksimum, tingkat prioritas data, *IP address*, dan *netmask*. Jika pengguna mengkonfirmasikan inputannya, proses dilanjutkan ke langkah ke-9, jika pengguna menekan tombol *Esc* atau memilih "*Cancel*", maka dilanjutkan ke langkah ke-6.
9) Fungsi *ubahipadd* dijalankan untuk menentukan tipe *IP address* inputan dari pengguna berdasarkan *netmask*-nya. Hasil *output* dari fungsi *ubahipadd* disimpan sebagai *IP address* baru dengan *format* penulisan CIDR. Setelah itu dilanjutkan ke langkah ke-10.
10) Prosedur validasi dipanggil untuk memeriksa apakah ada *field* yang kosong, apakah nama atau identitas klien menggunakan simbol atau spasi, apakah *bandwidth* minimal atau maksimal, tingkat prioritas data, atau *IP address* mengandung karakter bukan angka, apakah besar *bandwidth* minimal atau maksimal klien melebihi *bandwidth* minimal atau maksimal kelas, apakah nilai variabel *IP address* sama dengan 0. Jika iya, akan ditampilkan pesan kesalahan, dan dikembalikan ke langkah ke-8, jika tidak dilanjutkan ke tahap-11.
11) Nama atau identitas klien, *bandwidth* minimal, *bandwidth* maksimal, prioritas data, dan *IP address* dituliskan kedalam berkas konfigurasi *HTB-tools* dengan nama sesuai dengan

nama berkas, yang tersimpan didalam direktori */bin/HTBCron/rules/* , dan jumlah klien dikurangi 1. Kemudian proses dilanjutkan ke langkah ke-12.

12) Jika jumlah data klien yang belum diinputkan lebih dari 0, maka kembali ke langkah ke-8. Jika jumlah data klien yang belum diinputkan sama dengan 0, dilanjutkan ke langkah ke-13.
13) Jika jumlah data kelas yang belum diinputkan lebih dari 0, maka kembali ke langkah ke-4. Jika jumlah data kelas yang belum diinputkan sama dengan 0, fungsi *createrule* dihentikan, dan kembali ke menu utama.

5.2.3 Fungsi *continuecr8*

Fungsi *continuecr8* dibandingkan dengan fungsi *createrule*, hanya memiliki 2 tahapan *input* data, yakni :

1) *Input* Data Kelas, yang meliputi nama kelas, alokasi *bandwidth* minimal, alokasi *bandwidth* maksimal, tingkat prioritas data, dan banyak klien.
2) *Input* Data Klien, yang meliputi nama atau identitas klien, alokasi *bandwidth* minimal, alokasi *bandwidth* maksimal, tingkat prioritas data, *IP address*, dan *netmask*.

Tahapan *input* fungsi ini tidak melalui tahap *input* data berkas karena tahapan tersebut sudah digantikan dengan mengambil data

berkas yang ingin dilanjutkan didalam berkas */tmp/HTBCron/progress* . Data yang diambil meliputi nama berkas yang ingin dilanjutkan, jumlah kelas yang belum diinputkan datanya, nama kelas yang terakhir kali diinputkan, dan jumlah data klien yang belum diinputkan. Jadi pengguna tidak perlu lagi melakukan *input* data berkas.

Tampilan *form input* data yang ditampilkan kepada pengguna, sama seperti tahap *input* data kelas dan tahap *input* data klien pada Gambar 5.23.

Alur kerja fungsi *continuecr8*, jika diurutkan secara keseluruhan menjadi :

1) Pada saat fungsi dijalankan, fungsi ini mengambil isi dari berkas */tmp/HTBCron/progress* . Yang meliputi nama berkas, jumlah kelas, nama kelas, dan jumlah klien.
2) Jika jumlah klien lebih dari 0, pengguna harus menyelesaikan *input* data klien terlebih dahulu dengan dimana proses dilanjutkan ke langkah ke-7 , dan jika jumlah klien sama dengan 0 dan jumlah kelas lebih dari 0, maka dilanjutkan ke langkah ke-3 .
3) Pengguna menginputkan nama kelas, alokasi *bandwidth* minimal, alokasi *bandwidth* maksimal, tingkap prioritas data, dan jumlah klien yang dimiliki kelas ini. Jika pengguna mengkonfirmasikan inputannya, proses dilanjutkan ke

langkah ke-4, dan jika pengguna menekan tombol *Esc* atau memilih "*Cancel*", maka dilanjutkan ke langkah ke-5.

4) Prosedur validasi dipanggil untuk memeriksa apakah ada *field* yang kosong, apakah nama kelas memiliki simbol atau spasi, apakah *bandwidth* minimal, *bandwidth* maksimal, tingkat prioritas data, dan jumlah klien bukan angka. Jika iya, pesan kesalahan dimunculkan, dan dikembalikan ke langkah ke-3, jika tidak, akan dilanjutkan ke langkah ke-6.
5) Pengguna akan ditanyakan untuk menyimpan data berkas atau tidak, jika tidak, berkas konfigurasi *HTB-tools* dengan namanya sesuai dengan isi dari variabel nama berkas akan dihapus, kemudian fungsi *continuecr8* dihentikan dan kembali ke menu utama. Jika ingin menyimpan dan melanjutkannya nanti, nama berkas yang sedang dibuat, jumlah kelas yang belum diinput datanya, nama kelas yang datany diinputkan terakhir kali, dan jumlah klien yang belum diinputkan datanya disimpan kedalam berkas */tmp/HTBCron/progress*, fungsi *continuecr8* dihentikan, dan kembali ke menu utama.
6) Nama kelas, *bandwidth* minimal, *bandwidth* maksimal, dan prioritas data dituliskan kedalam berkas konfigurasi *HTB-tools* dengan nama sesuai dengan nama berkas, yang tersimpan didalam direktori */bin/HTBCron/rules/* , dan

jumlah kelas dikurangi 1. Kemudian proses dilanjutkan ke langkah ke-7.

7) Pengguna menginputkan nama atau identitas klien, alokasi *bandwidth* minimum, alokasi *bandwidth* maksimum, tingkat prioritas data, *IP address*, dan *netmask*. Jika pengguna mengkonfirmasikan inputannya, proses dilanjutkan ke langkah ke-8, jika pengguna menekan tombol *Esc* atau memilih "*Cancel*", maka dilanjutkan ke langkah ke-5.
8) Fungsi *ubahipadd* dijalankan untuk menentukan tipe *IP address* inputan dari pengguna berdasarkan *netmask*-nya. Hasil *output* dari fungsi *ubahipadd* disimpan sebagai *IP address* baru dengan *format* penulisan CIDR. Setelah itu dilanjutkan ke langkah ke-9.
9) Prosedur validasi dipanggil untuk memeriksa apakah ada *field* yang kosong, apakah nama atau identitas klien menggunakan simbol atau spasi, apakah *bandwidth* minimal atau maksimal, tingkat prioritas data, atau *IP address* mengandung karakter bukan angka, apakah besar *bandwidth* minimal atau maksimal klien melebihi *bandwidth* minimal atau maksimal kelas, apakah nilai variabel *IP address* sama dengan 0. Jika iya, akan ditampilkan pesan kesalahan, dan dikembalikan ke langkah ke-7, jika tidak dilanjutkan ke tahap-10.

