IMPLEMENTASI IP VERSI 6
PADA SISTEM OPERASI LINUX
PENGEMBANGAN TOOLS IPv6 BERBASIS SHELL SCRIPT

SKRIPSI

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
Jurusan Teknik Informatika

Disusun Oleh :

Antonius Arief Widiyatmoko

005314003

JURUSAN TEKNIK INFORMATIKA FAKULTAS TEKNIK
UNIVERSITAS SANATA DHARMA
YOGYAKARTA

2005

IMPLEMENTATION OF IP VERSION 6
ON LINUX OPERATING SYSTEM
DEVELOPMENT OF TOOLS IPv6 BASED ON SHELL SCRIPT

A Thesis

Presented as Partial Fulfillment of the Requirements
to Obtain the *Sarjana Teknik* Degree
in Departement of Informatics Technology

by

Antonius Arief Widiyatmoko

005314003

DEPARTEMENT OF INFORMATICS TECHNOLOGY
FACULTY OF ENGINEERING
SANATA DHARMA UNIVERSITY
YOGYAKARTA
2005

HALAMAN PERSETUJUAN

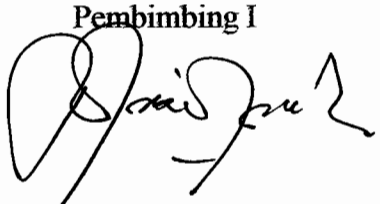
S K R I P S I

IMPLEMENTASI IP VERSI 6
PADA SISTEM OPERASI LINUX
PENGEMBANGAN TOOLS IPv6 BERBASIS SHELL SCRIPT

Disusun Oleh :
Antonius Arief Widiyatmoko
005314003

Telah disetujui oleh :

Pembimbing I



(Drs. Harris Sriwindono, M.Kom.) tanggal : 25 Mei 2005

Pembimbing II

(H. Agung Hernawan, S.T.) tanggal : 27 Mei 2005

HALAMAN PENGESAHAN

SKRIPSI

IMPLEMENTASI IP VERSI 6
PADA SISTEM OPERASI LINUX

PENGEMBANGAN TOOLS IPv6 BERBASIS SHELL SCRIPT

Dipersiapkan dan ditulis oleh :

Antonius Arief Widiyatmoko
005314003

Telah dipertahankan di depan Panitia Penguji
pada tanggal 16 Mei 2005
dan dinyatakan memenuhi syarat

Susunan Panitia Penguji

	Nama lengkap	Tanda tangan
Ketua	Drs. Harris Sriwindono, M.Kom	.....
Sekretaris	H. Agung Hernawan, S.T.	.....
Anggota 1	Alb. Agung Hadhiatma, S.T., M.T.	.....
Anggota 2	JB. Budi Darmawan, S.T., M.Sc.	.....

Yogyakarta, 23 Juni 2005
Fakultas Teknik
Universitas Sanata Dharma
Dekan,

(Ir. Greg. Heliarko S.J., S.S., B.S.T., M.A., M.Sc.)

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka, sebagaimana layaknya karya ilmiah.

Yogyakarta, Juni 2005

Penulis

Antonius Arief Widiyatmoko

Halaman Persembahan

Kupersembahkan karyaku ini teruntuk orang-orang yang kucintai:

Bapak dan Ibu yang telah membesarkanku dengan penuh kasih sayang
semoga karya sederhana ini dapat menjadi salah satu bagian dari
wujud tanggung jawab saya dalam menempuh pendidikan tinggi
di Universitas Sanata Dharma Yogyakarta

Kakak dan Adikku terima kasih atas dukungan dan doa yang diberikan selama ini

Halaman Motto

Jangan merasa bisa tapi hendaklah bisa merasakan

(ojo rumongso iso, ning isoo rumongso)

Berpikirlah selangkah didepan pikiran orang lain

ABSTRAKSI

Protokol jaringan internet yang digunakan dunia sekarang ini adalah protokol IP versi 4. IPv4 ini memiliki kombinasi alamat sebanyak 2^{32} atau sebesar 4294967296 alamat. Dengan semakin bertambahnya jumlah pengguna IP ini, resource yang disediakan untuk alamat IP publik ini semakin sedikit.

Oleh karena itu riset mengembangkan suatu versi protokol IP yang baru yaitu IP versi 6. IPv6 ini menyediakan kombinasi sebanyak 2^{128} alamat atau sebesar 340282366920938463463374607431768211456 alamat. Sehingga dengan resource alamat sebesar ini diharapkan komunikasi komputer di seluruh dunia tidak akan kehabisan alamat.

Namun sekarang ini komunikasi komputer di dunia masih menggunakan IPv4 dan masih jarang yang menggunakan IPv6. Maka penulis mencoba untuk mengiplemenstasikan suatu jaringan lokal dengan menggunakan IPv6 yang akan berkomunikasi dengan jaringan IPv6 lainnya dengan memanfaatkan suatu mekanisme tunneling.

ABSTRACT

Network protocol of internet used by the world at this time is protocol of IP version 4. This IPv4 have 2^{32} address combination or equal to 4294967296 address. Increasing consumer of this IP version, make reserved resource of this public IP address decreasing.

Therefore research to develop a new protocol version of IP that called IP version 6. This IPv6 provide 2^{128} address combination or equal to 340282366920938463463374607431768211456 address. So that with this large of range IP address resource, expected computer communications in all the world will not run out of address.

But at this time computer communications in this world still use IPv4 and still seldom using IPv6. Hence writer tries to make an implementation to communicate a local network by using IPv6 with other network of IPv6 by exploiting a mechanism of tunneling

KATA PENGANTAR

Puji syukur penulis haturkan kepada Tuhan Yang Maha Esa atas segala karunia yang diberikan, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "IMPLEMENTASI IP VERSI 6 PADA SISTEM OPERASI LINUX : PENGEMBANGAN TOOLS IPV6 BERBASIS SHELL SCRIPT" ini dengan baik. Penulisan ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik di Universitas Sanata Dharma pada program studi Teknik Informatika.

Selama penulisan skripsi ini penulis telah memperoleh bantuan dan bimbingan dari berbagai pihak. Oleh karena itu penulis mengucapkan terima kasih kepada:

1. Ibu A.M. Polina, selaku Ketua Jurusan Tenik Informatika Universitas Sanata Dharma.
2. Bapak Harris Sriwindono selaku pembimbing I yang telah banyak membantu dan membimbing selama mengerjakan tugas akhir ini.
3. Bapak H. Agung Hernawan selaku pembimbing II yang telah memberi banyak masukan dan bimbingannya.
4. Bapak, Ibu, Kakak dan Adikku yang telah memberi dorongan baik moril maupun materi.
5. Eka "Oni" Oktaviani yang memberi semangat dan support yang tiada habisnya.
6. Bapak Donny yang mengenalkan dunia Linux.

7. Bapak Belle dan Pak Dar yang mau menemani dan memberi semangat terus. Thanks Pak.
8. Teman-teman TI : Fito "Acong" Tatontos, Wawan "Suna" IGWB, Warih "Dono" Wibowo untuk ngelab di jarkom bareng, Antonius "Paijo" Andika, NG "Bobo" Wisnu, Johanes "Neng" Gunawan, Ari, Toro, Lukas, cepetan Lulus, Bram99, thanks ilmu web dan "lain"nya, Rosa "Mini", Wawan "Gundul", teman-temannya Bram, jangan nggosip terus, teman-teman lainnya, thanks a lot guys.
9. Teman-teman SMU: Sigit, Dyta, Amie, Jarwo, Tanto, dll, for supportnya.
10. Tanya-Jawab@linux.or.id untuk milisnya.
11. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah memberikan dukungan serta bantuannya guna penyusunan karya tulis ini

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih jauh dari kesempurnaan dan masih banyak kekurangan. Oleh karena itu penulis sangat mengharapkan kritik dan saran yang bersifat membangun demi perbaikan lebih lanjut. Penulis berharap semoga Tugas akhir ini dapat bermanfaat dan berguna bagi pembaca.

Yogyakarta, Juni 2005

Penulis

DAFTAR ISI

BAB IV IMPLEMENTASI PROGRAM

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kebutuhan komunikasi yang telah menjadi suatu kegiatan pokok akan semakin menuntut orang untuk menyediakan sarana dan prasarana dalam menunjang kegiatan komunikasi. Terlebih pada era informasi sekarang ini, semua kegiatan dapat berkembang pesat dengan bantuan informasi yang cepat dan akurat. Keterbatasan jarak dalam komunikasi menuntut orang untuk menciptakan sarana untuk mempermudah komunikasi.

Perkembangan teknologi yang pesat terutama dalam bidang elektronika dan komputer memunculkan suatu gagasan komunikasi yang dapat dilakukan oleh dua atau lebih perangkat komputer. Agar dua atau lebih perangkat komputer dapat berkomunikasi maka dibutuhkan suatu protokol untuk menjembatani komunikasi antar komputer. Protokol yang sekarang terkenal adalah protokol TCP/IP (*Transmition Control Protocol / Internet Protocol*). Pengalamatan yang dipakai sekarang ini adalah pengalamatan IP dengan menggunakan protokol IP versi 4 (IPv4).

Pengalamatan dengan IPv4 memberikan sejumlah alokasi alamat sebanyak 2^{32} atau 4294967296 alamat. Dari sekian banyak alokasi alamat yang tersedia, alamat-alamat tersebut didistribusikan ke seluruh dunia untuk keperluan komunikasi komputer seperti Internet. Namun dengan semakin bertambahnya jumlah pengguna layanan internet, maka dapat dipastikan bahwa alokasi alamat

yang tersedia akan habis. Untuk mengatasi keterbatasan alamat yang ada, dapat dilakukan dengan mengalokasikan kembali alamat-alamat yang sudah tidak terpakai. Namun dengan mengalokasikan kembali alamat yang sudah tidak terpakai, tidak menutup kemungkinan bahwa alamat yang disediakan pada akhirnya akan habis juga.

Oleh karena itu dikembangkan suatu pengalamatan IP model baru untuk menggantikan pengalamatan IP versi 4. Pengalamatan ini disebut sebagai pengalamatan IP versi 6 atau IPv6 atau sering juga disebut IPng (*Internet Protocol new generation*). Alokasi alamat yang disediakan oleh IPv6 sebesar 2^{128} atau 340282366920938463463374607431768211456 alamat. Dengan sumber daya alamat yang sebanyak ini diharapkan dapat mengganti atau setidaknya menutupi kekurangan sumber daya alamat yang ada sekarang ini.

Untuk mengganti langsung jaringan IPv4 dengan IPv6 akan memerlukan dana yang cukup besar karena harus mengganti infrastruktur yang mendukung IPv6. Oleh karena itu digunakan mekanisme transisi yang dapat menjembatani pergantian IPv4 dengan IPv6. Salah satu mekanisme yang digunakan dalam pergantian tersebut adalah mekanisme *tunneling*. Yaitu mekanisme yang menghubungkan dua buah jaringan IPv6 melalui jaringan IPv4.

Karena IPv6 adalah protokol baru yang digunakan untuk menggantikan pengalamatan IPv4 maka *adminitrator* harus mengetahui bagaimana cara mengaktifkan IPv6 untuk tiap-tiap komputer. *Administrator* juga harus mengetahui langkah-langkah dalam mensetting IPv6. Dalam mensetting IPv6 ada

beberapa langkah yang cukup sulit jika harus mengisikan kembali konfigurasi IPv6 jika komputer mengalami *reboot* atau network mengalami *restart*.

Dengan mempertimbangkan kesulitan konfigurasi dalam mensetting IPv6, maka penyusun mencoba untuk membuat program / tools yang dapat mempermudah *administrator* dalam mensetting IPv6 ke dalam komputer *client* dan komputer yang difungsikan sebagai *router*. Tools tersebut memanfaatkan file-file aplikasi yang dimiliki oleh Linux untuk setting IPv6 sehingga komputer dapat mengirimkan paket IPv6 dengan menggunakan mekanisme *tunneling* melalui jaringan IPv4.

1.2. Batasan Masalah

Dalam batasan masalah ini, penyusun membatasi permasalahan yang perlu, yaitu :

1. Sistem Operasi yang dipakai adalah Linux berbasis Red Hat dengan kernel minimal 2.4.
2. Mekanisme transisi yang digunakan adalah mekanisme *Tunneling*.
3. Aplikasi yang digunakan sebagai penguji koneksi *tunneling* adalah ping6, http dan ssh.
4. Instalasi, trouble shooting, dan keamanan Sistem Operasi tidak dibahas dalam tulisan ini.
5. Mode Linux yang dipakai adalah mode text.
6. Instalasi Sistem Operasi *router* adalah paket standar server.

1.3. Rumusan Masalah

Dalam mempersiapkan IPv6 sebagai protokol pengganti IPv4 yang diperlukan bukan hanya sekedar mengganti protokol IP saja. Permasalahan yang dihadapi adalah :

1. Bagaimana cara mengaktifkan IPv6?
2. Bagaimana cara membangun jaringan IPv6 diatas jaringan IPv4?
3. Bagaimana membuat suatu program untuk membantu *administrator* dalam mensetting IPv6?

1.4. Tujuan Penulisan Tugas Akhir

Adapun tujuan dari implementasi *tunneling* IPv6 melalui IPv4 ini adalah penulisan tugas akhir ini adalah :

1. Memahami bentuk protokol IPv6.
2. Dapat membangun jaringan IPv6 dalam lingkup lokal.
3. Membentuk koneksi antar jaringan IPv6 melalui jaringan IPv4 (*tunneling*) dengan model *point-to-point*.
4. Membuat tools yang dapat menyimpan dan menjalankan setting IPv6 secara otomatis.

1.5. Metodologi Penelitian

Beberapa tahapan yang dilakukan dalam menyelesaikan masalah adalah:

1. Studi pustaka.

   Penyusun mencari literatur-literatur IPv6 untuk mengenal dan mempelajari sistematika, arsitektur, dan struktur IPv6. Selain itu, penyusun juga mencari literatur tentang Sistem Operasi Linux yang akan digunakan sebagai suatu Sistem Operasi dalam membangun jaringan IPv6.

2. Mengikuti forum Mailing List.

   Selain dengan studi pustaka, penyusun juga mengikuti forum mailing list. Dengan mengikuti forum mailing list, penyusun dapat bertanya ke kepada forum jika terjadi kesulitan.

3. Perancangan sistem.

   Langkah selanjutnya adalah perancangan. Perancangan sistem yang dibuat adalah perancangan secara umum, perangkat keras dan perangkat lunak yang akan dipakai, perancangan implementasi jaringan dan perancangan program yang akan dibuat.

4. Uji coba tunneling.

   Selanjutnya setelah perancangan adalah uji coba *tunneling* dalam jaringan. Langkah ini digunakan untuk mengetahui kebutuhan dalam pembuatan tools.

5. Membuat tools.

   Dengan data yang didapat dari uji coba *tunneling*, langkah selanjutnya adalah membuat tools. Bahasa pemrograman yang dipakai adalah bahasa pemrograman bash.

6. Implementasi program dalam jaringan tunneling.

   Pada tahap ini dilakukan uji coba program pada jaringan untuk membuktikan bahwa program sudah dapat membangun jaringan IPv6 dan melewatkan paket dalam jaringan IPv4 melalui mekanisme *tunneling*.

1.6. Sistematika Penulisan

Adapun sistematika penulisan tugas akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang penulisan tugas akhir, batasan masalah yang diambil, rumusan masalah, tujuan penulisan dan metodologi penelitian tugas akhir.

BAB II LANDASAN TEORI

Bab ini membahas tentang konsep dasar jaringan, protokol yang dipakai dan arsitektur IPv6, serta penggunaan IPv6 dalam masa trasisi.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisa dasar teori terhadap permasalahan yang ada serta penjelasan perancangan desain dan langkah-langkah dari program

yang akan dibuat. Perancangan meliputi perancangan program dan opsi yang dimiliki oleh program.

BAB IV IMPLEMENTASI PROGRAM

Bab ini membahas tentang implementasi perancangan program kedalam bahasa perograman. Dan mengimplementasikan program ke dalam komputer untuk setting IPv6 dan penanganan terhadap *tunneling*.

BAB V ANALISIS HASIL IMPLEMENTASI

Bab ini berisi tentang hasil implementasi, kelebihan dan kekurangan mekanisme *tunneling* dan program.

BAB VI PENUTUP

Berisi tentang kesimpulan dari pembahasan dan implementasi yang telah dilakukan dalam penulisan tugas akhir ini serta saran-saran untuk pengembangan program selanjutnya.

BAB II

LANDASAN TEORI

2.1. Jaringan Komputer

Jaringan komputer adalah suatu media transmisi bersama serta rangkaian *hardware* dan *software* untuk menginterfacekan perangkat menjadi media serta mengatur akses menuju media tersebut dengan tepat (William Stallings, 2000).

Jaringan komputer dapat diimplemanetasikan dalam berbagai lingkup, baik besar maupun kecil. Dalam sebuah lembaga kecil biasanya diterapkan sebuah jaringan lokal atau sering disebut *Local Area Network* (LAN) (Wahana, 2003).

LAN adalah suatu jaringan internal yang terbatas meliputi satu lingkup lokal area. Jarak yang dijangkau oleh jaringan LAN harus kurang dari 1000 meter. LAN banyak digunakan untuk menghubungkan antara bagian dalam satu gedung atau sedikit lebih luas digunakan untuk menghubungkan antar departemen dalam satu perusahaan.

Untuk memahami cara kerja suatu jaringan komputer, maka harus terlebih dahulu memahami konsep referensi model jaringan komputer. Referensi model jaringan ini digunakan untuk mempermudah cara mempelajari, memprogram, maupun mendesain suatu jaringan komputer (Hendra Wijaya, 2002).

Keuntungan yang didapat dari penggunaan jaringan kumputer adalah sebagai berikut :

1. Dapat saling berbagi (*sharing*) baik *sharing* sumber daya (*harddisk, printer*) maupun *sharing* file atau data.

2. Penggunaan aplikasi secara bersama-sama (*multiuser*).
3. Penyampaian informasi yang lebih cepat melalui *e-mail* dan sebagainya.
4. Membuat salinan (*backup*) data ke tempat lain dengan lebih mudah.
5. Pembatasan hak ases kepada user tertentu.