10) Nama atau identitas klien, *bandwidth* minimal, *bandwidth* maksimal, prioritas data, dan *IP address* dituliskan kedalam berkas konfigurasi *HTB-tools* dengan nama sesuai dengan nama berkas, yang tersimpan didalam direktori */bin/HTBCron/rules/* , dan jumlah klien dikurangi 1. Kemudian proses dilanjutkan ke langkah ke-12.
11) Jika jumlah data klien yang belum diinputkan lebih dari 0, maka kembali ke langkah ke-8. Jika jumlah data klien yang belum diinputkan sama dengan 0, dilanjutkan ke langkah ke-12.
12) Jika jumlah data kelas yang belum diinputkan lebih dari 0, maka kembali ke langkah ke-4. Jika jumlah data kelas yang belum diinputkan sama dengan 0, berkas */tmp/HTBCron/progress* dihapus, dan fungsi *continuecr8* dihentikan, dan kembali ke menu utama.

5.2.4 Fungsi *ubahipadd*

Fungsi ini tidak berhubungan langsung dengan pengguna, melainkan berhubungan langsung dengan fungsi *createrule* dan fungsi *continuecr8*. Pada saat ingin menentukan tipe *IP address* dan *netmask* yang diinputkan pengguna adalah *Network ID* atau *Host ID*, dan mengubah bentuk penulisannya menjadi *format*

penulisan CIDR, fungsi ini dipanggil dengan *argument* pertamanya adalah *IP address* dan *argument* keduanya adalah *netmask*.

Alur kerja fungsi *ubahipadd*, jika diurutkan secara keseluruhan menjadi :

1) Fungsi *ubahipadd* dijalankan dengan *argument*-nya *IP address* dan *netmask*. Masing-masing *argument* dibagi menjadi 4 bagian sesuai desimalnya.
2) Desimal pertama dari *netmask* dicek nilainya apakah bernilai 128, 192, 224, 240, 248, 252, atau 254. Jika iya maka dilanjutkan ke langkah ke-3. Jika desimal pertama *netmask* bernilai 255, dilanjutkan ke langkah ke-4. Dan jika desimal pertama *netmask* tidak termasuk nilai manapun, maka *netmask* tersebut berstatus "salah".
3) Jika desimal pertama *IP address* adalah 0, maka status *IP address* tersebut adalah "*Network ID*". Atau jika desimal pertama *IP address* adalah kelipatan dari FPB antara nilai desimal *netmask* pertama dengan 256, maka status *IP address* tersebut adalah "*Network ID*". Dan jika dari kedua perbandingan tersebut tidak ada yang benar, maka status *IP address* tersebut adalah "*Host ID*".
4) Desimal kedua dari *netmask* dicek apakah bernilai 0. Jika iya, dilakukan perbandingan apakah desimal kedua *IP address* sama dengan 0, jika iya, maka status *IP address* adalah "*Network ID*", jika tidak, maka status adalah "*Host ID*". Jika

tidak, dilakukan pengecekan nilai desimal kedua *netmask* apakah sama dengan 128, 192, 224, 240, 248, 252, atau 254, jika iya, maka dilanjutkan ke langkah ke-5, jika tidak, dicek apakah nilai tersebut sama dengan 255, jika iya bernilai 255, maka dilanjutkan ke langkah ke-6. Dan jika nilai desimal kedua *netmask* tidak cocok dengan nilai yang disebutkan, maka *netmask* tersebut berstatus "salah".

5) Jika desimal kedua *IP address* adalah 0, maka status *IP address* tersebut adalah "*Network ID*". Atau jika desimal kedua *IP address* adalah kelipatan dari FPB antara nilai desimal kedua dengan 256, maka status *IP address* tersebut adalah "*Network ID*". Dan jika dari kedua perbandingan tersebut tidak ada yang benar, maka status *IP address* tersebut adalah "*Host ID*".
6) Langkah pengecekan dan perbandingan dari nilai desimal ketiga *netmask* dan nilai desimal ketiga *IP address* dilakukan sama dengan perbandingan desimal kedua. Dimana jika nilai desimal ketiga *netmask* sama dengan salah satu dari 0, 128, 192, 224, 240, 248, 252, 254, maka perbandingan antara desimal ketiga *IP address* dan desimal ketiga dari *netmask* dilakukan untuk menentukan status *IP address*. Jika desimal ketiga *netmask* bernilai 255, maka perbandingan desimal keempat akan dilakukan. Dan jika pada saat dilakukan pengecekan nilai desimal keempat *netmask* bernilai 255,

maka status *IP address* adalah "*Host ID*". Setelah status *IP address* ditentukan, proses dilanjutkan ke langkah ke-7.

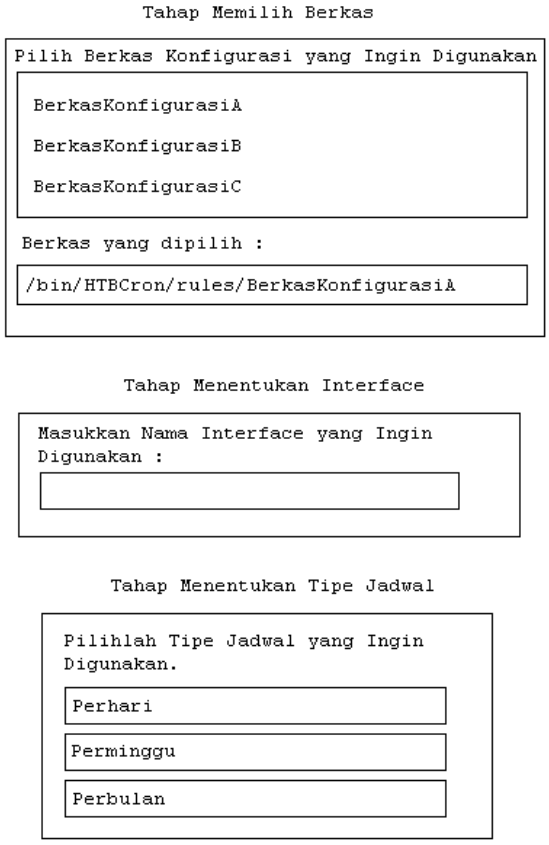
7) Fungsi *ubahipadd* akan memberikan *output* berdasarkan status dari *IP address*. Jika statusnya adalah "salah", maka fungsi ini akan memberikan *output* nilai 0. Jika statusnya adalah "*Host ID*", maka fungsi ini akan memberikan keluaran berupa *IP address*/32. Dan Jika statusnya adalah "*Network ID*", maka fungsi ini akan memberikan keluaran berupa *IP address*/*cidr*. Nilai *cidr* didapat dari *netmask* yang diubah bentuk penulisannya menjadi *format* penulisan CIDR.