Komunikasi data antar komputer pada suatu jaringan memerlukan proses yang panjang dan dibagi dalam beberapa tahapan. Proses ini oleh referensi model jaringan dapat dibagi menjadi beberapa lapisan (*layer*) untuk menjelaskan cara kerja suatu jaringan komputer. Salah satu referensi model jaringan yang terkenal adalah OSI (*Open System Interconnection*), yang diperkenalkan oleh ISO (International Standard Orgaization)[1]. Referensi model jaringan menurut OSI membagi jaringan komputer menjadi tujuh lapisan.

7	Aplication
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

Gambar 2.1 Referensi model jaringan menurut OSI

Dalam setiap lapisan pada model jaringan OSI memiliki fungsi dan protokol yang digunakan untuk berkomunikasi dengan lapisan yang sederajat pada

[1] http://en.wikipedia.org/wiki

komputer lain. Pada referensi model OSI, setiap lapisan dibuat agar hanya mengatur layanan dan protokol yang bekerja pada lapisan tersebut sehingga seolah-olah setiap lapisan berdiri sendiri. Tujuannya adalah untuk mempermudah pembuatan program untuk jaringan.

7	Aplication	HTTP, FTP, SMTP, Telnet, Ssh and Scp
6	Presentation	XML, XDR, SMB, AFP
5	Session	TLS, SSH, RPC, NET BIOS, ASP
4	Transport	TCP, UDP, SCTP, ATP
3	Network	IP, ICMP, IGMP, ARP, RARP, X.25
2	Data Link	Ethernet, Token ring, ATM, Frame Relay
1	Physical	Electricity, Radio, Laser

Gambar 2.2 Pembagian Protokol menurut OSI

Dengan melihat diagram diatas, pada setiap level menunjukkan fungsi yang dilakukan oleh protokol. Namun dalam kenyataannya, tiga lapisan teratas dalam model OSI (*Aplication, Presentation,* dan *Session*) biasa diperlakukan menjadi satu lapisan tunggal dalam deretan TCP/IP yaitu sebagai lapisan *Aplication* yang protokolnya dianggap menjadi satu kesatuan protokol aplikasi.

5	Aplication	HTTP, FTP, SMTP, Telnet, Ssh and Scp
4	Transport	TCP, UDP, SCTP
3	Network	IP, ICMP, IGMP
2	Data Link	Ethernet, Token ring, ATM, Frame Relay
1	Physical	Electricity, Radio, Laser

Gambar 2.3 Pembagian Protokol menurut TCP/IP

Diagram lapisan OSI maupun TCP/IP memiliki persamaan pada keempat layer dasar (*Physical, Data Link, Network* dan *Transport*). Dari level *Network* dan *Transport* dapat diketahui bahwa protokol IP yang terletak di layer *Network* dan TCP yang terletak di layer *Transport* digunakan sebagai protokol yang berfungsi untuk pengalamatan antar komputer dan pengendali transmisi. Layer *physical* dan *data link* dari susunan TCP/IP sering disebut sebagai satu layer yaitu *network interface layer*.

2.2. Routing

Protokol *routing* hanya digunakan oleh alat yang bertindak sebagai *router*. Salah satu fungsi *router* adalah menentukan jalur yang akan digunakan untuk melewatkan paket dari satu jaringan ke jaringan lain. Mekanisme pengambilan keputusan tentang jalur yang akan digunakan untuk mengirim paket dikelola oleh protokol *routing* (Zaenal Arifin, 2003).

Gambar 2.4 Jaringan yang dipisahkan oleh router

Router memberikan rekomendasi tentang jalur yang akan digunakan untuk melewatkan paket berdasarkan informasi yang terdapat dalam tabel *routing*. Tabel *routing* pada umumnya memiliki informasi :

a. Alamat network tujuan.
b. *Interface router* lokal yang terdekat dengan network tujuan.
c. Metric, yaitu sebuah nilai yang menunjukkan jarak untuk mencapai network tujuan.

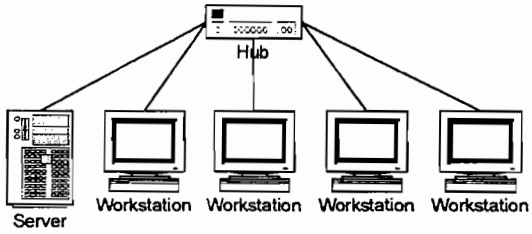
2.3. LAN (Local Area Network) dan WAN (Wide Area Network)

LAN digunakan dalam teknologi jaringan yang memiliki jarak yang pendek. Jaringan LAN banyak menggunakan *hub* dan *repeater* untuk menggabungkan peralatan komputer satu dengan yang lain dengan bekerja di lapisan *phisycal* dan menggunakan protokol Ethernet.

2.3.1. Repeater dan Hub

Penggunaan *hub* dan *repeater* sangat mudah karena tidak memerlukan konfigurasi yang rumit dan harga yang murah. Namun dari kelebihan tersebut, *hub* dan *repeater* memiliki kelemahan karena alat ini hanya meneruskan sinyal ke semua komputer tanpa mengenal alamat yang dituju. Hal ini dikarenakan protokol Ethernet atau IEEE 802.3 menggunakan mekanisme CSMA/CD (*Carrier Sense*

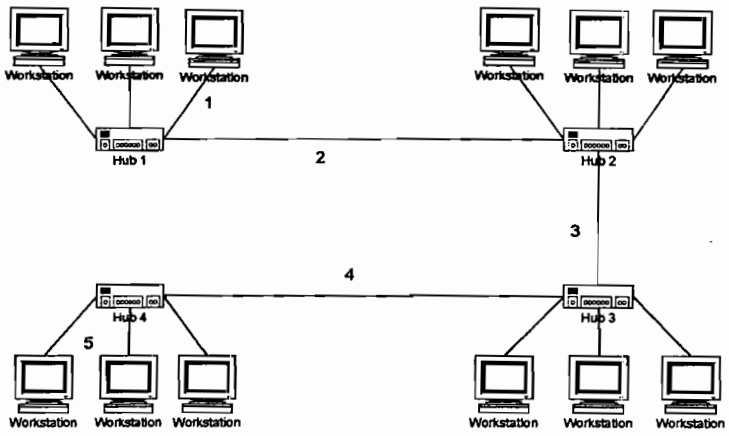
Multiple Access / Collision Domain) yaitu suatu cara dimana peralatan memeriksa dahulu jaringan apakah ada pengiriman data oleh pihak lain. Jika tidak ada pengiriman data oleh pihak lain, maka pengiriman data dapat dilakukan. Jika ada dua atau lebih peralatan mengirimkan paket secara bersama-sama, maka akan terjadi tabrakan (*collision*).



Gambar 2.5 Jaringan LAN yang menggunakan hub

Karena *hub* adalah peralatan yang memiliki satu *collision domain*, maka semua peralatan yang terhubung dengan *hub* menggunakan satu *collision domain* secara bersama walaupun dihubungkan dengan port yang berlainan dari *hub*. Dengan adanya kekurangan pada *collision domain*, maka *hub* dan *repeater* memiliki pembatasan-pembatasan sebagai berikut :

a. Antara dua komputer hanya diperbolehkan empat buah *hub* dan lima segmen kabel.
b. Panjang kabel antara komputer ke *hub* atau *hub* ke *hub* maksimum 100 meter.
c. Diameter jaringan yaitu panjang kabel maksimum antara dua komputer misalnya antara komputer A dan komputer B adalah 500 meter.
d. Panjang kabel minimum antar peralatan adalah 1 meter.



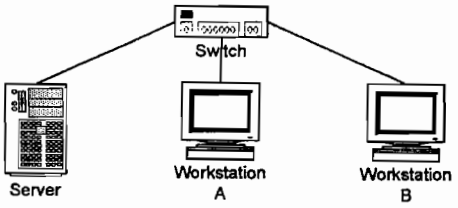
Gambar 2.6 Jaringan LAN dengan empat buah hub

2.3.2. Bridge dan Switch

Bridge dan *switch* bekerja di lapisan data link dan menggunakan MAC address untuk meneruskan paket-paket data ke tujuan. *Bridge* juga secara otomatis membuat tabel penterjemah untuk paket yang diterima pada masing-masing port. Dengan adanya *bridge* ini kepadatan lalu-lintas jaringan dapat dikurangi, karena *bridge* hanya menyiarkan (*broadcast*) paket-paket yang tidak dikenali oleh tabel penterjemah. Jadi *bridge* sudah dapat membagi jaringan menjadi dua *collision domain*. Metode ini sering disebut sebagai segmentasi. Hanya saja jika alamat yang diterima oleh *bridge* tidak dikenali, maka *bridge* akan mem-*broadcast* berita ke semua segmen yang ada, sehingga hal ini dapat menimbulkan *broadcast storm* (badai siaran) yang dapat menyebabkan kemacetan total pada jaringan.

Seperti halnya *bridge*, *switch* juga bekerja pada lapisan data link. Dan seperti sifat *bridge*, *switch* memiliki keunggulan membagi *collision domain* di tiap port yang dimilikinya. *Switch* mempunyai penterjemah pusat yang memiliki daftar penterjemah alamat untuk masing-masing port. Oleh karena itu, jika ada peralatan

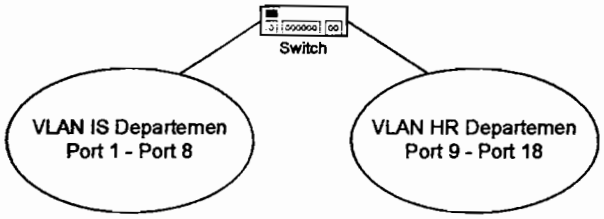
yang berkomunikasi dengan *switch*, alat tersebut akan berkomunikasi dengan menggunakan VPN (*Virtual Private Network*) antar port yang menghubungkan ke dua alat tersebut ke *switch*, sehingga tidak mengganggu segmen lainnya.



Gambar 2.7 Jaringan dengan menggunakan Switch

2.3.3. VLAN

Switch dapat dikonfigurasikan menjadi *Virtual LAN* (VLAN) yang bekerja mirip seperti konsep subnetting dari TCP/IP dimana network dapat disegmentasi secara virtual tanpa harus menuruti lokasi fisik peralatan (Hendra Wijaya, 2002).



Gambar 2.8 Jaringan VLAN

Setiap port dari *switch* dapat diterapkan menjadi milik suatu VLAN. Oleh karena berada dalam satu segmen, port-port yang bernaung dibawah satu VLAN dapat saling berkomunikasi langsung. Sedangkan port-port yang berada diluar VLAN atau bernaung di bawah VLAN lain, tidak dapat saling berkomunikasi karena VLAN tidak meneruskan *broadcast*. Komunikasi VLAN dengan VLAN

lain harus melalui *router*, demikian pula untuk komunikasi antara LAN dengan LAN yang terpisah lainnya harus menggunakan *router* agar dapat berkomunikasi.

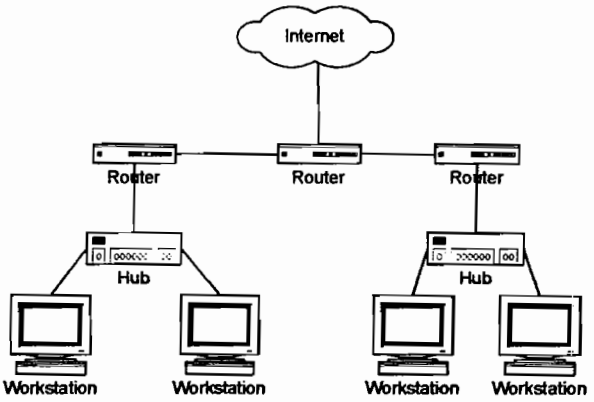
2.3.4. Router

Protokol *routing* diperlukan dalam jaringan karena ibarat jalan raya yang dipakai untuk kendaraan umum, *router* adalah jalan alternatif atau jalan tambahan yang dapat dipakai untuk mengantisipasi kemacetan jalan raya utama untuk sampai ke tujuan. Untuk melewati jalan alternatif tersebut dibutuhkan peta jalan agar tidak salah mengambil jalur jalan dan tidak tersesat.

Jaringan WAN memiliki berbagai segmen dan jaringan dengan jalur yang berbagai macam. Supaya paket yang dikirimkan sampai ke tujuan dan tidak hilang di tengah jalan, maka diperlukan suatu peralatan untuk mengatur paket-paket agar sampai ke tujuan dengan jalur yang tersingkat. Untuk itu digunakan *router* untuk menentukan jalur dan meneruskan paket-paket dari suatu jaringan ke jaringan lain. Untuk menentukan jalur terbaik, *router* menggunakan peta jaringan atau tabel *routing*. Tabel *routing* dapat dibuat secara *static*, *dynamic*, dan *default* (Hendra Wijaya, 2002).

Gambar 2.9 Arsitektur router

Routing bekerja pada lapisan internet yang berarti melewatkan paket IP ke alamat tujuan (Muhamad Syukri, 2003). *Router* inilah yang melakukan kontrol pada jaringan. *Routing* bekerja dengan cara *hop-by-hop*, artinya IP tidak mengetahui secara penuh jalur menuju tujuan setiap paket.



Gambar 2.10 Jaringan yang terhubung dengan router

Pengisian *routing table* dapat dilakukan dengan cara *static*, *dynamic*, dan *default*. Cara *static* adalah pengalamatan dengan membuat tabel *routing* secara manual. *Administrator* jaringan harus menginputkan sendiri tabel routing yang diperlukan oleh *router*. Cara ini dapat dipakai untuk jaringan sederhana yang menggunakan beberapa buah *router* dan berfungsi untuk menghemat penggunaan *bandwidth*. Sedangkan cara *dynamic* memungkinkan *router* untuk membuat tabel *routing* secara dinamis berubah-ubah secara otomatis berdasarkan informasi dari *router* lain di jaringan mengikuti topologi jaringan yang berubah.

2.4. TCP/IP (Transfer Control Protocol/Internet Protokol)

TCP/IP merupakan proyek yang dikembangkan oleh *Departement of Defence* (*DoD*) *Defense Advance Research Projects Agency* (*DARPA*) untuk menghubungkan antar jaringan (*network*) yang dikembangkan oleh *vendor* yang berbeda menjadi suatu jaringan dalam jaringan luas (*Network of Networks*) atau sekarang terkenal dengan nama Internet[2]. Rancangan ini sukses berkembang

[2] http://www.yale.edu

karena dapat memberikan layanan dasar yang dibutuhkan oleh orang-orang (*file sharing, electronic mail*) karena beberapa komputer dalam suatu departemen kecil dapat menggunakan TCP/IP (berjalan bersama dengan protokol lainnya) dalam satu jaringan LAN..

Seperti halnya protokol komunikasi lainnya, TCP/IP juga dibagi menjadi beberapa lapisan :

a. IP

IP bertanggung jawab untuk memindahkan paket data dari satu *node* ke *node* lainnya. IP meneruskan paket data berdasarkan empat byte alamat tujuan (*IP number / IP address*). Otoritas *Internet* membagi alamat IP menjadi beberapa organisasi yang berbeda. Setiap organisasi akan mengkelompokan alamat itu menjadi beberapa departemen. IP bekerja sebagai pintu gerbang dan alamat mesin dalam memindahkan paket data dari satu departemen ke satu organisasi dan diteruskan ke suatu wilayah regional dan kemudian seluruh dunia.

b. TCP

TCP bertanggung jawab untuk pengecekan pengiriman paket data dari *client* ke *server*. Dalam pengirimannya, data dapat hilang ditengah jalan dalam jaringan. Oleh karena itu, TCP menambahkan suatu paket tambahan untuk mengecek kesalahan data atau kehilangan data dan memerintahkan untuk mengirim ulang data sampai data diterima dalam keadaan benar dan lengkap.

2.4.1. IP (Internet Protocol)

Jaringan internet merupakan kumpulan dari beberapa komputer (*host*) yang tersambung melalui ratusan ribu jaringan komputer yang ada di seluruh dunia (Hendra Wijaya, 2002). Semua *host* yang terhubung ke jaringan internet dapat dikenali karena komputer tersebut mempunyai alamat IP. Alokasi alamat IP yang tersambung ke jaringan internet harus dilakukan dengan baik agar *routing* dapat berjalan dengan baik. Konsekuensi dari penerapan jaringan internet adalah mengharuskan seluruh komputer yang terhubung dengan internet menggunakan protokol TCP/IP untuk berkomunikasi dan menggunakan IP address untuk pengalamatan pada tiap-tiap komputer (*host*).

IP terletak di layer ketiga dalam referensi model OSI yaitu layer *network*. IP adalah sebuah alamat logika yang digunakan untuk memberikan alamat kepada komputer yang terhubung ke jaringan yang menggunakan protokol TCP/IP. Alamat logika ini sering disebut sebagai alamat IP atau IP Address. Setiap komputer yang terhubung dengan internet harus mempunyai alamat IP yang unik dan tidak boleh sama. IP address ditentukan oleh *subnetmask* yang berfungsi untuk membedakan bagian yang disebut *network* dan bagian yang disebut *host*.

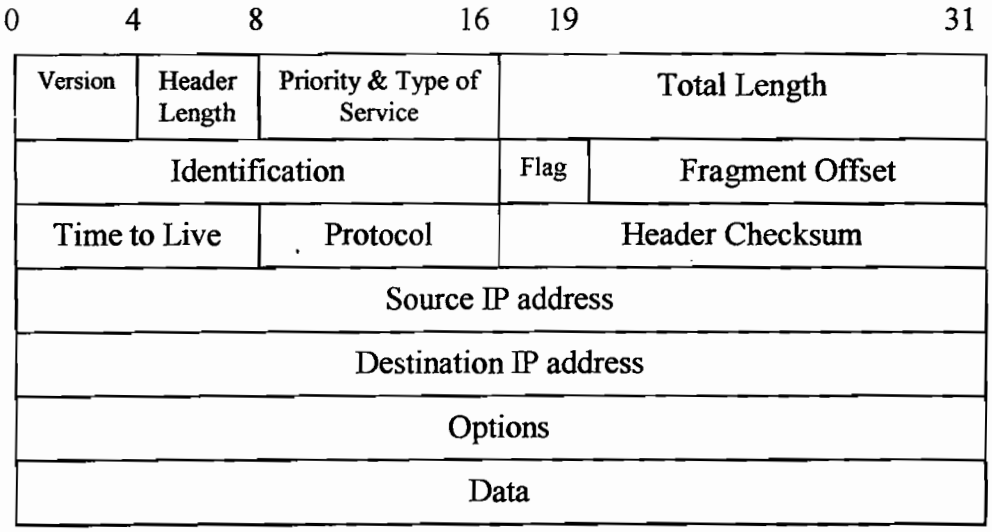
2.4.1.1. IPv4 (Internet Protocol version 4)

IP versi 4 adalah pengalamatan dengan menggunakan bilangan binner sepanjang 32 bit yang terbagi atas 4 segmen. Tiap segmen terdiri atas 8 bit yang dipresentasikan dalam bentuk bilangan desimal dari 0 sampai 255. IP versi 4 memiliki range address yang bisa digunakan dari 0.0.0.0 sampai 255.255.255.255.

Seluruh data dari layer diatasnya harus melewati IPv4 ini agar dapat dipaketkan menurut alamat pengirim dan tujuan dan dikirimkan sebagai paket IPv4 agar dapat samapi ke alamat tujuan.

Dalam mengirimkan data, IPv4 memiliki sifat-sifat yang mendukung pengiriman paket IPv4 yaitu, *connectionless*, *unreliable*, dan *datagram delivery service*.

a. *Connectionless* : IPv4 tidak melakukan perjanjian (*handshake*) untuk melakukan komunikasi.
b. *Unreliable* : IPv4 menganggap bahwa data yang dikirim pasti sampai di tujuan.
c. *Datagram delivery service* : Paket data IPv4 adalah independen terhadap paket data yang lain.



Gambar 2.11 Datagram IPv4 address

Format datagram IPv4 adalah :

a. *Version* : Versi yang menunjukkan versi IP. Format datagram ini adalah format datagram IP versi 4.
b. *Header Length* : Berisi pangjang *header* paket IPv4.
c. *Priority & Type of Service* : Berisi kualitas service yang dapat mempengaruhi cara penanganan paket IPv4.
d. *Total Length* : Panjang total datagram IPv4 dalam hitungan byte.
e. *Identification*, *Flags*, dan *Fragment offset* : Berisi tentang pemecahan dan penggabungan kembali data-data yang dikirim atau diterima.
f. *Time to Live* : Membatasi umur paket untuk sampai ke tujuan. Berisi jumlah hop maksimal paket IPv4 dalam melewati *router*.
g. *Protocol* : Mengandung angka yang mengidentifikasikan protokol layer atas pengguna dan isi paket IPv4.
h. *Header Checksum* : Berisi nilai *checksum* yang dihitung dan seluruh *field* dan *header* paket IPv4.
i. *Source IP address* : Berisi alamat IPv4 pengirim paket data.
j. *Destination IP address* : Berisi alamat IPv4 tujuan yang dikirimi paket data.
k. *Options* : Berisi *Strict Source Route* (SSR) dan *Loose Source Route* (LSR). SSR berisi daftar lengkap *IPv4 address* dan *Router* yang harus dilalui. Paket yang dikirim diharuskan singgah ke beberapa *router* yang ditunjuk.
l. *Data* : Berisi data yang dikirimkan melalui paket IPv4.

Format IPv4 dapat dinyatakan dalam dua bentuk yaitu bentuk desimal dan bentuk biner.

a. Bentuk biner

Format IPv4 dalam bentuk biner adalah alamat IPv4 yang dituliskan dalam bentuk bilangan biner dan ditulis dalam tiap 8 bit. Setiap 8 bit, alamat dipisahkan oleh sebuah tanda titik.

xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx

Simbol "x" adalah simbol untuk bilangan biner.

b. Bentuk desimal

Bentuk desimal ini juga sering disebut sebagai *dotted decimal* (desimal bertitik). Setiap bilangan desimal adalah satu *oktet* (8 bit) alamat IPv4. Bentuk pengalamatan IPv4 secara desimal adalah sebagai berikut :

Bit 0 32

11000000.00010000.10100000.00100100

192 . 16 . 160 . 36

penulisan alamat IPv4 ini adalah 192.16.160.36

Cara sederhana untuk menghitung nilai desimal tiap segmen dari bentuk biner adalah dengan cara :

Biner	1	1	1	1	1	1	1	1
Desimal	128	64	32	16	8	4	2	1

Tabel 2.1 IP address dalam bilangan Desimal dan Biner

Contoh mencari nilai desimal dalam satu segmen IPv4.

Biner	1	1	0	0	0	0	0	0
Desimal	128	64	32	16	8	4	2	1

Tabel 2.2 Contoh penghitungan desimal IPv4

Hasil desimal dalam satu segmen didapat dari hasil penjumlahan dari nilai desimal yang nilai binernya bernilai 1, sehingga contoh diatas didapat nilai desimal sebesar 128+64=192.

Sebuah IPv4 Address yang terdiri dari 32 bit mewakili *network* dan *host/node* address (Zaenal Arifin, 2003).

a. *Network address* secara unik menandai kelompok setiap jaringan. Setiap mesin pada jaringan yang sama menggunakan *network address* yang sama.
b. *Node address* dipasang secara unik terhadap setiap mesin yang terdapat dalam sebuah jaringan. Artinya, dalam sebuah jaringan tidak boleh terdapat dua atau lebih mesin yang menggunakan alamat yang sama.

Banyaknya bit yang digunakan oleh network dan host address diatur oleh nilai *subnet mask*. *Subnet mask* terbentuk dari bilangan 0 dan 1. Bit 1 mewakili bagian dari *network address* dan bit 0 mewakili bagian dari *host address*.

192 . 168 . 0 . 1

255 . 255 . 255 . 0

Network address dalam pembagian alamat diatas adalah 192.168.0.0 dan *host address* yang dimiliki adalah 192.168.0.1 sampai dengan 192.168.0.254.

Secara administrative IP *address* terbagi dalam lima kelas. Kelas A, B dan C memiliki subnet mask default, yakni:

a. Default subnet mask kelas A : 255.0.0.0

b. Default subnet mask kelas B : 255.255.0.0

c. Default subnet mask kelas C : 255.255.255.0

d. Kelas D dan E tidak memiliki subnet mask.

Pembagian kelas terhadap pengalamatan IP ini dimaksudkan untuk membedakan jumlah alamat *host* pada suatu *network*.

Dalam penggunaannya alamat IPv4 terbagi ke dalam dua jenis, yakni:

a. Alamat IPv4 Publik.

Alamat IPv4 yang biasa digunakan pada jalur publik. Penggunaan alamat IPv4 publik harus melalui proses registrasi ke suatu organisasi yang menangani masalah penggunaan IPv4 agar tidak terdapat *host* yang memiliki alamat IPv4 yang sama.

b. Alamat IPv4 Privat.

Alamat IPv4 yang biasa digunakan pada jaringan lokal. Penggunaannya tidak memelukan proses registrasi. Alamat-alamat IPv4 yang tergolong ke dalam IPv4 privat di antaranya:

Kelas A → 10 . 1-255 . 1-255 . 1-255

Kelas B → 172 . 16-31 . 1-255 . 1-255

Kelas C → 192 . 168 . 1-255 . 1-255

2.4.1.2. IPv6 (Internet Protocol versi 6)

Pengalamatan IPv6 memiliki 2^{128} kombinasi alamat. Sama seperti halnya IPv4, IPv6 ini menggunakan bilangan biner yang memiliki panjang 128 bit dan terbagi menjadi 8 segmen, dengan tiap segmen terdiri dari 16 bit yang dipresentasikan dalam bilangan hexadesimal 0 sampai FFFF. Dengan kapasitas itu maka pengalamatan IPv6 adalah dari :

0000:0000:0000:0000:0000:0000:0000:0000

sampai

FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF

Terdapat tiga jenis pengalamatan dalam IPv6 yaitu :

a. *Unicast* : Alamat pengenal satu *interface*. Paket yang dikirim ke alamat *Unicast* akan disampaikan ke *interface* yang diidentifikasikan oleh alamat tersebut.
b. *Anycast* : Alamat pengenal kumpulan *interface*. Paket yang dikirim ke alamat ini akan disampaikan ke salah satu *interface* yang terdekat menurut *routing*.
c. *Multicast* : Alamat pengenal kumpulan *interface*. Paket yang dikirim ke alamat ini akan disampaikan ke semua *interface* dan menggantikan fungsi alamat *broadcast* dalam IPv4.

Interface pada mesin diasosiasikan dengan alamat IPv6 dan boleh memiliki beberapa alamat IPv6 dari berbagai jenis. *Subnet* prefix diasosiasikan dengan jaringan. Pengecualian pemodelan alamat adalah alamat *Anycast* yang

diperlakukan sebagai sebuah *interface* di lapisan Internet dan bermanfaat untuk pembagian trafik melalui beberapa *interface*[3].

Representasi alamat IPv6 adalah sebagai berikut:

a. Bentuk x:x:x:x:x:x:x:x, x adalah nilai hexadesimal 8 bagian 16 bit alamat. Penulisan alamat IPv6 adalah sebagai berikut :

   3ffe:419:2ac6:44ff:10ab:f54:a9:34fa

b. Penulisan alamat IPv6 dapat menggunakan tanda "::" untuk menyederhanakan penulisan. Tanda menunjukkan kumpulan 16 bit yang terdiri dari bit nol. Tanda ini hanya boleh digunakan sekali dalam satu buah alamat.

   10ab:0:0:0:0:0:8fe:4017 dapat di tulis *10ab::8fe:4017*

c. Penulisan alamat IPv6 juga dapat digunakan dalam lingkungan gabungan IPv4 dan IPv6. Alamat dalam lingkungan gabungan dapat ditulis dengan x:x:x:x:x:x:d.d.d.d, dengan x adalah 6 segmen 16 bit alamat IPv6 (heksadesimal) dan d adalah 4 segmen 8 bit IPv4 (desimal).

   0:0:0:0:0:0:167.205.22.116 atau dapat ditulis *::167.205.22.116*

d. Prefix alamat IPv6 dapat ditulis dalam notasi : Alamat-IPv6/Panjang-Prefix. Panjang prefix menunjukan banyaknya bit di bagian kiri yang dibatasi nilai prefixnya.

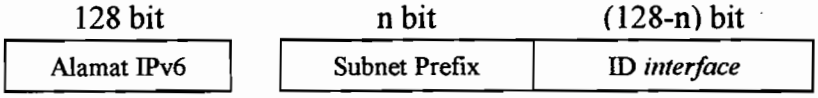
   12AB:0:0:CD30:123:4567:89AB:CDEF/60

a. Alamat Unicast

---

[3] RFC 2373

Jenis alamat *Unicast* IPv6 adalah alamat *Unicast* agregasi global, NASP, *Site-Local*, *Link-Local*, dan IPv6 dalam IPv4. Alamat IPv6 dapat memiliki struktur internal atau tidak memiliki struktur internal.



Gambar 2.12 Struktur Internal alamat Unicast

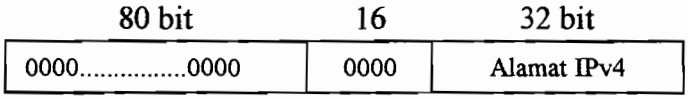
1. Pengenal alamat *Unicast* IPv6 mengidentifikasikan secara unik *interface* dalam hubungan komunikasi. Alamat ini umumnya sama dengan alamat lapisan *interface Data-Link*. Sebagai contoh :

   Alamat MAC = *00:30:84:9e:e4:25*

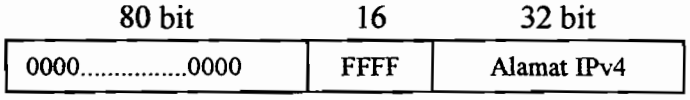
   Alamat tersebut dalam IPv6 akan menjadi = *fe80::0230:84ff:fe9e:e425*

2. Seperti halnya IPv4, alamat IPv6 ada yang tidak boleh digunakan sebagai alamat tujuan atau pada *header routing* yaitu alamat *0:0:0:0:0:0:0:0* karena alamat itu menandakan alamat tidak ada dan digunakan saat inisialisasi untuk mempelajari alamat.
3. Alamat *loopback* yang dimiliki oleh IPv6 adalah *0:0:0:0:0:0:0:1*. Alamat ini tidak boleh dijadikan alamat tujuan maupun alamat pengirim dan alamat ini diasosiasikan sebabagai *virtual interface*.
4. IPv6 mendukung pengalamatan yang memiliki IPv4 didalamnya. Terdapat dua jenis pengalamatan dengan IPv4 didalamnya yaitu alamat IPv6 "*IPv4-Compatible*" dan alamat IPv6 "*IPv4-mapped*". IPv6 "*IPv4-Compatible*" adalah alamat IPv6 yang membawa alamat IPv4 untuk mekanisme transisi.

Sedangkan IPv6 "*IPv4-mapped*" adalah alamat IPv6 yang hanya mendukung IPv4.

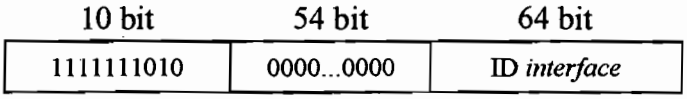


Gambar 2.13 Alamat IPv6 "IPv4-Compatible"

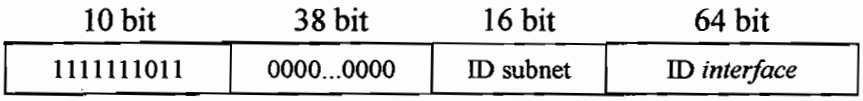


Gambar 2.14 Alamat IPv6 "IPv4-mapped"

5. Alamat *Unicast* IPv6 dapat digunakan dalam lingkungan lokal. Alamat untuk penggunaan lokal adalah *Link-Local* dan *Site-Local*. *Link-Local* adalah pengalamatan di satu koneksi dengan tujuan untuk otokonfigurasi, pencarian mesin di jaringan atau ketika tidak ada *router* di jaringan. *Link-Local* digunakan untuk identifikasi alamat MAC. Sedangkan *Site-Local* adalah pengalamatan di suatu tempat tanpa membutuhkan prefix global.



Gambar 2.15 Format alamat Unicast Link-Local



Gambar 2.16 Format alamat Unicast Site-Local

b. Alamat Anycast

Alamat *Anycast* mempunyai pengalamatan tersendiri. Prefix alamat digunakan untuk menandai daerah topologi alamat *Anycast* pada sistem *routing* yang

terpisah. Alamat *Anycast* hanya untuk mengidentifikasi kumpulan *router* sehingga penggunaan secara global harus dibatasi.

n bit	(128-n) bit
Prefix subnet	0000000000000

Gambar 2.17 Bentuk Subnet-Router alamat Anycast IPv6

c. Alamat Multicast

1. Alamat *Multicast* tidak diperbolehkan menjadi alamat pengirim atau menjadi *header Routing.*

8 bit	4 bit	4 bit	112 bit
11111111	flags	Scope	ID group

Gambar 2.18 Format Alamat Multicast

Format tersebut mempunyai penjelasan sebagai berikut :

a) *Flags* : *Flags* berisi 4 bit. Tiga bit depan adalah cadangan dan di set nol, sedangkan bit terakhir adalah penunjuk status alamat. Jika bernilai 0 berarti pemberian alamat secara permanen, dan jika bernilai 1 maka pemberian alamat secara non-permanen.

b) *Scope* : Berisi 4 bit yang membatasi ruang lingkup grup *Multicast.* Nilai yang dimiliki *scope* adalah :

0	Cadangan	8	Ruang Lingkup Organisasi lokal
1	Ruang Lingkup *Node-Local*	9	Belum ditentukan
2	Ruang Lingkup *Link-Local*	A	Belum ditentukan
3	Belum ditentukan	B	Belum ditentukan
4	Belum ditentukan	C	Belum ditentukan
5	Ruang Lingkup *Site-Local*	D	Belum ditentukan
6	Belum ditentukan	E	Ruang Lingkup Global
7	Belum ditentukan	F	Cadangan

Tabel 2.3 Hexadesimal nilai scope

a) *ID group* : Mengidentifikasikan group *Multicast*. Secara non-permanen hanya berarti untuk ruang lingkup yang diberikan dan secara permanen berarti tidak bergantung dari ruang lingkup.

2. Alamat *Multicast* yang telah ditetapkan yaitu alamat *Multicast* cadangan yang tidak boleh diberikan kepada *Group Multicast* manapun.