5.2.5 Fungsi *settime*

Fungsi *settime* dijalankan pada saat pengguna ingin mendaftarkan jadwal penukaran berkas konfigurasi *HTB-tools* pada waktu tertentu. Sebelum menentukan jadwalnya, pengguna harus menentukan dulu *network interface* dan berkas konfigurasi yang akan digunakan. Setelah itu, pengguna akan diberikan 3 macam jadwal yang bisa digunakan, yakni perhari, perminggu, dan perbulan. Tipe penjadwalan perhari, akan menukar berkas konfigurasi yang baru dengan yang lama pada saat jam yang ditentukan telah tiba. Tipe penjadwalan perminggu, akan menukar berkas konfigurasi yang baru dengan yang lama pada saat jam dan hari yang ditentukan telah tiba, dimana hari yang ditentukan bisa

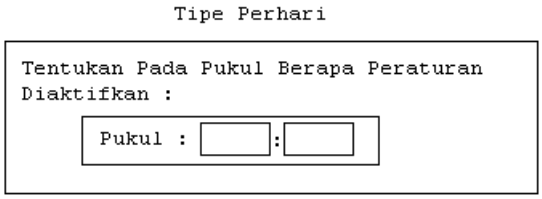
lebih dari 1. Tipe penjadwalan perbulan, akan mengaktifkan fungsi penukaran berkas konfigurasi pada tanggal dan waktu yang telah ditentukan.

Tahapan yang harus dilalui pengguna pada saat menetapkan jadwal dimulai dari pemilihan berkas konfigurasi yang akan digunakan, menginput nama *network interface* yang akan diimplementasikan peraturan, memilih tipe jadwal, dan menentukan waktunya. Setelah tahapan-tahapan tersebut dilalui tanpa ada masalah, sebuah *entry* baru akan dimasukkan kedalam tabel *cron* yaitu berkas */etc/crontab*. Pada halaman berikut rancangan bentuk tampilan fungsi *settime* pada pengguna.

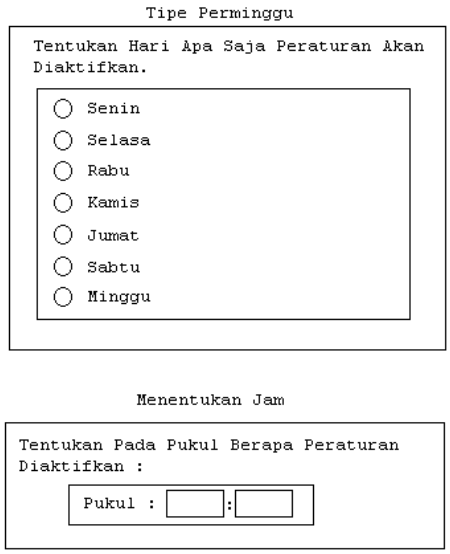


Gambar 5.24 Desain Tampilan Fungsi *settime*

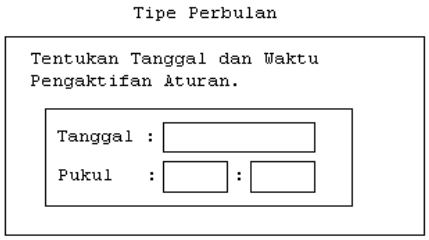
Gambar diatas adalah bentuk tampilan fungsi *settime* pada bagian awal saat pengguna menginputkan nama berkas, nama *network interface*, dan pemilihan tipe jadwal. Untuk bentuk tampilan dari masing-masing tipe jadwal, bisa dilihat pada gambar - gambar berikutnya.



Gambar 5.25 Desain Tampilan Tipe Jadwal Harian



Gambar 5.26 Desain Tampilan Tipe Jadwal Mingguan



Gambar 5.27 Desain Tampilan Tipe Jadwal Bulanan

Alur kerja fungsi *settime*, jika diurutkan secara keseluruhan, menjadi :

1) Pada saat fungsi dijalankan, nilai variabel menit, jam, tanggal, bulan, dan hari ditentukan sebagai asterik (*). Hal ini dilakukan untuk menentukan nilai awal dari variabel-variabel tersebut, dimana nilainya nanti akan berubah setelah pengguna menginputkan datanya. Lalu dilanjutkan ke langkah ke-2.
2) Pengguna menginputkan nama berkas konfigurasi yang akan digunakan, sesuai dengan berkas - berkas yang ditampilkan pada daftar berkas yang ada. Setelah pengguna mengkonfirmasikan inputannya, hasil inputan tersebut diperiksa. Jika inputan pengguna kosong atau berkas yang ingin digunakan tidak ada, akan muncul pesan kesalahan dan *input* harus diulangi. Jika inputan pengguna tidak terdapat kesalahan, dilanjutkan ke langkah ke-2.
3) Pengguna menginputkan nama *network interface* yang akan digunakan. Kemudian inputan pengguna akan diperiksa apakah *network interface* yang ingin digunakan ada atau tidak. Jika tidak ada, pesan kesalahan akan dimunculkan, dan *input* harus diulangi, jika ada, dilanjutkan ke langkah ke-3.
4) Pengguna memilih salah satu tipe penjadwalan yang ingin digunakan. Jika pengguna memilih tipe perhari, maka proses dilanjutkan ke-4a, jika memilih tipe perminggu, maka proses

dilanjutkan ke-4b, jika memilih tipe perbulan, maka proses dilanjutkan ke-4c.

(4a) Pengguna menginputkan jam dan menit tepatnya berkas konfigurasi tersebut akan ditukar dengan berkas yang lama. Hasil inputan pengguna diperiksa apakah ada *field* yang kosong, apakah ada karakter bukan angka, atau apakah nilai waktunya sudah benar. Jika ya, pesan kesalahan akan dimunculkan, *input* harus diulangi, dan jika tidak, maka dilanjutkan ke langkah ke-5.

(4b) Pengguna memilih hari apa saja berkas konfigurasi akan ditukar, dan jika pengguna tidak memilih hari apapun, pesan peringatan akan dimunculkan. Kemudian pengguna harus menginputkan waktu tepatnya berkas konfigurasi akan ditukar. Inputan waktu akan diperiksa kebenarannya, jika terdapat kesalahan, menginput waktu harus diulangi, dan jika sudah benar, dilanjutkan ke langkah ke-5.

(4c) Pengguna menginputkan tanggal dan waktu kapan berkas konfigurasi akan ditukar. Hasil inputan pengguna akan diperiksa kebenarannya, jika terdapat kesalahan, pengguna akan diberitahu bahwa terdapat kesalahan *input*, *input* harus diulangi, dan jika tidak ada kesalahan, maka dilanjutkan ke langkah ke-5.

5) Hasil inputan pengguna dimasukkan sebagai *entry* baru kedalam berkas */etc/crontab*, dengan bentuk penulisan sesuai

dengan ketentuan *entry* tabel *cron*, yakni menit, jam, tanggal, bulan, hari, pengguna (*root*), dan perintah penukaran berkas. Setelah itu fungsi *settime* dihentikan dan pengguna dikembalikan ke menu utama.

Pada langkah ke-5, bagian *entry* tabel *cron* berupa perintah penukaran berkas adalah perintah menjalankan fungsi *rulechanger* dengan *argument* pertamanya adalah *network interface* yang digunakan, dan *argument* keduanya adalah berkas konfigurasi yang dipilih.

5.2.6 Fungsi *rulechanger*

Fungsi ini dijalankan pada saat jadwal penukaran berkas konfigurasi *HTB-tools* yang didaftarkan kedalam tabel *cron* telah tiba. Pada saat beroperasi, fungsi ini akan mencatat kegiatannya pada berkas *log*, yang nama berkas *log*-nya sesuai dengan bulan dan tahun pada saat fungsi ini dijalankan. Hal ini dilakukan agar pengguna dapat mengikuti aktifitas dari penukaran berkas konfigurasi *HTB-tools* dari waktu ke waktu, dan lebih mudah untuk mencari dan mengecek hasil dari kegiatan penukaran berkas konfigurasi pada kurun waktu tertentu.

Alur kerja fungsi ini, jika diurutkan secara keseluruhan, menjadi :

1) Pada saat fungsi dijalankan, nilai variabel nama berkas *log* diambil berdasarkan bulan dan tahun pada saat itu.
2) Servis *HTB-tools* pada *network interface* yang telah ditentukan pada *argument* pertama pada saat fungsi dijalankan, dihentikan, dan hasil penghentiannya dicatat kedalam berkas *log* yang sudah ditentukan.
3) Berkas konfigurasi yang sudah ditentukan dari *argument* kedua pada saat fungsi ini dijalankan, diduplikasikan kedalam direktori */etc/htb/* , dengan nama xxxx-qos.cfg (xxxx adalah nama *network interface* yang ditentukan). Kegiatan ini dicatat kedalam berkas *log* jika proses penduplikasian berhasil dilakukan.
4) Servis *HTB-tools* pada *network interface* tersebut dijalankan kembali, dan kegiatan ini dicatat juga hasilnya kedalam berkas *log*.
5) Satu baris kosong disisipkan kedalam berkas *log*, untuk memisahkan pencatatan hasil kegiatan yang sekarang dengan pencatatan hasil kegiatan yang akan datang. Dan fungsi *rulechanger* dihentikan.

5.2.7 Fungsi *config*

Fungsi *config* digunakan pada saat pengguna ingin melakukan instalasi *HTB-tools* dan program aplikasi yang

dilaporkan pada skripsi ini. Pada saat fungsi ini dijalankan, berkas - berkas fungsi program aplikasi diduplikasikan kedalam direktori */bin/HTBCron/*. Kemudian *tool-tool* dari *HTB-tools* dipindahkan kedalam direktori - direktori yang sudah ditentukan.

Alur kerja fungsi *config* secara keseluruhan langkah - langkahnya adalah :

1) Dilakukan pengecekan, apakah pengguna menggunakan identitas *root* atau bukan, jika bukan, fungsi *config* dihentikan, dan jika iya, dilanjutkan ke tahap berikutnya.
2) Direktori */bin/HTBCron/* , */bin/HTBCron/rules/* , */bin/HTBCron/log/* diperiksa apakah ada atau tidak, jika tidak ada, direktori-direktori tersebut akan dibuat. Kemudian dilanjutkan ke tahap berikutnya.
3) Berkas fungsi *createrule*, *continuecr8*, *editrule*, *edittimeset*, *menu*, *rulechanger*, *settime*, dan *ubahipadd* diduplikasikan kedalam direktori */bin/HTBCron/* .
4) Berkas *tools* dari *HTB-tools* masing-masing dipindahkan ketempatnya. Berkas *htb*, *htbgen*, *q_checkcfg*, *q_parser*, *q_show* ke direktori /sbin/ , dan berkas *eth0-qos.cfg.new* dan *eth1-qos.cfg.new* dipindahkan kedalam */etc/htb/* dengan kata "*new*" dihilangkan. Jika ada berkas *tool* yang sudah ada pada direktori yang dituju, pengguna akan ditanyakan apakah ingin menimpa berkas yang ada atau tidak, jika ya, berkas yang dimaksudkan akan ditimpa, dan jika tidak, berkas tersebut

tidak akan diduplikasikan. Setelah itu, fungsi *config* dihentikan.

5.2.8 Fungsi *editrule* dan *edittimeset*

Kedua fungsi tambahan ini dijalankan pada saat pengguna ingin melakukan pengubahan isi dari berkas - berkas konfigurasi yang tersimpan atau isi dari berkas */etc/crontab*.

Fungsi *editrule*, pada saat dijalankan, didepan layar pengguna akan ditampilkan pemilihan berkas konfigurasi yang ingin diubah dengan bentuk tampilan sama seperti pemilihan berkas konfigurasi fungsi *settime* pada Gambar 5.23. Setelah pengguna menginputkan nama berkas yang ingin diubah tanpa kesalahan (jika terdapat kesalahan, *input* harus diulangi), berkas konfigurasi tersebut akan dibuka dengan menggunakan *nano text editor*, dan pengguna dapat mengubah isi didalam berkas tersebut.

Dan untuk fungsi edittimeset, pada saat dijalankan, berkas */etc/crontab* yang merupakan tabel *cron* utama akan dibuka dengan menggunakan *nano text editor* juga, dan pengguna dapat mengubah jadwal - jadwal penukaran berkas konfigurasi *HTB-tools* yang terdaftar didalamnya.

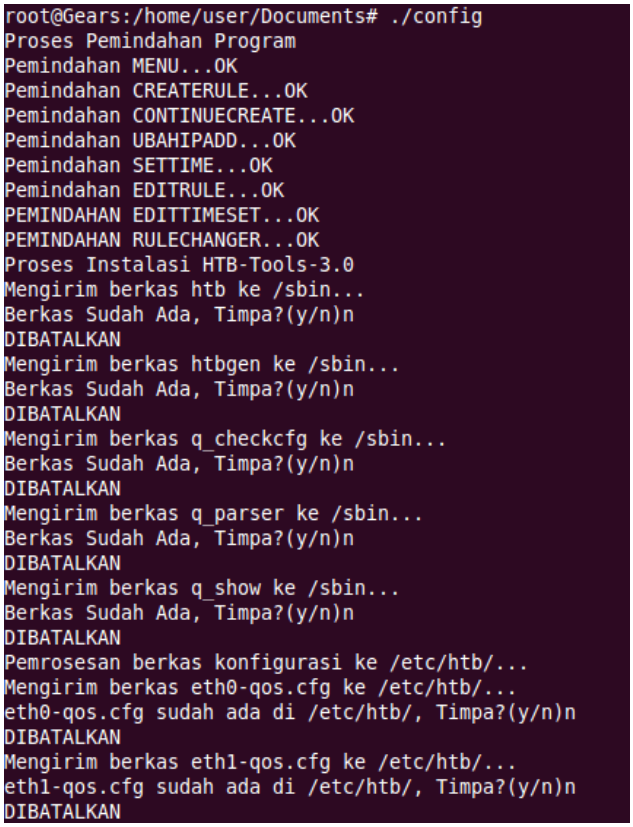
5.3 Penerapan Program

Setelah selesai merancang dan menulis kode masing-masing fungsi sesuai dengan rancangannya dilakukan, tahap berikutnya adalah tahap penerapan program secara keseluruhan untuk melihat apakah program aplikasi sudah bisa berjalan sesuai dengan yang diinginkan oleh penulis.