FF00:0:0:0:0:0:0:0	FF04:0:0:0:0:0:0:0	FF08:0:0:0:0:0:0:0	FF0C:0:0:0:0:0:0:0
FF01:0:0:0:0:0:0:0	FF05:0:0:0:0:0:0:0	FF09:0:0:0:0:0:0:0	FF0D:0:0:0:0:0:0:0
FF02:0:0:0:0:0:0:0	FF06:0:0:0:0:0:0:0	FF0A:0:0:0:0:0:0:0	FF0E:0:0:0:0:0:0:0
FF03:0:0:0:0:0:0:0	FF07:0:0:0:0:0:0:0	FF0B:0:0:0:0:0:0:0	FF0F:0:0:0:0:0:0:0

Tabel 2.4 Alamat cadangan Multicast

Alamat *All-nodes* yang menandai group *router* dalam ruang lingkup 1 atau 2 adalah :

FF01:0:0:0:0:0:0:1 FF02:0:0:0:0:0:0:1

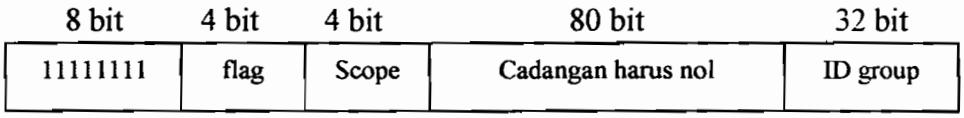
Alamat *All-routers* untuk mengenali group *router* dalam ruang lingkup 1, 2 atau 5 adalah :

FF01:0:0:0:0:0:0:2 FF02:0:0:0:0:0:0:2 FF05:0:0:0:0:0:0:2

Alamat *Solicited-Node* mengurangi jumlah alamat *Multicast* yang harus bergabung yaitu dengan mengambil 24 bit alamat dan ditambahkan prefix FF02:0:0:0:0:1:FF00::/104 :

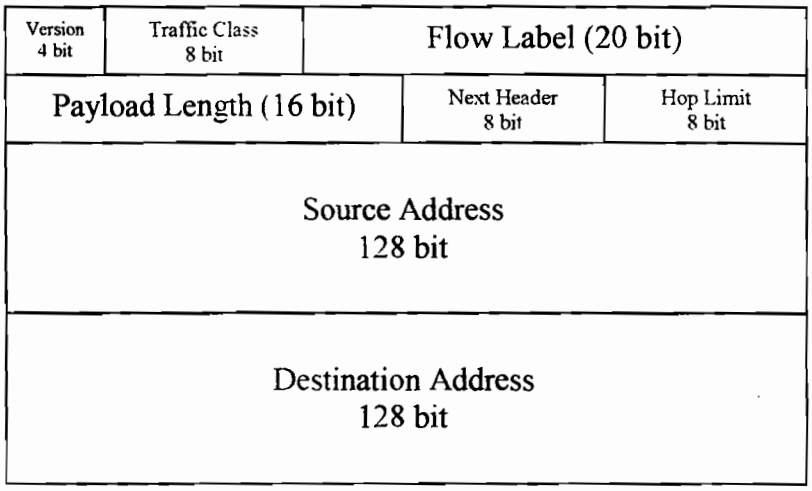
FF02:0:0:0:0:1:FFXX:XXXX

3. Pembagian alamat *Multicast* kedalam alamat MAC IEEE 802 dilakukan dengan mengambil 32 bit terakhir alamat *Multicast* (RFC 1972). ID group akan menghasilkan alamat MAC yang unik.



Gambar 2.19 Pemetaan alamat Multicast IPv6 ke alamat MAC IEEE 802

Header yang digunakan oleh IPv6 berbeda dengan header yang digunakan oleh IPv4. Panjang header yang digunakan pun berbeda. IPv4 memiliki panjang header yang bervariasi menurut keperluan, sedangkan IPv6 hanya memiliki 40 oktet.



Gambar 2.20 Format Header IPv6

Format header IPv6 terdiri dari :

a. *Version* : Versi dari IP yang digunakan.
b. *Traffic class* : Fungsinya mirip dengan TOS dalam IPv4 yaitu mengidentifikasi dan membedakan antar kelas.
c. *Flow Label* : Merupakan experimental dan digunakan oleh simpul sumber untuk menandai urutan paket.
d. *Payload Length* : Berfungsi sama dengan *Total Length* di IPv4.
e. *Next Header* : Merupakan header yang digunakan untuk fungsi tambahan.

f. *Hop Limit* : Merupakan lama waktu yang diperbolehkan paket untuk berada di jaringan. Fungsinya sama dengan *Time to Live* pada IPv4.

Pembagian alamat *network* dan *host* pada IPv6 dilakukan dengan menggunakan prefix. Nilai prefix menunjukkan alamat *network*.

2002:1ab2::2fe1:1/112

Alamat *network* dalam IPv6 diatas adalah 112 bit terdepan dan 16 bit di belakang adalah alamat *host*. Dengan demikian alamat *network* IPv6-nya adalah 2002:1ab2::2fe1:0 sedangkan alamat host yang dapat digunakan adalah 2002:1ab2::2fe1:1 sampai 2002:1ab2::2fe1:fffe

2.4.1.3. Transisi IPv4 ke IPv6

Dalam masa transisi ini, penggunaan IP yang kian banyak dan persediaan alamat IPv4 yang kian menipis membutuhkan suatu penanganan pengalamatan yang mampu untuk menangani pelayanan terhadap kebutuhan akan alamat IP. Keterbatasan IPv4 dalam memberikan suplai alamat IP dapat ditanggulangi dengan menggunakan beberapa mekanisme kerja yang dapat dilihat sebagai berikut :

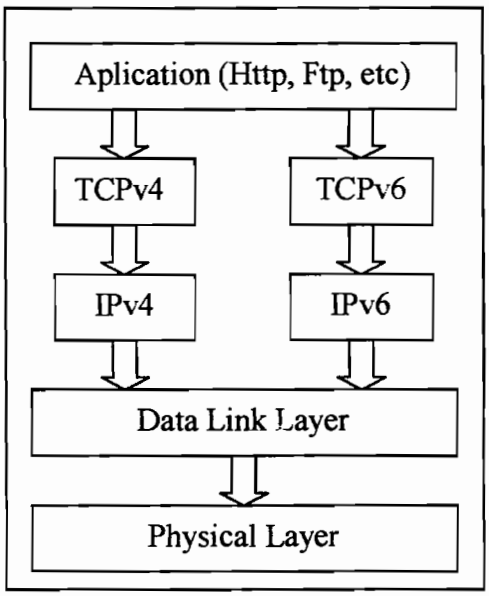
a. Menggunakan alokasi IPv4 yang dinamis dengan *Point to Point Protocol* (PPP).
b. Penggunaan alamat privat IPv4 dengan menggunakan protokol NAT (*Network Address Translation*)
c. Dengan melakukan *renumbering* terhadap IPv4.

Penggunaan alamat privat IPv4 dengan menggunakan protokol NAT sangat populer dan mudah. Karena dengan memiliki alokasi alamat IPv4 yang sedikit, *user* dapat menghubungkan jaringan dalam suatu institusi dengan dunia internet. Namun dalam beberapa kasus, penggunaan NAT tidak dapat dilakukan karena beberapa protokol tidak dapat berjalan melewati sebuah jaringan yang alamatnya telah tertranslasikan.

Pengalamatan menggunakan IPv6 menjadi suatu alternatif yang dapat digunakan untuk mengantisipasi keterbatasan alamat IPv4. Namun dengan penggunaan IPv6 saja, masalah belum terselesaikan. Hal ini dikarenakan masih banyak jaringan yang menggunakan IPv4, dan masih sedikit *router* yang mendukung penggunaan IPv6. Terutama adalah *dedicated router* versi lama yang masih digunakan sekarang ini (Cisco Router, 3COM) belum dilengkapi dengan pengalamatan IP yang menggunakan protokol IPv6. Jika akan beralih ke sebuah jaringan IPv6 secara keseluruhan, maka akan dibutuhkan biaya yang sangat besar untuk merombak susunan jaringan yang telah ada untuk digantikan dengan IPv6. Hal ini tidak mungkin dilakukan secara langsung karena beberapa pertimbangan diatas.

Berdasarkan kasus ini, maka dibuatlah suatu mekanisme komunikasi yang memungkinkan terjadinya komunikasi IPv6 yang memanfaatkan jaringan IPv4 sebagai jalur perantara. Pada masa transisi ini, penggunaan IPv6 dapat dilakukan dengan memanfaatkan jaringan IPv4 dengan menggunakan *dual IP layer* atau dengan menggunakan *tunneling*.

a. *Dual IP Layer* : Dalam satu komputer menyediakan dukungan untuk IPv4 maupun IPv6. Cara ini merupakan cara termudah dalam menangani proses transisi karena menyediakan implementasi IPv4 dalam mesin IPv6, sehingga dapat berhubungan dengan mesin IPv4 atau IPv6. Mesin dengan kemampuan seperti ini disebut sebagai mesin IPv6/IPv4.



Gambar 2.21 Format dual IP layer

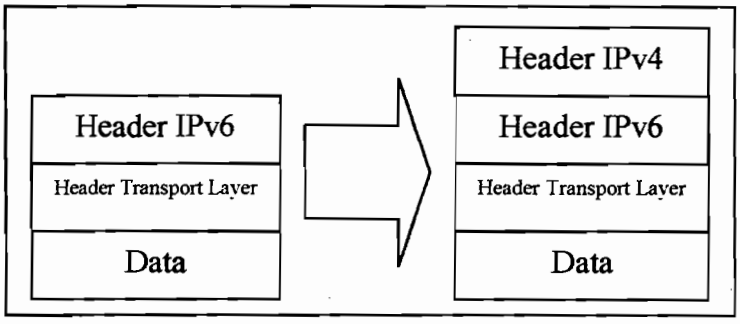
b. *Tunneling* : Pengelompokan teknik *tunneling* bergantung bagaimana mesin pembentuk *tunneling* menetukan alamat ujung *tunneling*. Pembentukan *tunneling* dapat dibagi menjadi 4 macam yaitu :

1. *Router* ke *router*.
2. Mesin ke *router*.
3. Mesin ke mesin.
4. *Router* ke mesin.

Pada dua *tunneling* pertama, ujung *tunneling* adalah *router* yang harus mendekapsulasi paket IPv6 dan mengirimkannya ke alamat tujuan. Jika

alamat IPv6 *router* berbeda dengan alamat tujuan paket dan paket tidak menyediakan alamat IPv4 *ruouter*, maka alamat *router* ditentukan dari konfigurasi mesin pembentuk *tunneling*. *Tunneling* yangng menggunakan cara seperti ini disebut sebagai *tunneling* terkonfigurasi dan dapat langsung terhubung dengan *backbone* IPv6.

Sedangkan pada dua *tunneling* terakhir, paket IPv6 dikirim ke alamat tujuan yang menentukan ujung *tunneling*. Jika alamat ujung *tunneling* adalah alamat IPv4-*Compatible*, maka 32 bit terakhir dapat dijadikan alamat IPv4. *Tunneling* yang menggunakan cara seperti ini disebut sebagai *tunneling* otomatis. Penentuan paket IPv6 yang dikirim ke *tunneling* menggunakan informasi *routing* IPv6. Implementasinya dapat berupa *routing* statis untuk prefix 0:0:0:0:0:0/96.

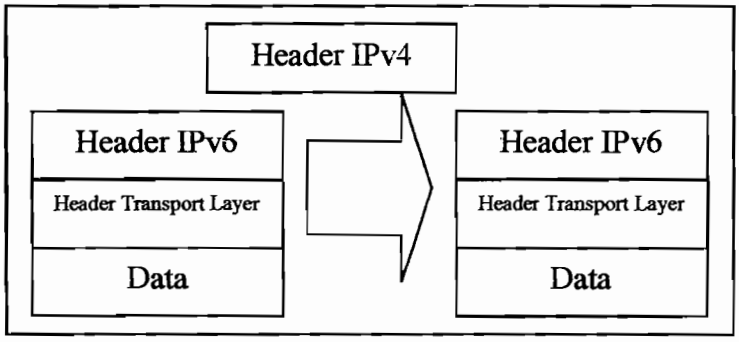


Gambar 2.22 Proses Enkapsulasi IPv6 atas IPv4

Beberapa header IPv4 yang harus diatur dalam proses enkapsulasi :

1. *Version* : 4
2. *IP header length* (32-bit word) : 5 (tidak ada pilihan IPv4 dalam *header* enkapsulasi).
3. *Type-of-service* : 0
4. *Total length* (byte) adalah panjang paket IPv6 dan *header* IPv4.

5. *Identification* dihasilkan secara unik untuk setiap paket IPv4 yang dikirim sistem.
6. *Flags* digunakan untuk mengontrol fragmentasi.
7. *Fragment offset* dipasang tergantung kebutuhan fragmentasi.
8. *Time-to-live* dipasang sesuai implementasi secara spesifik dan tergantung kebutuhan.
9. *Protocol* : 41 (nomor tipe muatan yang diberikan untuk IPv6).
10. *Header checksum* menghitung checksum dari *header* IPv6.
11. *Source address* adalah alamat IPv4 dari *interface* tempat paket enkapsulasi dikirim.
12. *Destination address* adalah alamat IPv4 pada ujung *tunneling*

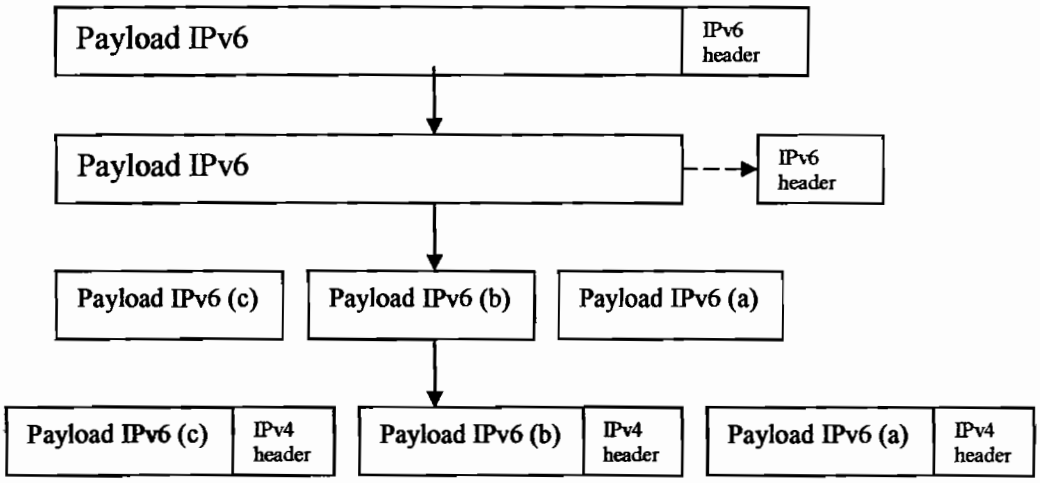


Gambar 2.23 Proses Dekapsulasi paket IPv6 atas IPv4

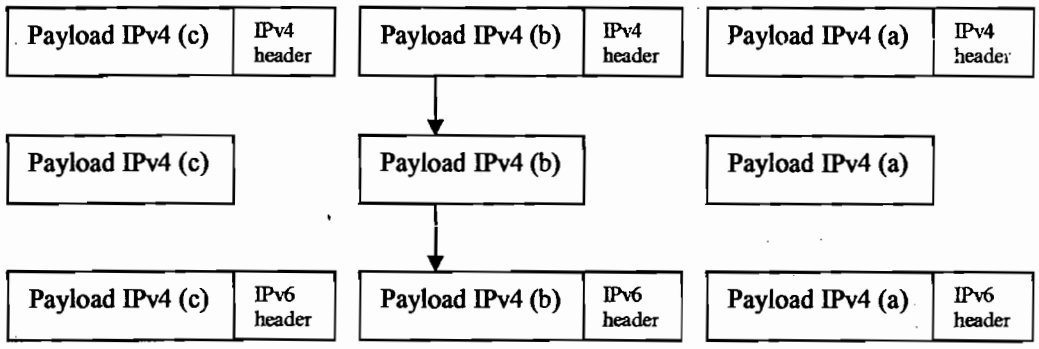
Jika mesin IPv6/IPv4 menerima paket IPv4 yang protokolnya bernilai 41, maka *header* IPv4 akan dibuang. *Header* IPv6 tidak akan berubah dan diproses seperti halnya paket IPv6 lainnya. Jika paket ternyata diteruskan, maka *Hop Limit*-nya akan berkurang satu.

c. Translasi : Metode ini adalah metode yang digunakan untuk mentranslasikan paket dengan alamat IPv6 menjadi paket dengan alamat IPv4 atau sebaliknya.

Dalam merubah paket tersebut, header paket IPv6 dihilangkan dan diganti dengan header IPv4. Panjang *payload* juga disesuaikan dengan paket yang baru, sehingga *payload* dipecah menjadi beberapa bagian sesuai panjang *payload* IPv4, dan dibungkus dengan alamat baru IPv4.



Gambar 2.24 Proses Translasi Paket IPv6 ke Paket IPv4



Gambar 2.25 Proses Translasi Paket IPv4 ke Paket IPv6

Terdapat beberap fasilitas yang dimiiiki oleh IPv6 dan menjadi sebuah nilai lebih jika dibandingkan dengan IPv4. Beberapa fasilitas tersebut adalah sebagai berikut :

a. Jumlah pengalamatan IP yang sangat luas (2^{128}).

b. Server-less Auto-Configuration.

c. Reconfigurasi (*Renumbering* / Pengalamatan Kembali).

d. Aggregasi berdasarkan hirarki alamat routing yang lebih effisien.

e. Header IPv6 yang lebih simpel (40 bytes).

f. Built-in IP-Layer privacy and authentification.

g. Streamleaned header format dan flow identification.

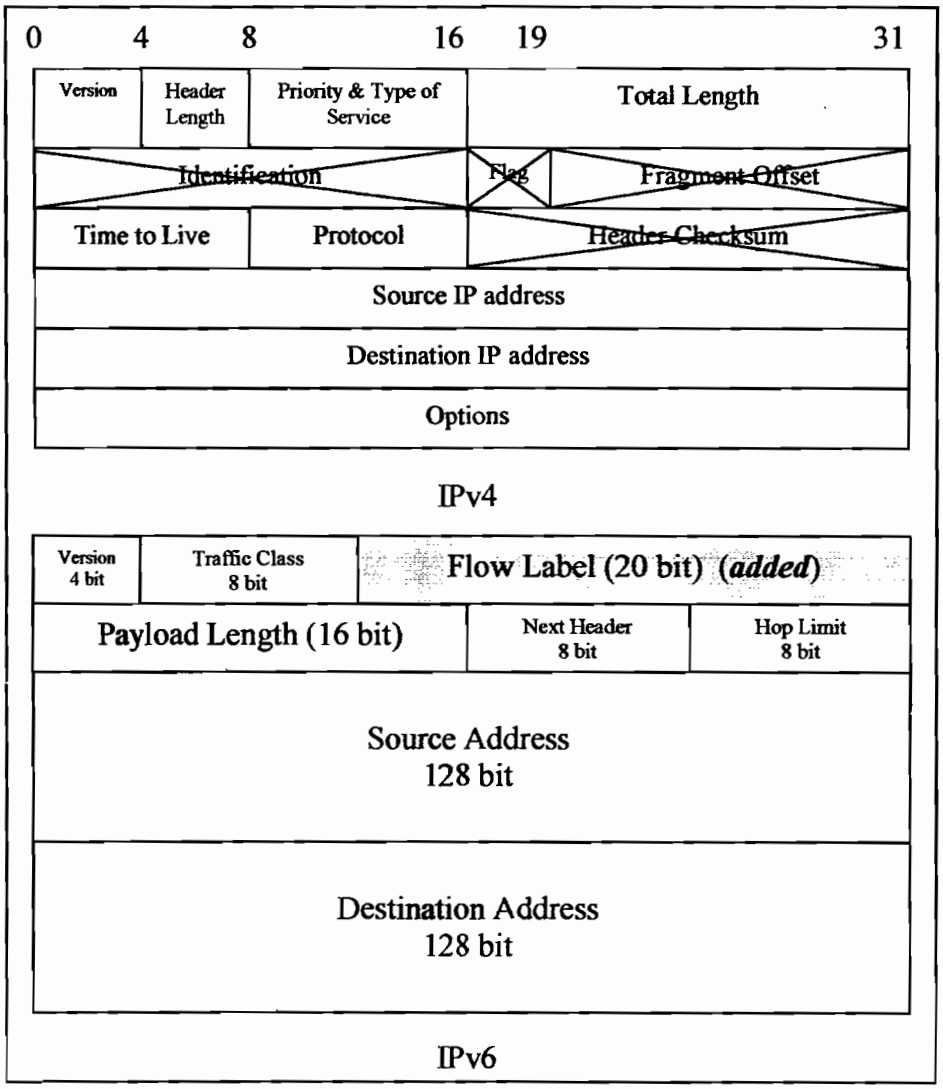
h. Improved support for option / extension.