Program aplikasi ini jika dijalankan mulai dari proses instalasi, maka fungsi *config* dijalankan sebagai *root* dengan menggunakan perintah.

```
#./config
```

Kemudian proses instalasi akan dijalankan. Gambar 5.31 menunjukkan contoh dari proses instalasi.

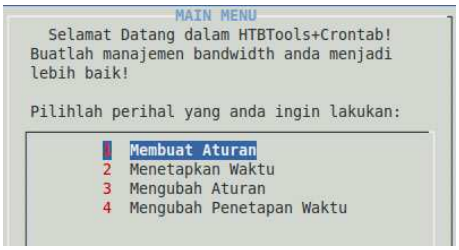


Gambar 5.31 Eksekusi Fungsi *config*

Setelah proses instalasi diselesaikan, program aplikasi ini sudah bisa digunakan. Untuk menjalankan program ini bisa dimulai dengan perintah :

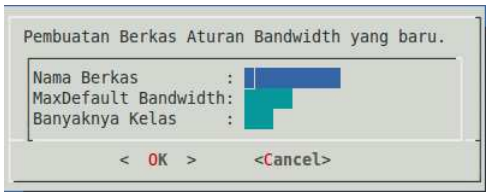
```
#sh /bin/HTBCron/menu
```

Kemudian akan muncul tampilan menu utama seperti pada Gambar 5.32 pada halaman selanjutnya.



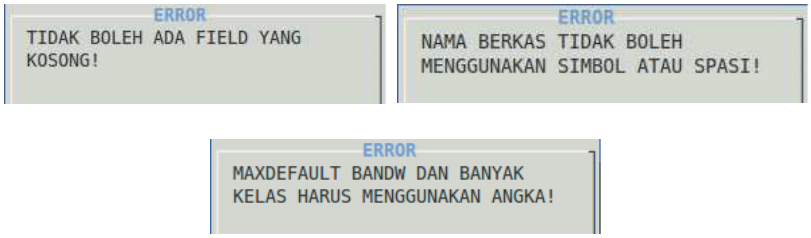
Gambar 5.32 Tampilan Menu Utama

Jika pengguna memilih "Membuat Aturan", maka fungsi *createrule* akan dipanggil dan proses pembuatan berkas konfigurasi akan dimulai. Kotak dialog yang pertama dimunculkan adalah untuk *input* data berkas.



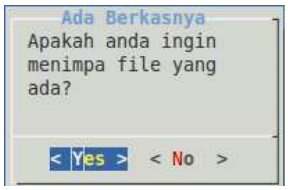
Gambar 5.33 *Form Input* Data Berkas

Jika terdapat kesalahan dalam penginputan data berkas, akan muncul salah satu pesan kesalahan seperti pada Gambar 5.34.



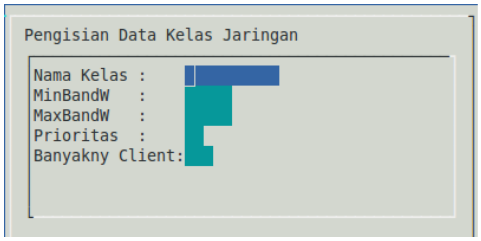
Gambar 5.34 *Error Input* Data Berkas

Kemudian, bila terdapat berkas dengan nama sama dengan yang diinputkan pengguna, maka akan muncul kotak dialog seperti pada Gambar 5.35.



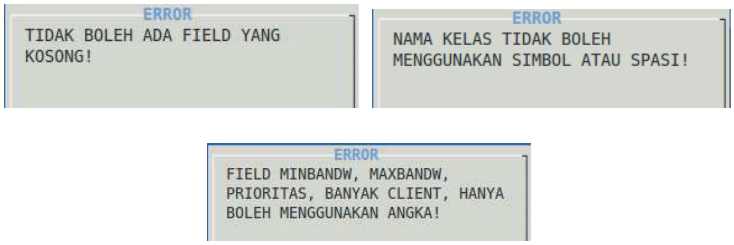
Gambar 5.35 Pesan Penimpaan Berkas

Setelah *input* data berkas selesai dilakukan, maka *form* untuk menginput data kelas akan muncul.



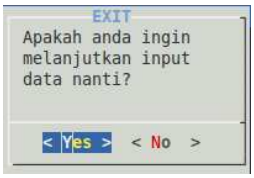
Gambar 5.36 *Form Input* Data Kelas

Setelah pengguna menginput data, dan bila terdapat kesalahan dalam salah satu *field* didalam *form*, akan muncul pesan kesalahan.



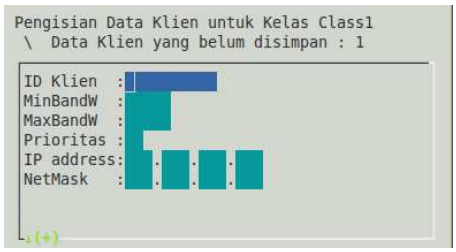
Gambar 5.37 *Error Input* Data Kelas

Dan jika pengguna memilih "*Cancel*" atau menekan tombol *Esc*, akan muncul pesan untuk menyimpan pembuatan berkas.



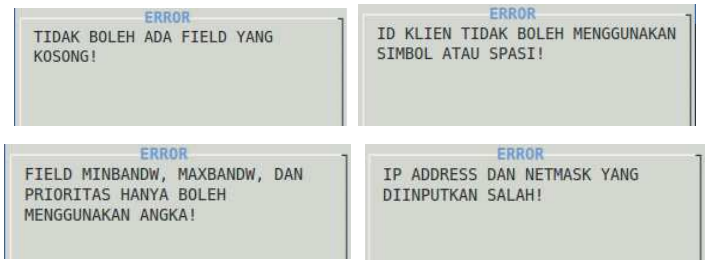
Gambar 5.38 Menyimpan Pembuatan Berkas

Setelah konfirmasi data-data kelas, langkah selanjutnya adalah *input* data klien, akan muncul *form* seperti pada Gambar 5.39.



Gambar 5.39 *Form Input* Data Klien

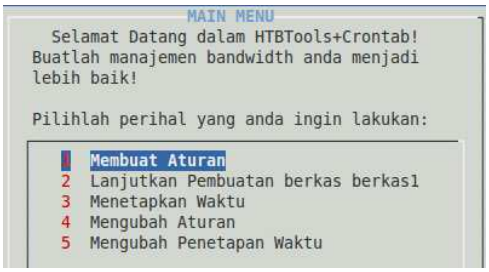
Pada saat validasi data inputan, jika terdapat kesalahan didalam salah satu *field*, kesalahan tersebut akan diberitahukan kepada pengguna melalui salah satu kotak dialog seperti berikut ini.