Berikut ini adalah beberapa perbedaan antara IPv4 dan IPv6 yang dipandang dari beberapa tipe alamat dan fasilitas yang dimiliki :

	IPv4	IPv6
Alamat	32 bits (4 oktet)	128 bits (16 octet)
Ruang Alamat	2^{32} (lebih dari $10^9 \approx 4{,}2$ juta)	2^{128} (lebih dari $10^{38} \approx 3{,}40E38$)
Header Paket	Ukuran variabel - memakan waktu untuk menanganinya	Ukuran tetap (40 oktet) - Lebih efisien
Alokasi Alamat	Kelas A, B, C CIDR	Berbasis CIDR
Type Alamat	Unicast Anycast Broadcast	Unicast -Link-Local Address -Site-Local Address -Global Address Anycast Multicast
QoS	Digambarkan (ToS), tetapi tidak biasa digunakan	Flow Label Traffic Class
Security	Terbatas	IPsec Built-in
Konfigurasi	Konfigurasi Manual	Konfigurasi otomatis

Tabel 2.5 Perbedaan IPv4 dan IPv6

Dapat dilihat pula bahwa format header IPv4 berbeda dengan format header IPv6. Perbandingan format header IPv4 dan IPv6 adalah sebagai berikut :



Gambar 2.26 Perbandingan datagram paket IPv4 dengan IPv6

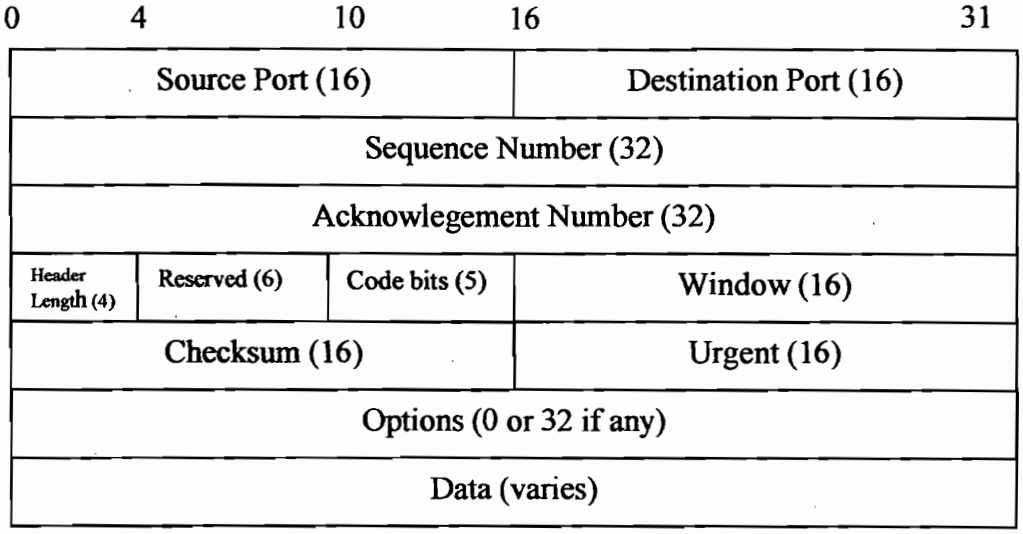
Dengan melihat perbandingan datagram antara IPv4 dengan IPv6 diatas, dapat dilihat bahwa IPv4 memiliki konten header yang lebih banyak dari IPv6. Dengan adanya konten header yang lebih sedikit dari IPv4, paket IPv6 akan mengalami *routing* yang lebih cepat dari pada paket IPv4. Hal ini dikarenakan pada saat melewati sebuah *node router*, *router* hanya mengecek header IP dan

meneruskannya kembali. Lama proses pengecekan paket ketika berada di *router* ditentukan oleh banyaknya konten header IP yang dimiliki oleh paket.

Namun dengan kelebihannya tersebut, pasti terdapat kekurangan yang dimiliki oleh datagram IPv6. Dengan mengurangi header paket, maka akan ada pengurangan fungsi dari header paket tersebut. Dengan menghilangkan *Header Checksum*, maka fasilitas *checksum* terhadap header paket juga dihilangkan. Dengan demikian dengan meningkatnya kecepatan dalam *routing*, penggunaan IPv6 harus menerima resiko bahwa dengan menggunakan IPv6 haruslah meningkatkan realibilitas dalam pengiriman paket.

2.4.2. TCP (Transmision Control Protocol)

TCP adalah pengarahan koneksi, protocol handal yang berada pada layer transport TCP/IP Protocol Stack. TCP bertugas memecah paket data menjadi beberapa bagian (*segmen*), menyatukan kembali (*reassemble*) pada stasiun tujuan. Apabila stasiun tujuan tidak menerima paket, atau menerima paket tetapi dalam keadaan rusak, TCP bertugas untuk mengirimkan kembali paket tersebut hingga paket diterima oleh stasiun tujuan secara lengkap dan tanpa kerusakan, dan menyatukan kembali pesan-pesan tersebut dari beberapa *segmen* menjadi satu paket utuh. TCP juga bekerja untuk mengatur bagaimana cara membuka hubungan komunikasi, jenis aplikasi apa yang digunakan dalam komunikasi tersebut. Dengan kata lain, TCP mengatur seluruh proses koneksi antar satu komputer dengan komputer yang lain dalam suatu jaringan.



Gambar 2.27 Segment TCP

Format segment TCP adalah :

a. *Source Port* : Nomor port pemanggil pada *host* pengirim (*client*).

b. *Destination Port* : Nomor port yang dipanggil pada *host* tujuan (*server*).

c. *Sequence Number* : Nomor untuk memastikan urutan kedatangan paket.

d. *Acknoledgement Number* : Perkiraan TCP oktet berikutnya.

e. *Header Lenght* : Nomor dari 32 bit pada panjang header.

f. *Reserved* : Diset nol.

g. *Code Bits* : Berfungsi sebagai pengendali termasuk setup dan akhiran sesi.

h. *Window* : Nomor oktet yang dapat diterima oleh suatu perangkat.

i. *Checksum* : Perhitungan checksum dari suatu header dan data field.

j. *Urgent* : Menunjukkan akhir dari sebuah data yang mendesak.

k. *Options* : Suatu penegasan → Ukuran segmen TCP maksimum.

l. *Data* : Layer atas data protokol.

Mekanisme kerja TCP adalah *connection oriented* yaitu TCP membangun suatu hubungan secara logika antar satu komputer dengan komputer lainnya. Dalam waktu yang ditentukan komputer yang sedang berhubungan harus mengirimkan data atau *acknowledge* agar hubungan tetap berlangsung. Jika hal ini tidak sanggup dilakukan maka dapat diasumsikan bahwa komputer yang sedang berhubungan mengalami gangguan dan hubungan secara logika dapat diputus.

Hal penting yang perlu dipahami pada TCP adalah *port number*. *Port number* menentukan service yang dilakukan oleh aplikasi diatas TCP. Penomoran port ini telah ditentukan oleh *Network Information Center* (NIC) dalam *Request For Comment* (RFC) 1010. Prinsip kerja TCP didasarkan pada prinsip *Client-Server*. *Server* adalah program yang secara pasif akan mendengarkan (*listen*) *port number* yang telah ditentukan oleh TCP. Sedangkan *client* adalah program yang secara aktif akan membuka hubungan TCP ke komputer *server* untuk meminta service yang dibutuhkan.

2.5. LINUX

Linux adalah suatu sistem operasi yang serupa dengan sistem operasi UNIX, dan merupakan implementasi independen dari sistem operasi POSIX, dengan extensi SYSV dan BSD sistem operasi UNIX, yang terutama berjalan pada mesin (*microprosesor*) Intel 80386DX, atau yang lebih baru. Linus Trovald mengembangkan sistem operasi ini dari hasil proyek hobinya dan terinspirasi dari Minix, suatu sistem UNIX kecil yang dikembangkan oleh Andy Tanenbaum. Linux versi 0.01 dikerjakan sekitar bulan Agustus 1991 dan resmi dipublikasikan

Linux versi 0.02 pada tanggal 5 Oktober 1991. Versi ini hanya dapat menjalankan *Bash* (GNU *Bourne Again Shell*) dan gcc (GNU *C Compiler*) (Wahana, 2003).

Linux dipublikasikan secara gratis dibawah lisensi GNU *General Public Licencell* (GPL). Hal ini berarti *source code* diikutsertakan dalam distribusinya. Dengan disertakan *source code* ini, setiap orang dapat melihat dan mengetahui isi program tersebut, sehingga setiap orang dapat mengubah, mengadaptasi atau mengembangkan program sesuai dengan keinginannya sendiri.

Sebagai sebuah sistem operasi yang dapat diperoleh secara gratis, Linux mempunyai keunggulan antara lain :

a. Freeware

   Linux adalah *freeware* yang dapat digunakan secara bebas tanpa perlu membeli lisensi.

b. Open Source

   Linux bersifat *open source* sehingga semua orang dapat melihat kode program dan mengembangkan secara bebas.

c. Minimal Hardware

   Linux dapat dioperasikan dengan dukungan *hardware* yang kecil. Bahkan Linux dapat dijalankan tanpa menggunakan hardisk (*diskless*).

d. Skalabilitas

   Linux dapat dijalankan dalam sebuah mesin sekecil Palm Pilot, atau bahkan sebuah mesin besar yang menggunakan *multiprocessors*.

e. Stabilitas

Linux adalah suatu sistem operasi yang stabil. Jika sudah terkonfigurasi secara benar dengan, maka Linux tidak perlu di reset.

f. Shared Libraries

Linux menggunakan penomoran versi *Shared Library* pada filenya yang memungkinkan pengguna untuk meng-*update* ke versi baru tanpa takut merusak keterkaitannya dengan file lain.

g. Non-Fragmentasi

Linux menggunakan file sistem *ext2fs* (*Second Extended File System*) yang memiliki keunggulan reduksi fragmentasi otomatis sehingga pengguna dapat menghapus dan menambah file tanpa takut terjadi fragmentasi.

h. Kebal Virus

Linux kebal terhadap virus.

i. Bugfix

Masalah keamanan sistem operasi tersebut terbuka untuk umum, sehingga jika menemukan bug dan langsung dipublikasikan, maka dengan segera dapat dicari di internet *bugfix*, *workaround*, *advisory*, dan sebagainya.

j. TCP/IP

Linux memiliki native protokol TCP/IP sehingga semua yang memanfaatkan TCP/IP akan dapat dilakukan lebih cepat dibanding sistem operasi lain non-UNIX.

k. File System 32 bit

Linux telah mendukung penuh file system 32 bit. Bahkan telah mendukung file system 64 bit. Sehingga linux dapat digunakan sebagai *server*.

l. Multi User

Linux dapat digunakan oleh lebih dari satu orang di mesin yang sama atau berbeda, menjalankan program yang sama atau berbeda pada saat bersamaan.

m. Multiconsole

User dapat login dengan nama yang sama atau berbeda di *console* lain tanpa menutup sesi sebelumnya.

n. Multitasking

Linux memungkinkan user untuk mengakses data atau menjalankan program secara bersama-sama pada *console* yang berbeda tanpa takut terjadi *stack* atau *hang* pada sistem operasi. Ini merupakan salah satu kestabilan Linux

o. Virtual Memory

Linux memiliki virtual memory yang memungkinkan user dapat memaksimalkan penggunaan memory lebih dari memory fisik.

p. Login User

Setiap user yang akan menggunakan Linux harus melakukan login terlebih dahulu. Linux tidak membatasi jumlah user yang akan dimasukkan, dan dengan user yang ada, Linux dapat memungkinkan untuk mengakses sampai 254 client secara bersamaan. Setiap user yang ada di Linux telah dilengkapi dengan password.

q. Akses Sistem File

Linux dapat digunakan untuk mengakses file sistem yang berbeda, seperti FAT, FAT32, NTFS, HPFS, MINIX, XENIX, Apletalk, Marsnwe, dan lain-lain.

r. Partition Mounting

Sebagaimana pembacaan partisi di sistem operasi MS Windows, Linux juga dapat melakukan pembacaan partisi. Hanya saja untuk mengenali partisi tersebut, linux harus melakukan *mount partition* dan ditempatkan di sebuah direktori.

s. WEB, FTP dan Proxy Server

Linux dapat digunakan sebagai WEB server, FTP server atau sebagai Proxy server yang dapat dimanfaatkan untuk keperluan internet.

t. Remote

Linux dapat diakses secara jarak jauh (remote).

u. Router dan Firewall

Linux dapat dipergunakan sebagai router maupun sebagai firewall meskipun dengan spesifikasi hardware yang minimal.

v. Shell Programable

Shell ini memungkinkan sistem menerima perintah dari user dan menjalankannya. Shell merupakan sebuah *interface* dalam Linux yang bersifat CLI (*Command Line Interpreter/Interface*).

w. Program

Distribusi linux sudah menyediakan program yang dibutuhkan sehingga tidak perlu membeli, mencari atau mendownload program aplikasi tambahan.

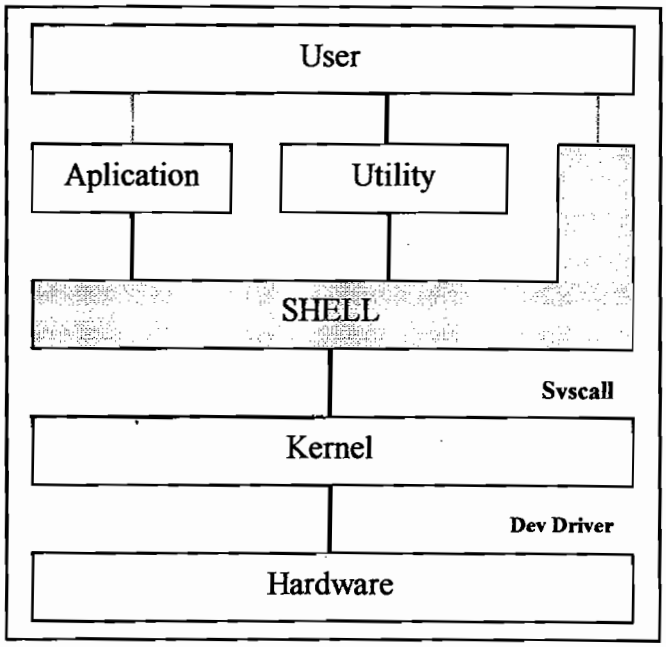
x. GUI (*Graphic User Interface*)

Linux juga telah mendukung tampilan berbasis grafis. Hal ini dimaksudkan untuk mempermudah user dalam memakai fasilias yang ada di Linux.

2.6 Shell

Shell merupakan nama yang diberikan untuk suatu program yang berfungsi untuk menjebatani user dengan sistem operasi UNIX (Susanto, 2001). Shell adalah bahasa penterjemah perintah (*command interpreter language*) atau sebuah prosesor makro yang menjalankan perintah (Rofiq Yuliardi, 2002). Shell menerima setiap perintah dari user dan menjalankan sesuai dengan fungsinya. Dengan kata lain Shell adalah penterjemah perintah (*command interpreter*).

Sebagai sebuah *interpreter* shell juga mempunyai kemampuan untuk menjalankan sekumpulan perintah. Perintah-perintah itu disimpan dalam satu file dan disebut sebagai *shell script*.



Gambar 2.28 Arsitektur Sistem Linux dan lokasi SHELL

2.6.1. Jenis-jenis Shell

Ada beberapa jenis pemrograman shell pada UNIX antara lain (Budi Susanto, 2001) :

a. **csh** (c shell) : suatu shell yang sintaknya mirip dengan bahasa C.
b. **tcsh** : c shell dengan penambahan karakteristik.
c. **ksh** (korn shell) : Korn shell merupakan shell yang dikembangkan melalui laboratorium Bell, AT&T oleh David Korn.
d. **zsh, pdksh** : keduanya adalah turunan dari korn shell dengan beberapa penambahan seperti TC shell.
e. **sh, bash** (shell, bourne again shell) : shell yang paling banyak digunakan.

2.6.2. Bash Shell

Bash adalah singkatan dari *Bourne Again Shell*. Bash adalah suatu bahasa pemrograman yang dapat menterjemahkan bahasa perintah, yang terdapat dalam sistem operasi GNU. Bash kompatibel dengan shell sh dan ditambah dengan kemampuan atau karakteristik yang dimiliki oleh Korn Shell (ksh) dan C Shell (csh) (Wahana, 2003).

Sebagai sebuah *programming language*, bash memungkinkan user dapat memprogram secara terstruktur. Namun sebagai *command language*, bash memungkinkan user untuk mengontrol eksekusi.

Sebagai *command language*, *Bourne Shell* memiliki karakteristik sebagai berikut :

a. Eksekusi Program : Memungkinkan menjalankan program secara berurutan maupun paralel.
b. Pembuatan File : Dapat membuat file yang baru dan menambah isi dari file yang sudah ada.
c. Argument Passing : Memungkinkan untuk melewatkan argumen ke program melalui baris perintah maupun environment variabel.
d. Eksekusi Program Berkondisi : Memungkinkan untuk menjalankan program berdasarkan pada keberhasilan atau kegagalan program lain.
e. Predefined Procedures : User dapat menyusun suatu script yang mendefinisikan suatu urutan program yang akan di eksekusi.

Sedangkan sebagai *programming language*, bash memiliki karakteristik sebagai berikut :

a. Variables : Dapat mendefinisikan variabel dan melakukan operasi terhadapnya.
b. Structured Language Constructs : Menyediakan struktur sequence, selection dan iterasi.
c. Scope : User dapat membatasi ruang lingkup dimana antara variabel dan program dapat saling mengenal.
d. Macro Substitution : Bash menyediakan macro dan header file.
e. Subroutines : User dapat menulis dan memanggil suatu subroutin dan memungkinkan untuk membuat suatu rekursi didalamnya.

Berikut adalah contoh penulisan bash script dalam suatu konsole[4] :

[4] Selanjutnya format penulisan bash adalah seperti berikut.

```
[root@linux /]# echo "hello word"
```

Script diatas ditulis langsung dalam konsol dan jika program dijalankan maka akan langsung menampilkan pesan dalam bentuk sebagai berikut :

```
[root@linux /]# echo "hello word"
hello word
[root@linux /]#
```

Dan dapat pula program ditulis dalam sebuah script. Penulisan script adalah sebagai berikut :

```
#! /bin/sh
echo "hello word"
echo "selamat pagi"
```

Ketika program itu dijalankan maka program tersebut akan menampilkan tulisan berupa :

```
hello word
selamat pagi
[root@linux /]#
```

Bash adalah suatu bahasa pemrograman yang dapat dijalankan dengan memanggil langsung nama filenya. Namun dalam file Linux terdapat mode yang digunakan untuk pembatasan akses oleh user (*file permission*). Untuk dapat menjalankan script bash, user harus mengganti mode file dengan perintah *chmod*.

File permission di Linux memiliki tiga mode yang berbeda untuk tiga pengguna. Yaitu mode baca (*read*), tulis (*write*) dan eksekusi (*execution*). Ketiga mode ini dimiliki oleh masing-masing pengguna.

	User	Group	Other
chmod	rwx	rwx	rwx

File permission biasanya hanya mengijinkan user untuk mengeksekusi script, namun tidak menutup kemungkinan penguna lain yang masih berada dalam satu group atau pengguna lain dapat mengeksekusi program tersebut.

Dari script diatas dimisalkan disimpan dalam file hello di directori home. Maka untuk mengaktifkan mode eksekusi script hello diatas adalah dengan cara :

```
[root@linux /]# chmod 755 /home/hello
```

Dan untuk menjalankan script diatas dapat dituliskan :

```
[root@linux /]# /home/hello
```

Namum apabila user berada dalam direktori dimana file itu berada, maka user dapat memanggil dengan perintah :

```
[root@linux /home]# ./hello
```

BAB III

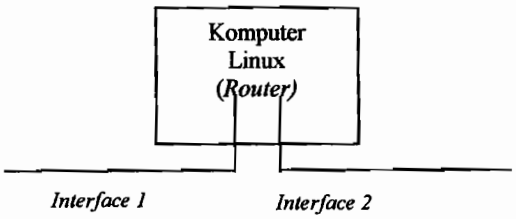
ANALISIS DAN PERANCANGAN SISTEM

3.1. ANALISIS

Dengan habisnya resource alamat IPv4, penggunaan IPv6 diharapkan dapat mencukupi kebutuhan alamat IP pada masa sekarang ini. Namun IPv6 adalah protokol baru yang tidak semua peralatan yang telah terhubung dengan internet mampu berkomunikasi menggunakan protokol IPv6 ini. Untuk mengganti jaringan IPv4 dengan IPv6 pun memerlukan dana yang tidak sedikit. Penggunaan mekanisme yang memanfaatkan jaringan IPv4 menjadi salah satu alternatif dalam penghematan biaya.

3.1.1. Strategi pada masa Transisi

Sistem operasi Linux memiliki fungsi yang dapat digunakan untuk *routing* paket. Dengan demikian sebuah komputer yang terisi dengan sistem operasi Linux dapat difungsikan sebagai sebuah komputer *router*. Sebuah komputer yang memiliki dua kartu jaringan (*interface*) dan masing-masing kartu memiliki alamat yang berbeda dan berada dalam segmen yang berbeda dapat difungsikan sebagai sebuah komputer *router*.



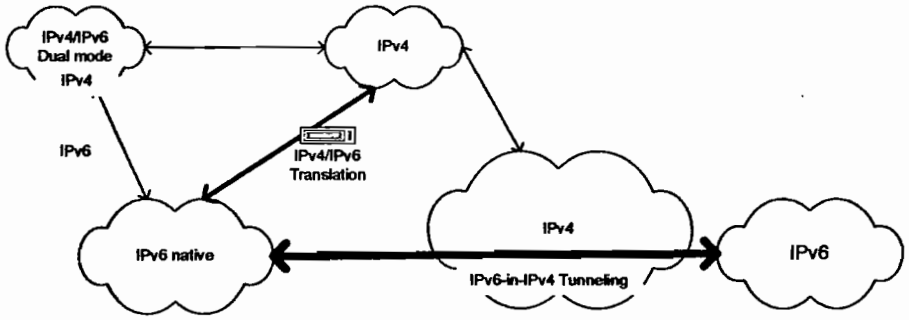
Gambar 3.1 Linux sebagai Router

Routing yang biasa ditemui adalah *routing* dengan menggunakan IPv4. Namun pada saat sekarang ini, IP yang dipakai bukan hanya IPv4 saja, namun protokol IPv6 telah dipakai dalam mengantisipasi keterbatasan alamat IPv4. Dalam masa transisi ini, orang harus mempelajari cara mensetting IPv6 untuk dipasang pada komputer.

Kecenderungan orang merasa malas untuk mempelajari serangkaian perintah yang sulit dan merepotkan. Oleh karena itu dibutuhkan suatu *tools* yang dapat membantu *administrator* dalam mensetting dan menyiapkan komputer agar dapat berhubungan dengan menggunakan IPv6. Dalam penelitian ini digunakan pemrograman bash untuk membuat *tools* yang dapat membantu *administrator* dalam melakukan setting konfigurasi IPv6 pada komputer, atau bahkan untuk mensetting komputer sebagai *router* secara mudah dan cepat.

Terutama dalam masa transisi penggunaan IPv4 ke IPv6 akan dibutuhkan suatu mekanisme yang dapat membantu penggunaan IPv6 dalam jaringan luas IPv4. Jaringan IPv6 dapat digunakan didalam jaringan IPv4 dengan menggunakan mekanisme *tunneling*, *dual stack* atau dapat menggunakan suatu komputer penterjemah (IPv4/IPv6 *Translation*). Dengan menggunakan metode seperti ini,

penggunaan IPv6 dapat digunakan secara lebih effisien meskipun sebuah jaringan memiliki satu buah IPv4 global.

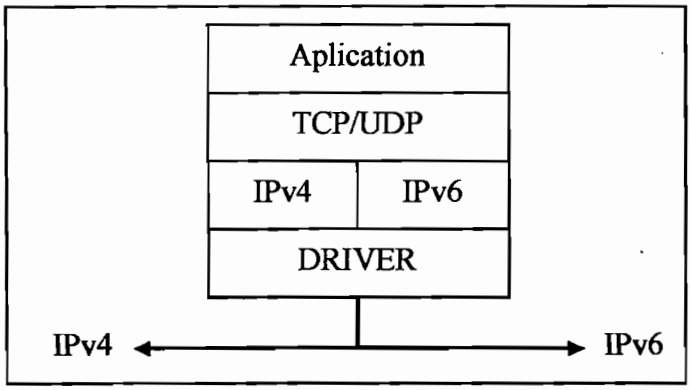


Gambar 3.2 Mekanisme Transisi IPv6

Dalam masa transisi IPv4 ke IPv6 ini dapat menggunakan tiga model dalam berkomunikasi. Komunikasi dari IPv6 ke IPv6, komunikasi IPv6 ke IPv4 atau sebaliknya dari IPv4 ke IPv6, atau bahkan komunikasi yang memungkinkan melewatkan IPv6 dalam jaringan IPv4. Model yang dipakai adalah *dual stack*, *tunneling* dan IPv4/IPv6 *Translation*.

a. Dual Stack

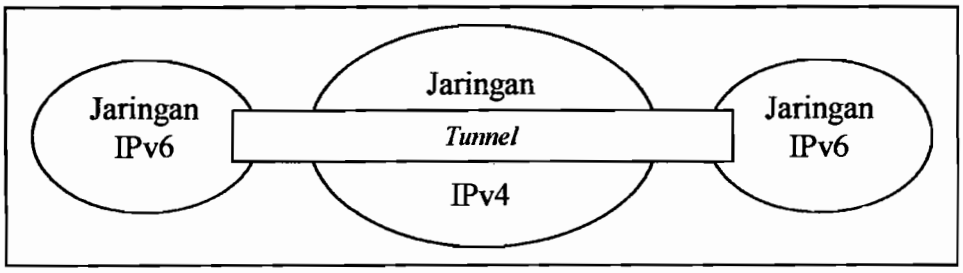
   Dual stack juga sering disebut sebagai dual IP layer adalah sebuah cara yang digunakan agar *host* atau *router* mendukung protokol IPv4 dan IPv6 secara bersama-sama.



Gambar 3.3 Metode Dual Stack atau Dual IP Layer

b. Tunneling

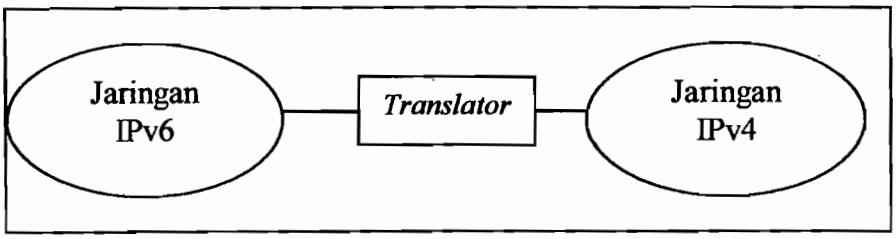
Tunneling adalah sebuah cara melakukan koneksi *point-to-point* dimana paket IPv6 beserta *header*-nya diubah menjadi paket IPv4 dan dikirim melalui infrastruktur jaringan IPv4.



Gambar 3.4 Metode Tunneling

c. IPv4/IPv6 Translation

IPv4/IPv6 *Translation* adalah cara melakukan koneksi antara IPv6 dengan IPv4 dan menterjemahkan alamat dari IPv6 menjadi IPv4 atau sebaliknya dari IPv4 menjadi IPv6.



Gambar 3.5 Metode IPv4/IPv6 Translation

3.1.2. Kasus Penanganan Routing IPv6

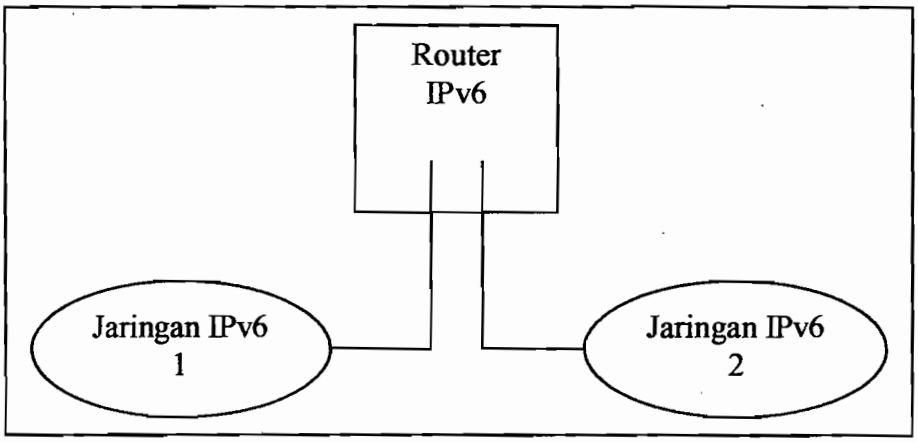
Ada beberapa kasus penanganan terhadap *routing* IPv6 dalam masa transisi ini. *Routing* sebenarnya digunakan untuk melewatkan paket dari jaringan satu ke jaringan lainnya. Sehingga memungkinkan bahwa jaringan terdekat adalah jaringan yang memiliki protokol IP yang sama atau bahkan jaringan yang memiliki protokol IP yang berbeda.

a. IPv6 Router

Kasus pertama adalah *router* yang menangani masalah *routing* terhadap sesama IPv6. Hal ini memungkinkan *administrator* untuk mensetting *router* hanya untuk IPv6 saja. Sehingga tidak membutuhkan mekanisme yang berhubungan dengan IPv4. Konsep yang dimiliki pun sama seperti konsep *routing* terhadap protokol IPv4.

Jaringan seperti ini dalam masa transisi sekarang ini hanya digunakan dalam lingkungan intranet saja, sehingga jarang yang menggunakan langsung terhubung dengan *backbone* IPv6 di jaringan internet. Hal ini disebabkan karena masih sedikit ISP yang menyediakan alamat IPv6 yang langsung terkoneksi dengan *backbone* IPv6 di internet. Sehingga dalam

pembangunan jaringan IPv6 sama sekali tidak memerlukan mekanisme yang berhubungan dengan IPv4.



Gambar 3.6 Mekanisme Router IPv6 terhadap jaringan IPv6

Jaringan semacam ini tidak dapat digunakan untuk koneksi internet jika salah satu *interface router* tidak ada yang terhubung ke *backbone* IPv6 di jaringan internet. Namum jika terdapat ISP yang telah terhubung dengan *backbone* IPv6, maka *router* ini dapat langsung berfungsi seperti layaknya *router* biasa.

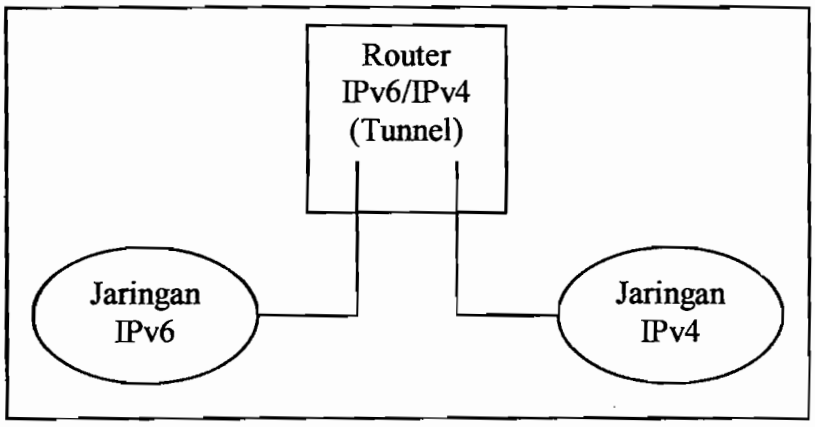
b. IPv6-to-IPv4 Router

Kasus yang kedua adalah *router* yang menangani masalah *routing* yang menghubungkan dua jaringan yang memiliki protokol IP yang berbeda yaitu antara jaringan yang menggunakan protokol IPv4 dengan jaringan yang menggunakan protokol IPv6. Hal seperti ini yang digunakan untuk mengantisipasi proses transisi dari IPv4 ke IPv6.

Jaringan IPv6 dapat diterapkan di sebuah jaringan intranet yang terhubung dengan sebuah *router* yang dapat menangani *forwarding packet* dari jaringan intranet IPv6 ke jaringan IPv4 di internet. Dengan menggunakan

model seperti ini, penggunaan IPv4 dapat dihemat karena hanya dengan menggunakan satu buah IPv4 publik, sebuah jaringan lokal IPv6 yang menggunakan alamat global yang dapat terhubung dengan internet.

Prinsip kerja dari model ini mirip dengan model kerja NAT (*Network Address Translation*). Hanya satu *interface* di komputer router yang terhubung ke jaringan internet dengan menggunakan IPv4 dan yang dan *interface* lainnya terhubung dengan jaringan lokal. Hanya saja jaringan lokal yang dipakai adalah jaringan lokal dengan alamat global IPv6.



Gambar 3.7 Mekanisme Router IPv6 yang terhubung dengan IPv4

3.1.3. Program IP pada Linux

Linux memiliki program yang terus berkembang. Terdapat beberapa macam perintah untuk menambahkan alamat IPv6 pada sebuah interface di komputer Linux. Salah satu perintah tersebut adalah perintah dari program *ip*[1]. Perintah *ip* digunakan karena memiliki struktur seperti *ifconfig* tetapi memiliki beberapa kelebihan yang tidak dimiliki oleh *ifconfig*.

---

[1] Sintak penulisan *ip* adalah menunjuk pada program *ip* yang berjalan pada Linux sedangkan alamat IP ditunjukkan dengan penulisan versi dibelakang IP. Contoh IPv4 atau IPv6.