Gambar 5.310 *Error Input* Data Klien

Jika pengguna memilih "*Cancel*" atau menekan tombol *Esc*, akan keluar pesan seperti pada Gambar 5.38 yang juga berfungsi untuk menyimpan data-data berkas yang bisa dilanjutkan kemudian.

Apabila pengguna melakukan penyimpanan data berkas yang pembuatan berkasnya akan dilanjutkan lagi, maka tampilan menu utama akan berubah menjadi seperti pada Gambar 5.311.

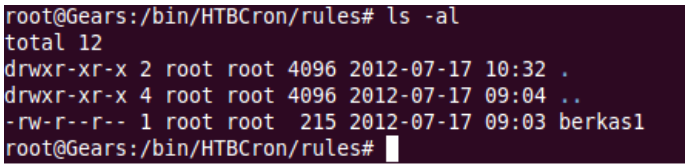


Gambar 5.311 Tampilan Menu Utama ke-2

Untuk melanjutkan pembuatan berkas yang terakhir kali disimpan yang belum diselesaikan, dengan memilih pilihan "Lanjutkan Pembuatan berkas ...", didepan layar komputer pengguna akan ditampilkan lagi *form input* data kelas atau *form input* data klien seperti pada Gambar 5.36 atau Gambar 5.39. Apabila terjadi kesalahan dalam melakukan *input* data, akan keluar pesan kesalahan seperti pada Gambar 5.37 dan Gambar 5.310. Dan

jika pengguna memilih *Cancel* atau menekan tombol *Esc*, akan muncul kotak dialog untuk yang menanyakan pengguna untuk menyimpan data berkas dan melanjutkannya nanti atau tidak, seperti Gambar 5.38.

Setelah pembuatan berkas konfigurasi diselesaikan, berkas yang tersimpan didalam */bin/HTBCron/rules/* tersebut perlu dijadwalkan pengaktifannya. Pilihan "Menentukan Waktu" pada menu bisa membantu pengguna dalam pendaftaran berkas kedalam */etc/crontab*.

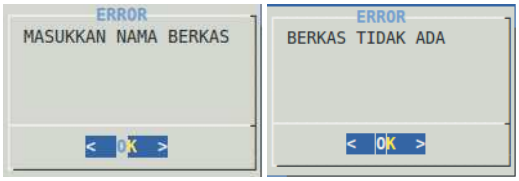


Gambar 5.312 Isi Direktori */bin/HTBCron/rules/*

Jika ingin mengubah isi dari sebuah berkas konfigurasi, pada tampilan menu dengan memilih "Mengubah Aturan", maka didepan layar pengguna akan ditampilkan pemilihan berkas yang ingin diubah.

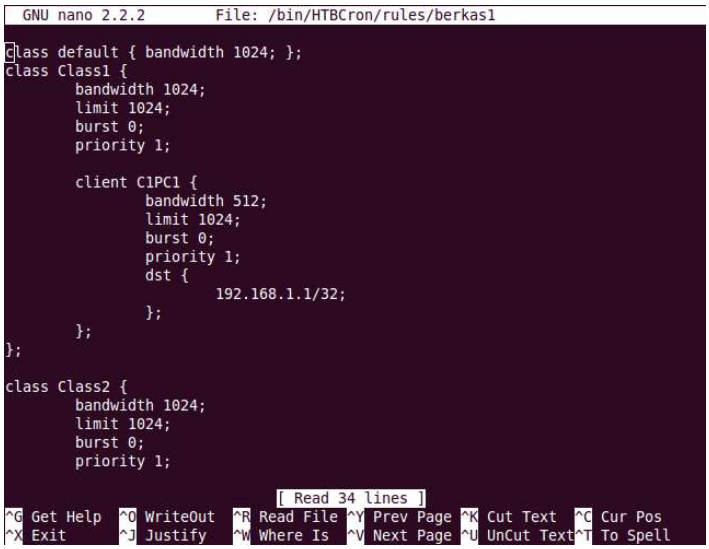
Gambar 5.313 Pemilihan Berkas Konfigurasi pada *editrule*

Setelah selesai menginput nama berkas yang ingin diubah, inputan dari pengguna akan diperiksa lebih dahulu apakah mengandung kesalahan. Jika terdapat kesalahan akan muncul peringatan seperti pada Gambar 5.314,



Gambar 5.314 Kesalahan *Input* Nama Berkas pada *editrule*

dan jika tidak, maka pengubahan berkas yang dipilih akan dilakukan seperti pada Gambar 5.315



Gambar 5.315 Mengubah Berkas Konfigurasi

Setelah proses pembuatan berkas selesai dilakukan, tahap berikutnya adalah penetapan jadwal kapan sebuah berkas konfigurasi yang telah dibuat akan ditetapkan kedalam jaringan. Dengan memilih pilihan

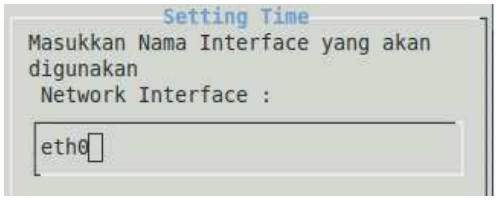
"Menetapkan Waktu", dilayar pengguna akan ditampilkan tampilan seperti pada Gambar 5.316.

Gambar 5.316 Pemilihan Berkas Konfigurasi pada *settime*

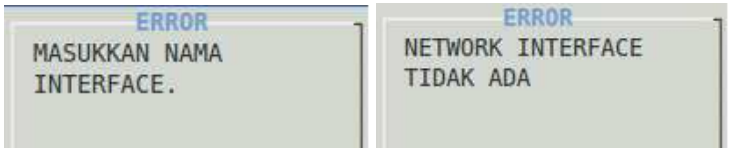
Jika pada saat pemilihan berkas konfigurasi, pengguna lupa memasukkan berkas yang digunakan, atau berkas yang diinputkan tidak ada, maka akan keluar pesan seperti pada Gambar 5.317.

Gambar 5.317 *Error* Pemilihan Berkas Konfigurasi

Kemudian, langkah berikutnya adalah menginputkan nama network interface yang ingin digunakan. Akan muncul kotak dialog seperti pada Gambar 5.318, dan jika *network interface* tidak diberikan *input*, atau tidak ada, akan keluar pesan kesalahan seperti pada Gambar 5.319.

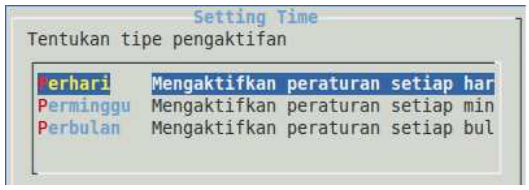


Gambar 5.318 *Input* Nama *Interface*



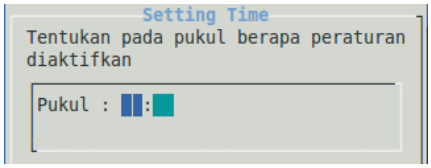
Gambar 5.319 *Error Input* Nama *Interface*

Setelah selesai menginput nama network interface yang akan digunakan, akan muncul tampilan menu untuk memilih tipe jadwal yang ingin digunakan, seperti pada Gambar 5.320.

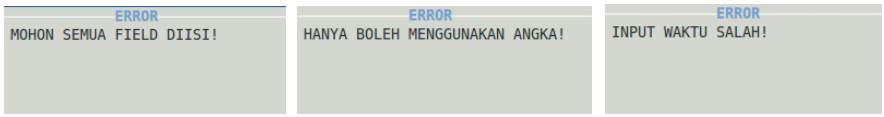


Gambar 5.320 Menu Tipe Jadwal

Untuk pilihan "Perhari", akan muncul kotak dialog lagi seperti pada Gambar 5.321, dan bila inputan pengguna terdapat kesalahan, akan diberikan pesan kesalahan seperti pada Gambar 5.322.



Gambar 5.321 Penentuan Jam

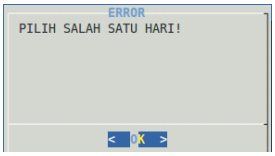


Gambar 5.322 *Error* Penentuan Jam

Untuk pilihan "Perminggu", akan muncul kotak dialog dengan jenis *radiobox*, dimana pengguna bisa memilih lebih dari 1 hari untuk jadwal aktifasinya, seperti Gambar 5.323.

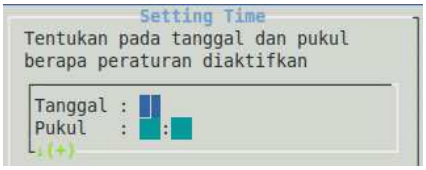
Gambar 5.323 Penentuan Hari Dalam Satu Minggu

Jika pengguna lupa untuk memilih minimal satu hari, akan keluar pesan peringatan seperti gambar berikut.



Gambar 5.324 *Error* Penentuan Hari

Setelah penentuan hari dilakukan, berikutnya adalah penentuan jam pada hari yang telah dipilih. Kotak dialog penentuan jam seperti pada Gambar 5.321 akan dimunculkan. Untuk pilihan "Perbulan", akan dimunculkan kotak dialog *form input* penentuan tanggal seperti Gambar 5.325 dimana didalam kotak tersebut, penentuan jam langsung dilakukan.



Gambar 5.325 Penentuan Tanggal dan Waktu

Jika didalam *form input* penentuan tanggal dan waktu terdapat kesalahan, maka akan muncul pesan kesalahan seperti gambar berikut.

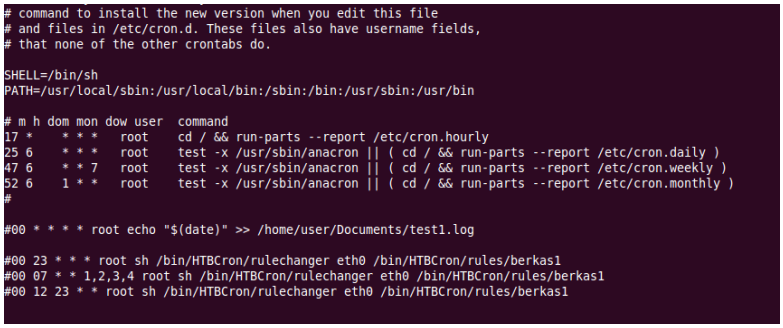
Gambar 5.326 *Error* Penentuan Tanggal dan Waktu

Entry akan langsung dimasukkan kedalam berkas */etc/crontab* setelah inputan dari pengguna sudah melewati proses validasi dan hasilnya tidak terdapat kesalahan. Berikut contoh dari *entry* didalam berkas *crond* berdasarkan masing-masing tipe, dari tipe harian, mingguan, dan bulanan.

```
00 23 * * * root sh /bin/HTBCron/rulechanger eth0 /bin/HTBCron/rules/berkas1
00 07 * * 1,2,3,4 root sh /bin/HTBCron/rulechanger eth0 /bin/HTBCron/rules/berkas1
00 12 23 * * root sh /bin/HTBCron/rulechanger eth0 /bin/HTBCron/rules/berkas1
```

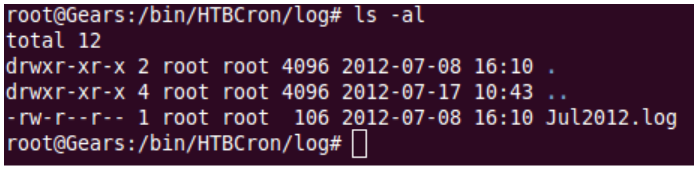
Gambar 5.327 Isi Berkas */etc/crontab*

Jika pengguna ingin melakukan pengubahan penetapan waktu yang telah ditentukan, pengguna bisa memilih pilihan "Mengubah Penetapan Waktu" pada tampilan menu. Setelah memilih pilihan tersebut, pengguna akan dihadapkan langsung dengan isi dari berkas */etc/crontab* seperti pada Gambar 5.328.



Gambar 5.328 Pengubahan Isi Berkas */etc/crontab*

Kemudian, pada saat waktu yang ditentukan telah tiba, fungsi *rulechanger* akan diaktifkan dan hasil kegiatannya akan disimpan kedalam *logfile* dengan nama bulan dan tahun pada saat itu. Tempat penyimpanan berkas log terdapat pada direktori */bin/HTBCron/log/*.



Gambar 5.329 Isi Direktori */bin/HTBCron/log/*

Pada contoh gambar diatas, fungsi *rulechanger* diaktifkan pada bulan Juli tahun 2012, sehingga isi kegiatan dari *rulechanger* disimpan kedalam berkas *log* dengan nama *Jul2012.log*. Berikut isi didalam berkas log *Jul2012.log*, yang setiap masukkannya dimulai dari tanggal lengkapnya fungsi *rulechanger* diaktifkan, proses mematikan *HTB-tools* pada *interface* yang ditentukan, menduplikasikan berkas konfigurasi menjadi (nama *interface*)*-qos.cfg* kedalam direktori */etc/htb/*, dan menjalankan kembali servis *HTB-tools* pada *interface* tersebut dengan peraturan pembagian *bandwidth* yang baru.

Gambar 5.330 Isi Sebuah *Logfile*

5.4 Kelebihan dan Kekurangan Perangkat Lunak

5.4.1 Kelebihan

Kelebihan yang bisa didapatkan oleh pengguna pada saat menggunakan perangkat lunak ini adalah:

1) Dengan menggunakan fungsi pembuatan berkas konfigurasi, pengguna bisa membuat berkas konfigurasi untuk *HTB-tools* secara cepat dan mudah.
2) Pengguna bisa dengan mudah menjadwalkan pengeksekusian servis *HTB-tools* kedalam tabel cron dengan bantuan salah satu fungsi dari perangkat lunak untuk mengaktifkan servis dari *HTB-tools* pada jaringan komputer tertentu secara otomatis.
3) Dengan kemampuan untuk menjalankan servis *HTB-tools* secara otomatis, pengguna dapat membuat kebijakan manajemen *bandwidth* jaringan komputer berubah-ubah sesuai dengan kebutuhan pada waktu-waktu yang ditentukan.