Program *ip* memiliki beberapa *option* dan objek yang dapat digunakan dalam mensetting IPv4 maupun IPv6. Beberapa *option* dan objek yang dimiliki oleh program *ip* adalah :

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
where OBJECT := { link | addr | route | rule | neigh | tunnel
| maddr | mroute | monitor }
OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |
-f[amily] { inet | inet6 | ipx | dnet | link } | -o[neline]
```

Sedangkan sintak program yang dipakai dalam penulisan perintah program adalah sebagai berikut :

```
ip addr {add|del} IFADDR dev STRING
ip addr {show|flush} [ dev STRING ] [ scope SCOPE-ID ]
                     [ to PREFIX ] [ FLAG-LIST ]
                     [ label PATTERN ]
IFADDR := PREFIX | ADDR peer PREFIX
          [ broadcast ADDR ] [ anycast ADDR ]
          [ label STRING ] [ scope SCOPE-ID ]
SCOPE-ID := [ host | link | global | NUMBER ]
FLAG-LIST := [ FLAG-LIST ] FLAG
FLAG := [ permanent | dynamic | secondary | primary |
    tentative | deprecated ]
```

Dengan menggunakan dasar ini, dapat ditulis perintah untuk menambahkan alamat IPv6 dalam *device*. Namun dalam menambahkan IPv6, modul IPv6 harus aktif terlebih dahulu. IPv6 dapat berjalan dalam kernel yang telah mendukung IPv6. Cara untuk *cheking* kernel apakah telah mendukung IPv6 adalah dengan menggunakan perintah :

```
# test -f /proc/net/if_inet6 && echo "Kernel telah mendukung IPv6"
```

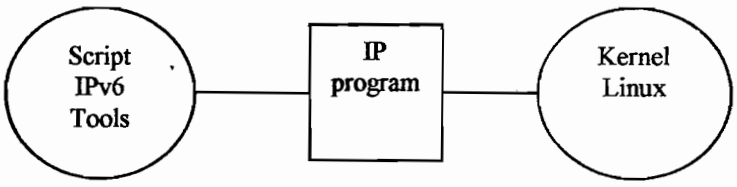
Untuk menambahkan IPv6 pada *interface* mengharuskan modul IPv6 terinstall terlebih dahulu. Sedangkan perintah untuk mengaktifkan modul IPv6 adalah dengan memberikan perintah :

```
# modprobe ipv6
```

Perintah untuk menambahkan IPv6 dalam *device* dapat menggunakan perintah :

```
# ip -f inet6 addr add 2002:abcd::1234 dev eth0
```

Perintah diatas adalah perintah untuk menempatkan alamat IPv6 pada komputer *host* dengan alamat IPv6 2002:abcd::1234. Penempatan alamat IPv6 tersebut diletakkan pada *device interface* eth0. Protokol IP yang digunakan ditunjukkan dengan perintah *–f inet6* yang berarti *–f(amily) inet6* atau protokol yang digunakan adalah protokol *inet6*.



Gambar 3.8 IP program sebagai Jembatan Aplikasi

Karena banyaknya perintah yang digunakan dalam mensetting IPv6, maka dibutuhkan *tools* yang dapat mempermudah pengunaan program *IP* dalam mensetting IPv6. Terlebih dalam setting ini hanya berlaku sementara atau jika komputer hidup, maka alamat IPv6 ini akan terus digunakan. Tetapi jika komputer

mengalami gangguan catu daya misalnya listrik mati atau komputer mengalami *hang* sehingga komputer harus di-*reboot*, maka alamat IPv6 ini akan hilang.

3.1.4. Analisis Kebutuhan

Analisis kebutuhan di sini dimaksudkan untuk mengumpulkan kebutuhan *administrator* dalam membangun jaringan IPv6 dengan mekanisme *tunneling*.

1. Analisis sistem.

   Berdasarkan analisis yang telah dilakukan, maka yang diperlukan dalam implementasi ini adalah:

   a. Menghubungkan jaringan IPv6 melalui jaringan IPv4.

   b. Memberikan kemudahan bagi *administrator* dalam mensetting IPv6.

   c. Memberikan fungsi otomatis terhadap setting IPv6.

2. Analisis pengguna.

   Sistem ini hanya ditujukan kepada *administrator* jaringan.

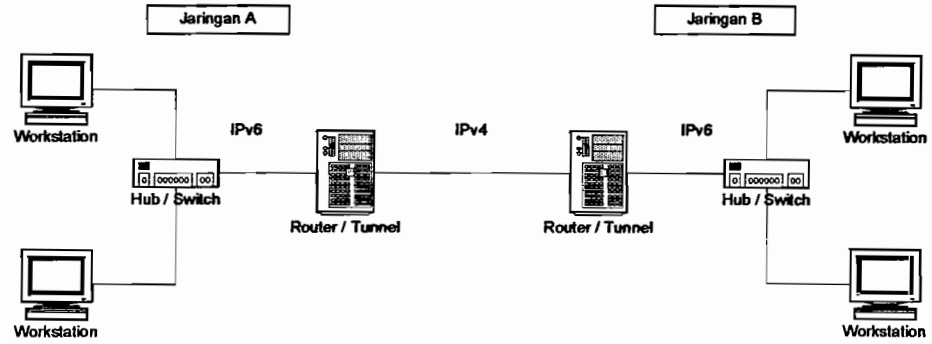
3.2. DESAIN

Tahap selanjutnya dari penelitian ini adalah perancangan program / *tools* yang akan digunakan pada komputer. Perancangan ini meliputi perancangan perangkat keras (*hardware*) dan perangkat lunak (*software*) yang akan dipakai.

3.2.1. Konfigurasi Jaringan

Konfigurasi jaringan yang dipakai dalam penelitian ini adalah menggunakan metode *tunneling*. Model jaringan ini adalah menempatkan

komputer yang memiliki IPv6 didalam dua buah jaringan yang terpisahkan oleh jaringan IPv4. Komputer yang memliki IPv6 akan melakukan pengiriman paket ke komputer IPv6 lainnya yang berada di jaringan lain, dengan melewatkan paket di jaringan IPv4.



Gambar 3.9 Konfigurasi Sistem

Konfigurasi diatas adalah bentuk konfigurasi secara fisik. Dalam bentuk logika, konfigurasi jaringan dapat digambarkan sebagai berikut :

Gambar 3.10 Konfigurasi Logika Sistem

3.2.2. Konfigurasi Hardware dan Software

Dalam penelitian ini perancangan dan pembuatan *tools* memerlukan perangkat bantu. Perangkat bantu tersebut adalah berupa perangkat keras (*hardware*) dan perangkat lunak (*software*). Perangkat keras yang dipakai adalah *Personal Computer* (PC) yang digunakan sebagai *host* maupun *router*.

PC yang digunakan untuk *host* adalah PC dengan spesifikasi sembarang yang memungkinkan IPv6 dapat terinstall, dengan memiliki satu buah NIC

(*Network Interface Card*) atau kartu jaringan. Sedangkan untuk *router* menggunakan PC dengan memiliki spesifikasi tertentu yang memiliki minimal dua buah kartu jaringan. Kartu jaringan pertama terhubung dengan jaringan IPv4 yang diasumsikan merupakan jaringan global (*internet*). Sedangkan kartu jaringan kedua terhubung dengan jaringan lokal IPv6.

Perangkat keras yang digunakan untuk *router* diharapkan memiliki spesifikasi minimal :

a. Prosesor : Intel Pentium II 350 MHz
b. Hardisk : 2 Gbyte
c. RAM : 32 Mbyte
d. NIC : NE2000 compatible sebanyak 2 buah

Penggunaan perangkat keras dengan spesifikasi minimal dapat dilakukan jika jaringan yang terhubung adalah jaringan lokal atau jaringan LAN kecil. Untuk jaringan yang terhubung dengan jaringan global (*internet*) memerlukan spesifikasi yang lebih tinggi karena *router* akan memerlukan *tabel routing* untuk menentukan arah *routing*. Maka rekomendasi perangkat keras yang dibutuhkan untuk router yang terhubung dengan jaringan global adalah:

a. Prosesor : Intel Pentium III 667 MHz atau lebih tinggi
b. Hardisk : 4 Gbyte atau lebih
c. RAM : 128 Mbyte atau lebih
d. NIC : NE2000 compatible sebanyak 2 buah

Client menggunakan spesifikasi perangkat keras yang lebih variatif karena kebutuhan dari tiap *client* berbeda-beda. Komputer *client* lebih sering digunakan

untuk keperluan kantor, sehingga diperlukan spesifikasi yang mampu menyediakan program keperluan kantor seperti program *office*. Oleh karena itu, spesifikasi yang diperlukan juga lebih tinggi dari komputer untuk *router*. Rekomendasi perangkat keras yang digunakan untuk *client* adalah:

a. Prosesor : Intel Pentium III 800 MHz atau lebih tinggi
b. Hardisk : 10 Gbyte atau lebih
c. RAM : 128 Mbyte atau lebih
d. NIC : NE2000 compatible sebanyak 1 buah

Konfigurasi perangkat lunak yang dipakai dalam pembuatan *tools* ini adalah berupa :

a. Linux Redhat 9.0 sebagai Sistem Operasi yang digunakan pada PC *router*. Paket RPM yang terinstall adalah paket standar untuk server.
b. Linux Redhat 9.0 atau Sistem Operasi lainnya yang telah mendukung IPv6 yang digunakan sebagai sistem operasi yang digunakan oleh *client*.

3.2.3. Bentuk umum program ip6

Tools yang akan dibuat adalah *tools* yang memanfaatkan program *ip* adalah merujuk pada option yang dimiliki oleh program *ip*. Karena *tools* ini dapat digunakan untuk mensetting IPv6, maka *tools* ini akan disebut sebagai *ip6*.

Program *ip6* adalah bentuk penyederhanaan program *ip*. Sehingga bentuk umum dari program yang akan dibuat akan mirip dengan program *ip*.

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
where OBJECT := { link | addr | route | rule | neigh | tunnel
| maddr | mroute | monitor }
OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |
-f[amily] { inet | inet6 | ipx | dnet | link } | -o[neline]
```

Dengan mempertimbangkan fasilitas yang akan digunakan oleh program *ip6*, maka *option* dari program *ip* dihilangkan. Program *ip6* hanya menggunakan *family* IPv6 saja, sehingga *option* langsung di set dengan protokol *family inet6*. Dengan demikian bentuk penulisan program *ip6* adalah:

```
ip6 OPSI {command | help}
OPSI := {add | del | tunnel | route}
command := {up | down | list | help}
```

OPSI yang dimiliki oleh perintah ipv6 ini adalah pilihan yang dapat digunakan dalam *tools*.

3.2.3.1. ip6 up

Perintah ini digunakan untuk mengaktifkan modul IPv6 dalam kernel sehingga komputer Linux dapat mengakses IPv6. Semua fasilitas IPv6 masih dalam keadaan mati sampai perintah ini dijalankan. Ketika IPv6 di-*install*-kan, sistem akan menaikkan modul IPv6, sehingga modul IPv6 akan langsung tertanam ke dalam sistem. Perintah ini juga akan menyisipkan perintah service ipv6 ke dalam service network. Jadi ketika service network di *restart*, service ipv6 juga akan ikut di-*restart*.

Perintah untuk mengaktifkan modul IPv6 ini adalah sebagai berikut:

```
ip6 up
```

3.2.3.2. ip6 down

Perintah ini merupakan kebalikan dari perintah ip6 up. Perintah ini digunakan untuk mematikan modul IPv6 yang berada dalam kernel Linux. Perintah ini digunakan untuk menghapus sisipan service ipv6 dari file *service network*.

Perintah yang digunakan untuk menghapus fungsi otomatis IPv6 dari *service* Linux adalah sebagai berikut:

```
ip6 down
```

3.2.3.3. ip6 add

Perintah ini digunakan untuk menambahkan IPv6 kedalam sebuah kartu jaringan. Karena IPv6 bersifat sementara dalam arti jika komputer mengalami *restart* maka IPv6 akan hilang, maka program ini akan mengingat alamat IPv6 ke dalam sebuah file pada saat menambahkan IPv6 kedalam *interface*.

Penulisan perintah yang akan digunakan untuk menambahkan alamat IPv6 adalah sebagai berikut :

```
ip6 add [alamat ipv6] [gw [gateway]] [dev]
```

Perintah di atas adalah untuk menambahkan alamat IPv6 kedalam suatu *interface* tertentu. *Gateway* dapat ditambahkan secara opsional, artinya boleh diisikan atau tidak diisikan.

3.2.3.4. ip6 del

Perintah ini digunakan untuk menghapus alamat IPv6 dari sebuah kartu jaringan. Dengan perintah ini, tabel alamat IPv6 yang menunjukkan kartu jaringan tertentu juga akan dihapus.

Penulisan perintah yang akan digunakan untuk menghapus alamat IPv6 adalah sebagai berikut:

```
ip6 del [ alamat ipv6 ] [ interface ]
```

Perintah diatas adalah untuk menghapus alamat IPv6 dari suatu *interface* tertentu. *Interface* ini dapat berupa *device* fisik maupun logika.

3.2.3.5. ip6 list

Perintah ini digunakan untuk menunjukkan atau melihat daftar interface beserta alamat-alanat yang dimiliki. Penulisan perintah yang digunakan untuk menampilkan alamat-alamat yang dimiliki oleh *interface* yang ada adalah :

```
#ip6 show [dev]
```

3.2.3.6. ip6 tunnel

Perintah ini digunakan untuk membangun jaringan *tunnel*, yaitu membangun koneksi *point-to-point* dengan mendefinisikan alamat IPv4 tujuan dan sumber pada sebuat *interface* logika. Perintah yang digunakan adalah :

```
ip6 tunnel [cmd] [dev] remote [ipv4 remote] local [ipv4 local]
```

Dalam perintah ini, cmd memiliki dua buah pilihan, yaitu *add* yang digunakan untuk menambahkan *interface* logika, dan *del* yang digunakan untuk menghapus *interface* logika tersebut.

3.2.3.7. ip6 route

Perintah ini digunakan untuk menangani masalah *routing* ipv6 pada jaringan ipv6. Perintah ini terbagi menjadi dua bagian yaitu untuk mengatur *routing* dan untuk melihat tabel *routing*. Perintah untuk pengaturan *routing* adalah sebagai berikut:

```
ip6 route [ command ] [ destination ] via [ neigh addr ]
```

Dimana *command* dapat digunakan untuk menambah tabel routing atau dapat digunakan untuk menghapus tabel *routing*. Untuk menambah *routing* digunakan *add*, sedangkan untuk menghapus tabel *routing* digunakan perintah *del*. Sedangkan untuk melihat tabel *routing*, perintah yang dipakai adalah:

```
ip6 route show
```

BAB IV

IMPLEMENTASI PROGRAM

Implementasi merupakan tahap pengkodean hasil perancangan dan menerapkannya pada lingkungan yang tepat. Pada bab sebelumnya telah dijelaskan bagaimana perancangan program untuk membentuk koneksi otomatis IPv6 dalam mekanisme *tunneling*. Pada bab ini akan dijelaskan mengenai implementasi beserta analisa hasilnya.

4.1. Lingkungan Implementasi

Pada tahap impelmentasi, pengembang harus mengetahui dan mendefinisikan lingkungan yang hendak dipakai. Lingkungan-lingkungan yang mendukung antara lain adalah lingkungan perangkat keras, dan lingkungan perangkat lunak.

4.1.1. Lingkungan perangkat lunak

Perangkat lunak yang dipakai di dalam implementasi ini adalah sebagai berikut:

a. Sistem Operasi : Linux Redhat 9.0

b. Bahasa pemrograman : Bash

4.1.2. Lingkungan perangkat keras

Perangkat keras yang mendukung dalam implementasi ini dibagi menjadi dua bagian, yaitu perangkat keras untuk *router* dan perangkat keras untuk *client*. Adapun perangkat keras yang digunakan sebagai *router* adalah sebagai berikut:

a. Komputer Pentium II 300 Mhz atau lebih tinggi.
b. Memori minimal 64 MB.
c. Harddisk minimal 2 GB.
d. VGA card 1 MB.
e. NIC 2 buah.
f. Alat input berupa keyboard.
g. Alat keluaran berupa monitor.

Sedangkan kebutuhan perangkat keras yang digunakan untuk *client* adalah sebagai berikut:

a. Komputer Pentium III 800MHZ atau lebih tinggi.
b. Memori minimal 128 MB.
c. Harddisk minilam 5 MB.
d. VGA card minimal 8 MB.
e. Alat input berupa keyborad dan mouse.
f. Alat keluaran berupa monitor SVGA denga resolusi 800 x 600.

4.2. Karakteristik pengguna

Sistem ini hanya ditujukan kepada *administrator*, sehingga hanya *administrator* saja yang dapat menggunakan program ini. Adapun syarat-syarat yang harus dimiliki oleh *administrator* yaitu:

a. Memahami konsep jaringan secara umum.

b. Menguasai jaringan dengan protokol IPv4 maupun IPv6.

c. Memahami konsep *point-to-point* protokol.

d. Memahami konsep *tunneling*.

4.3. Implementasi Program ip6

Dalam implementasi program ini akan dijelaskan kegunaan dari masing-masing fungsi. Dalam bab perancangan telah disebutkan bahwa program ini memanfaatkan program yang telah ada dalam linux yaitu program ip.

4.3.1. Inisialisasi lokasi direktori

Untuk menentukan lokasi direktori; maka inisialisasi alamat dibutuhkan untuk mempermudah dan mempersingkat penulisan lokasi.

```
ADDR_MOD=/etc/sysconfig
ADDR_DIR=/etc/sysconfig/IPv6
ADDR_NET=/etc/init.d
TMP=/tmp
```

Inisialisasi alamat ini digunakan untuk menentukan dimana proses program akan dijalankan atau dimana file akan diletakkan. Semua setting IPv6 nantinya akan diletakkan dalam lokasi ADDR_DIR atau menunjuk pada /etc/sysconfig/IPv6 pada sistem. Sedangkan semua services akan ditunjukkan oleh

ADDR_NET atau /etc/init.d pada sistem. Sedangkan *temporary* didefinisikan di TMP yang menunjuk lokasi /tmp di sistem.

4.3.2. Inisialisasi Passing Parameter

Inisialisasi passing parameter digunakan untuk mendefinisikan nama variabel input.

```
INPUT1=$1
INPUT2=$2
INPUT3=$3
INPUT4=$4
INPUT5=$5
INPUT6=$6
INPUT7=$7
```

Inisialisasi ini berguna untuk mempermudah *tracking* errorr pada saat input variabel ke dalam program.