5.4.2 Kekurangan

Kekurangan yang ada pada saat menggunakan perangkat lunak ini terletak pada :

1) Dikarenakan *HTB-tools* yang digunakan hanya bisa dijalankan pada sistem operasi *Linux Ubuntu* untuk prosesor *Intel*, saat ini perangkat lunak ini tidak bisa digunakan untuk sistem operasi *Linux* untuk prosesor lainnya.
2) Jika didalam sistem operasi pengguna belum dilakukan instalasi paket *Dialog v1.1* keatas, perangkat lunak ini tidak bisa digunakan.
3) Untuk membuat sebuah manajemen *bandwidth* yang kebijakannya dapat berubah-ubah pada waktu tertentu, pengguna harus terlebih dahulu membuat lebih dari satu berkas konfigurasi *HTB-tools* dan menjadwal pengaktifan masing-masing berkas kedalam *crontab*.
4) Untuk saat ini perangkat lunak ini hanya bisa diakses pada komputer yang telah dilakukan instalasi didalam sistemnya, dan tidak memiliki fasilitas untuk akses melalui media lainnya.

5.5 Pemeliharaan Program

Agar kinerja dari program yang dibangun oleh penulis ini bisa tetap bekerja dengan baik dalam jangka waktu yang lama, ada pula ketentuan-

ketentuan yang harus ditaati pada saat menggunakan program aplikasi ini. Ketentuan yang harus diikuti yakni :

1) Jika pengguna tidak memahami isi didalam berkas fungsi - fungsi aplikasi ini (*createrule*, *continuecr8*, *ubahipadd*, dst) maupun *tool-tool* dari *HTB-tools* (*htb*, *htbgen*, *q_parser*, dst), sebaiknya tidak melakukan pengubahan dari isi didalam berkas-berkas fungsi tersebut. Karena bila kode tersebut dirubah, bisa menyebabkan fungsi-fungsi tersebut tidak berjalan.
2) Sediakan duplikasi dari program ini sebelum dilakukan instalasi, karena jika pengguna secara tidak sengaja atau sengaja mengubah isi dari berkas-berkas fungsi aplikasi ini maupun *HTB-tools*, dan terjadi tidak bisa digunakannya salah satu aplikasi, dengan menjalankan fungsi instalasi dari berkas yang diduplikasi tadi, pengguna bisa memperbaiki kembali masing-masing berkas baik berkas fungsi program ini ataupun *HTB-tools*.
3) Jika pengguna ingin melakukan pengubahan dari isi berkas konfigurasi *HTB-tools* atau berkas */etc/crontab*, pastikan isi yang telah diubah sudah sesuai dengan *format* penulisan berkas masing-masing. Karena jika tidak sesuai dengan *format* penulisannya, bisa terjadi *crond* tidak bisa beroperasi sama sekali.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan pembahasan pada Bab V, didapatkan bahwa perangkat lunak yang dikembangkan didalam skripsi ini dengan menggunakan bahasa *shell*, bisa mempermudah dan mempercepat penggunanya dalam melakukan pengubahan berkas konfigurasi yang digunakan oleh *HTB-tools* sebagai dasar kebijakan pengalokasian *bandwidth* sesuai dengan jadwal yang telah ditentukan pengguna didalam tabel *cron*. Sehingga dengan menggunakan perangkat lunak ini, pengguna yang tidak terlalu mengerti tentang cara melakukan pembagian *bandwidth* dengan menggunakan sistem operasi *Linux*, bisa melakukannya dengan mudah dan tanpa masalah.

6.2 Saran

Saran dikhususkan bagi mahasiswa yang berminat untuk mengembangkan penelitian yang dilaporkan didalam skripsi ini lebih lanjut. Perangkat lunak ini masih memiliki sejumlah bagian yang bisa dikembangkan. Salah satu pengembangan lebih lanjut dari fungsi-fungsi yang sudah ada didalamnya bisa dengan cara menambahkan fungsi baru

kedalam perangkat lunak. Atau bisa dengan mengembangkan fungsi-fungsi yang sudah ada, contohnya pada fungsi melanjutkan pembuatan berkas konfigurasi, bisa ditambahkan pilihan untuk menyimpan dan melanjutkan lebih dari satu berkas. Atau mengembangkan perangkat lunak ini menjadi sebuah halaman *web*, yang bisa memberikan kemudahan akses bagi pengguna dimanapun mereka berada.

DAFTAR PUSTAKA

Andrew, Jean. 2009. *A+ Guide to Managing and Maintaining Your PC Comprehensive*. Florence : Cengage Learning , Inc.

Fatta, H. A. 2007. *Analisis & Perancangan Sistem Informasi*. Yogyakarta : Andi.

Hermawan, Asep. 2005. *Penelitian Bisnis Paradigma Kuantitatif*. Jakarta : Grasindo.

Johnson. 2011. *Implementasi Pembagian Bandwidth Koneksi Jaringan Internet Menggunakan Hierarchical Token Bucket (HTB) Pada CV. AN Mandiri*. LPKL tidak diterbitkan. Palembang : Program Studi Teknik Informatika STMIK Palcomtech Palembang.

Kuncoro, Mudrajad. 2003. *Metode Riset untuk Bisnis & Ekonomi*. Jakarta : Erlangga.

Mulyanta, E. S. 2008. *Pengenalan Protokol Jaringan Wireless Komputer*. Yogyakarta : Andi.

Munir, R. 2007. *Algoritma & Pemrograman Dalam Bahasa PASCAL dan C*. Bandung : Informatika.

Sofana, I. 2008. *Membangun Jaringan Komputer Mudah Membuat Jaringan Komputer (Wire & Wireless Untuk Pengguna Windows dan Linux.* Bandung : Informatika.

Sopandi, D. 2008. *Instalasi dan Konfigurasi Jaringan Komputer*. Bandung : Informatika.

Sunarto. 2006. *Teknologi Informasi dan Komunikasi*. Jakarta : Grasindo.

Sutarbi, T. 2004. *Analisa Sistem Informasi*. Yogyakarta : Andi.

http://www.linuxjournal.com/article/1189

Raithel, J. 1996. *Scheduled Activity : cron and at*. Linux Journal. diunduh pada 20 Mei 2012.

http://www.linuxjournal.com/article/3290

Keller, M. S. 1999. *Cron : Job Scheduler*. Linux Journal. diunduh pada 20 Mei 2012.

http://www.linuxjournal.com/article/7562

Benita, Y. 2005. *Kernel Korner - Analisys of the HTB Queuing Discipline*. Linux Journal. diunduh pada 17 Mei 2012.

http://www.scribd.com/doc/63542555/traffic-control-htb

Balan, D. G. & Dan Alin Potorac. 2009. *Extended Linux HTB Queuing Discipline Implementation.* Electrical Engineering and Computer Science Faculty. diunduh pada 17 Mei 2012.