4.3.3. Fungsi aktif

Sistem operasi Redhat 9.0 dalam keadaan default hanya mengaktifkan protokol IPv4 saja. Oleh karena itu untuk dapat menggunakan protokol IPv6, sistem harus di install modul IPv6. Pada dasarnya Redhat 9.0 sudah mendukung IPv6, namun belum diaktifkan secara otomatis. Untuk mengaktifkannya *administrator* perlu melakukan instalasi modul IPv6 dan menseting agar modul IPv6 langsung terangkat ketika sistem mengalami *reboot*. Dalam implementasi ini, instalasi modul akan didefinisikan sebagai angka satu dan disimpan dalam suatu file. Instalasi modul ini juga menempatkan inisialisasi pada file yang terletak di ADDR_MOD/network dan ditambahkan NETWORKING_IPV6=yes pada file tersebut. Fungi dari inisialisasi tersebut adalah agar IPv6 dapat langsung

terinstall ketika komputer mengalami *reboot*. Demikian pula disisipkan perintah untuk *start* dan *stop service* IPv6 pada file *service network* yang terletak di ADDR_NET/network. Kode program dari fungsi ini adalah sebagai berikut:

```
aktif()
{
# jika direktori /etc/sysconfig/IPv6 ada
if [[ -d $ADDR_DIR ]]
then
    # cek apakah sudah aktif atau belum
    if [[ -a $ADDR_DIR/IPv6 && `cat $ADDR_DIR/IPv6` == 1 ]]
    then
      echo "Modul IPv6 dalam kondisi otomatis"
    else
      # jika belum
      # Mengaktifkan fungsi otomatis IPv6 dalam sistem
      echo "1" > $ADDR_DIR/IPv6
      modprobe IPv6
      sed '/NETWORKING/a NETWORKING_IPV6=yes' \
      $ADDR_MOD/network > $TMP/temp6
      cp --reply=yes $TMP/temp6 $ADDR_MOD/network
      rm -f $TMP/temp6

      # Menyisipkan fungsi IPv6 ke dalam fungsi service network
      # agar IPv6 dapat diangkat secara otomatis
      sed '/touch/a #aktifkan IPv6' $ADDR_NET/network >$TMP/net
      sed '/#aktifkan/a service IPv6 start' $TMP/net >$TMP/net1
      sed '/# and/a #deaktif IPv6' $TMP/net1 > $TMP/net
      sed '/#deaktif/a service IPv6 stop' $TMP/net > $TMP/net1
      cp --reply=yes $TMP/net1 $ADDR_NET/network
      rm -f $TMP/net
      rm -f $TMP/net1
    fi
else
    # jika folder IPv6 tidak ada, maka buat dulu
    mkdir $ADDR_MOD/IPv6
    aktif
fi
}
```

Dalam fungsi ini terdapat beberapa bagian yang digunakan untuk mendefinisikan keadaan otomatis IPv6. Untuk mengetahui apakah ADDR_DIR ada atau tidak adalah dengan menjalankan perintah berikut ini:

```
if [[ -d $ADDR_DIR ]]
```

Parameter –d adalah parameter untuk test apakah ADDR_DIR itu ada dan berupa direktori atau bukan. Kemudian jika direktori ditemukan, maka langkah

selanjutnya adalah melakukan test terhadap status modul IPv6. Perintah untuk mengetahui status modul adalah sebagai berikut:

```
if [[ -a $ADDR_DIR/IPv6 && `cat $ADDR_DIR/IPv6` == 1 ]]
```

Jika dalam ADDR_DIR terdapat file IPv6 dan isi dari file IPv6 tersebut adalah satu, maka modul IPv6 sudah dalam keadaan otomatis dan fungsi selesai. Namun bila kondisi diatas ternyata salah, maka fungsi akan menjalankan perintah selanjutnya yaitu menginstall modul kemudian menyisipkan inisialisasi otomatis dan service IPv6. Perintah yang dijalankan adalah sebagai berikut:

```
modprobe IPv6
sed '/NETWORKING/a NETWORKING_IPV6=yes' \
      $ADDR_MOD/network > $TMP/temp6
cp --reply=yes $TMP/temp6 $ADDR_MOD/network
rm -f $TMP/temp6
sed '/touch/a #aktifkan IPv6' $ADDR_NET/network >$TMP/net
sed '/#aktifkan/a service IPv6 start' $TMP/net >$TMP/net1
sed '/# and/a #deaktif IPv6' $TMP/net1 > $TMP/net
sed '/#deaktif/a service IPv6 stop' $TMP/net > $TMP/net1
cp --reply=yes $TMP/net1 $ADDR_NET/network
rm -f $TMP/net
rm -f $TMP/net1
```

Jika dalam ADDR_DIR tidak ditemukan file IPv6, maka fungsi akan membuat file IPv6 terlebih dahulu dan kemudian dilanjutkan dengan menjalankan kembali fungsi aktif ini secara rekursif. Perintahnya adalah sebagai berikut:

```
mkdir $ADDR_MOD/IPv6
aktif
```

4.3.4. Fungsi deaktif

Hampir sama dengan fungsi aktif, fungsi ini hanya bertugas kebalikan dari fungai aktif. *Administrator* mengubah nilai pada file ADDR_DIR/IPv6 menjadi nol untuk mengubah status modul IPv6. Dalam kasus ini modul yang telah terinstall akan berada dalam sistem selama komputer menyala. Untuk itu

administrator juga perlu menghapus baris NETWORKING_IPV6=yes dari lokasi ADDR_MOD/network. Hal ini ditujuakan agar modul tidak terinstall ketika komputer mengalami *restart*. *Administrator* juga harus menghapus service IPv6 yang disisipkan di service network. Berikut adalah kode program dari fungsi deaktif:

```
deaktif()
{
# cek kondisi IPv6 apakah sudah dalam kondisi off
if [[ `cat $ADDR_DIR/IPv6` == 0 ]]
then
    echo "Modul IPv6 dalam kondisi manual"
else
    # Menghapus fungsi otomatis IPv6 dari sistem
    echo "0" > $ADDR_DIR/IPv6
    sed '/NETWORKING_IPV6/d' $ADDR_MOD/network > $TMP/temp6
    cp --reply=yes $TMP/temp6 $ADDR_MOD/network
    rm -f $TMP/temp6

    # Menghapus fungsi IPv6 dari fungsi service network
    sed '/#aktifkan IPv6/d' $ADDR_NET/network > $TMP/net
    sed '/service IPv6 start/d' $TMP/net > $TMP/net1
    sed '/#deaktif IPv6/d' $TMP/net1 > $TMP/net
    sed '/service IPv6 stop/d' $TMP/net > $TMP/net1
    cp --reply=yes $TMP/net1 $ADDR_NET/network
    rm -f $TMP/net
    rm -f $TMP/net1
fi
}
```

Untuk melihat status dari IPv6 adalah dengan melihat isi dari file IPv6 yang berada dalam ADDR_DIR. Perintah yang digunakan untuk melihat status tersebut adalah sebagai berikut:

```
if [[ `cat $ADDR_DIR/IPv6` == 0 ]]
```

Jika nilai yang dikandung file IPv6 adalah nol, maka program IPv6 ini berjalan pada kondisi manual dalam arti jika komputer mengalami *reboot*, maka modul IPv6 tidak terinstall, ketika *service network* di *start* atau di *stop*, *service* IPv6 tidak ikut dijalankan.

```
echo "0" > $ADDR_DIR/IPv6
sed '/NETWORKING_IPV6/d' $ADDR_MOD/network > $TMP/temp6
cp --reply=yes $TMP/temp6 $ADDR_MOD/network
rm -f $TMP/temp6
```

Skript diatas digunakan untuk menghapus fungsi otomatis instalasi IPv6 ketika komputer mengalami *reboot*. Sedangkan fungsi untuk menghapus sisipan *service* IPv6 dari file *service network* adalah sebagai berikut:

```
sed '/#aktifkan IPv6/d' $ADDR_NET/network > $TMP/net
sed '/service IPv6 start/d' $TMP/net > $TMP/net1
sed '/#deaktif IPv6/d' $TMP/net1 > $TMP/net
sed '/service IPv6 stop/d' $TMP/net > $TMP/net1
cp --reply=yes $TMP/net1 $ADDR_NET/network
rm -f $TMP/net
rm -f $TMP/net1
```

4.3.5. Fungsi tambah

Fungi ini digunakan untuk menambah alamat IPv6 kedalam *interface* baik *interface* fisik maupun *sit device*. Dalam fungsi ini untuk menambahkan alamat ke dalam *interface* dapat memilih dengan menambahkan *default gateway* atau tidak. Pemilihan pengisian *gateway* hanya dapat dilakukan untuk *interface* fisik saja. Sedangkan untuk *sit device* hanya dapat diisi oleh alamat IPv6 saja. Kode program untuk menambahkan alamat IPv6 kedalam *intercafe* adalah sebagai berikut:

```
Tambah()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_tambah
elif [[ $INPUT3 == eth? ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT3"
elif [[ $INPUT3 == gw ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT5
    ip route add ::/0 via $INPUT4
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT5"
```

```
    echo "gateway= $INPUT4" >> "$ADDR_DIR/ip6cfg-$INPUT5"
elif [[ $INPUT3 == sit? ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "$INPUT2 dev $INPUT3" > $ADDR_DIR/ip6cfg-$INPUT3
else
    help_tambah
fi
}
```

Dalam fungsi diatas setiap perintah penambahan akan menyimpan konfigurasi IPv6 kedalam file tertentu. Untuk menambahkan alamat IPv6 kedalam *interface ethernet* dapat dilihat dalam kode program berikut ini:

```
elif [[ $INPUT3 == eth? ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT3"
```

Pada potongan program diatas adalah untuk menambahkan alamat IPv6 kedalam *interface* tertentu. Kemudian alamat IPv6 tersebut disimpan dalam sebuah file yang berada di ADDR_DIR dan diberi nama ip6cfg-eth?. Hal ini dimaksudkan untuk menambahkan alamat IPv6 ke *interface* yang di tunjuk agar dapat ditambahkan secara otomatis.

```
elif [[ $INPUT3 == gw ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT5
    ip route add ::/0 via $INPUT4
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT5"
    echo "gateway= $INPUT4" >> "$ADDR_DIR/ip6cfg-$INPUT5"
```

Potongan program diatas juga digunakan untuk menambahkan alamat IPv6 kedalam *interface* yang di tunjuk. Namun dalam program ini ditambahkan pula fungsi untuk memberikan konfigurasi *default gateway*. Dalam bagian ini konfigurasi alamat IPv6 dan setting *default gateway* juga langsung di simpan dalam file.

```
elif [[ $INPUT3 == sit? ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "$INPUT2 dev $INPUT3" > $ADDR_DIR/ip6cfg-$INPUT3
```

Sedangkan potongan program diatas digunakan untuk menambahkan alamat IPv6 kedalam *sit device*. *Interface* yang digunakan diberi nama sit?. Jadi untuk menambahkan alamat IPv6 ke dalam *interface sit* ini, terlebih dahulu *device sit* ini ditambahkan dari fungsi *tunnel*. Konfigurasi alamat ini juga disimpan dalam file yang terletak di direktori ADDR_DIR dan diberi nama ip6cfg-sit?.

4.3.6. Fungsi hapus

Dalam fungsi hapus ini fungsi mempelajari alamat yang telah ada dalam sistem kemudian membandingkan dengan alamat yang ada dalam file konfigurasi yang ada. Jika alamat yang akan dihapus antara alamat yang berada dalam sistem dengan yang ada dalam file konfigurasi ternyata sama, maka alamat dalam sistem dihapus kemudian file konfigurasi juga akan dihapus. Proses tersebut berlaku untuk menghapus alamat IPv6 dari *interface* fisik maupun logika. Bentuk kode program dari fungsi hapus adalah sebagai berikut:

```
Hapus()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_hapus
elif [[ $INPUT3 == eth? ]]
then
# Membandingkan IPv6 yang aktif dengan yang ada di tabel IPv6
    ip6 list $INPUT3 > $TMP/list
    cat $TMP/list |grep inet6|grep global > $TMP/list6
    cat "$ADDR_DIR/ip6cfg-$INPUT3"|grep address > $TMP/addr
    asli=`awk '{print $2}' "$TMP/addr"`
    scan=`awk '{print $2}' "$TMP/list6"`
    # Untuk menghapus IPv6 dari sistem
    if [[ $asli == $scan ]]
    then
      ip -f inet6 addr del $INPUT2 dev $INPUT3
```

```
      rm -f $ADDR_DIR/ip6cfg-$INPUT3
    fi
    rm -f $TMP/list
    rm -f $TMP/list6
    rm -f $TMP/addr
elif [[ $INPUT3 == sit? ]]
then
    ip6 list $INPUT3 |grep global > $TMP/hpssit
    asli=`awk '{print $2}' "$TMP/hpssit"`
    scan=`awk '{print $1}' "$ADDR_DIR/ip6cfg-$INPUT3"`
    if [[ $asli == $scan ]]
    then
        ip tunnel del $INPUT2 dev INPUT3
      rm -f $ADDR_DIR/ip6cfg-$INPUT3
    fi
    rm -f $TMP/hpssit
else
    help_hapus
fi
}
```

4.3.7. Fungsi tampil

Fungsi tampil adalah fungsi yang digunakan untuk menampilkan daftar alamat IPv6 yang ada dalam sistem. Dalam fungsi ini program dapat menampilkan keseluruhan daftar alamat IPv6 yang ada dalam sistem, atau hanya *interface* tertentu saja. Kode program fungsi ini adalah sebagai berikut:

```
tampil()
{
if [[ $INPUT2 == eth? || $INPUT2 == sit? ]]
then
    ip -f inet6 addr show dev $INPUT2
else
    ip -f inet6 addr
fi
}
```

4.3.8. Fungsi Tunnel

Dalam program ini, fungsi *tunnel* adalah fungsi yang digunakan untuk membentuk koneksi IPv6 melalui jaringan IPv4. Dalam fungsi ini hanya terdapat

dua bagian yaitu untuk menambahkan *device tunnel* kedalam sistem serta membentuk koneksinya, dan untuk menghapus *device tunnel* dari sistem.

```
tunel()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_tunel
elif [[ $INPUT2 == add && $INPUT3 == sit? && \
     $INPUT4 == remote && $INPUT6 == local ]]
then
    ip tunnel add $INPUT3 mode sit remote $INPUT5 local \
    $INPUT7 2> $TMP/ceksit

    if [[ `cat $TMP/ceksit` == "" ]]
    then
      UP="ip link set $INPUT3 up"
      $UP
      echo "$INPUT3 remote $INPUT5 local $INPUT7" > \
      $ADDR_DIR/$INPUT3
    else
      echo "Perhatikan penulisan tunneling"
    fi
    rm -f $TMP/ceksit
elif [[ $INPUT2 == del && $INPUT3 ]]
then
    if [[ -e $ADDR_DIR/$INPUT3 ]]
    then
      DOWN="ip link set $INPUT3 down"
      ip tunnel del $INPUT3
      rm -f $ADDR_DIR/$INPUT3
    fi
else
    help_tunel
fi
}
```

Dalam fungsi diatas dapat dilihat bahwa untuk membangun koneksi *tunnel* dibutuhkan identifikasi alamat IPv4 lokal dan IPv4 tujuan agar terbentuk koneksi *point-to-point*. Untuk menambahkan *sit device* pada sistem, program memberi syarat sebagai berikut:

```
elif [[ $INPUT2 == add && $INPUT3 == sit? && \
     $INPUT4 == remote && $INPUT6 == local ]]
then
  ip tunnel add $INPUT3 mode sit remote $INPUT5 local \
  $INPUT7 2> $TMP/ceksit
  if [[ `cat $TMP/ceksit` == "" ]]
```

```
then
  UP="ip link set $INPUT3 up"
  $UP
  echo "$INPUT3 remote $INPUT5 local $INPUT7">$ADDR_DIR/$INPUT3
```

Syarat untuk menambahkan *device sit* pada sistem adalah dengan mendefinisikan nama sit? pada *device*, mendefinisikan alamat IPv4 tujuan dan mendefinisikan alamat IPv4 lokal. Konfigurasi *device*, alamat IPv4 tujuan dan alamat IPv4 lokal disimpan dalam file yang di beri nama sesuai dengan sit *device* yang dibuat.

```
elif [[ $INPUT2 == del && $INPUT3 ]]
then
    if [[ -e $ADDR_DIR/$INPUT3 ]]
    then
      DOWN="ip link set $INPUT3 down"
      ip tunnel del $INPUT3
      rm -f $ADDR_DIR/$INPUT3
    fi
```

Sedangkan untuk menghapus *sit* ini memiliki bagian untuk menghapus bagian *device* dari sistem dan bagian untuk menghapus file konfigurasi seperti bagian program diatas. Jadi ketika komputer mengalami *restart*, tidak ada konfigurasi *tunnel* yang ikut di *start*.

4.3.9. Fungsi Rute

Fungsi rute adalah fungsi yang digunakan untuk menambahkan tabel *routing* pada sistem agar koneksi terhadap jaringan IPv6 dapat terjadi. Fungsi rute ini dibagi menjadi beberapa bagian yaitu untuk mengaktifkan IPv6 *forwarding*, menonaktifkan IPv6 *forwarding*, menambah tabel *routing*, menghapus tabel *routing* dan melihat tabel *routing*. Bentuk kode program *routing* adalah sebagai berikut:

```
Rute()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_route
elif [[ $INPUT2 == up ]]
then
    echo "1" > /proc/sys/net/IPv6/conf/all/forwarding
    echo "1" > $ADDR_DIR/IPv6-forwarding
elif [[ $INPUT2 == down ]]
then
    echo "0" > /proc/sys/net/IPv6/conf/all/forwarding
    echo "0" > $ADDR_DIR/IPv6-forwarding
elif [[ $INPUT2 == add ]] && [[ $INPUT4 == via || $INPUT4 == gw
]]
then
    ip route add $INPUT3 via $INPUT5 2> $TMP/cek
    if [[ `cat $TMP/cek` == "" ]]
    then
      ip6 route show > $TMP/tur
      echo $INPUT3 > $TMP/addrrut
      sed 's/::...\//::\//' $TMP/addrrut > $TMP/addrbr
      sed 's/::..\//::\//' $TMP/addrbr > $TMP/inpbaru
      sed 's/::.\//::\//' $TMP/inpbaru > $TMP/addrrut
      tujuan=`cat $TMP/addrrut`
      cat /tmp/tur |grep $tujuan |grep $INPUT5 > $TMP/smpntur
      awk '{print $1, "via", $2}' $TMP/smpntur >> $TMP/unik
      uniq $TMP/unik > $TMP/antik
      cat $TMP/antik > $ADDR_DIR/IPv6-route6
    else
      echo "Perhatikan penulisan tabel routing!!!"
    fi
    rm -f $TMP/antik
    rm -f $TMP/unik
    rm -f $TMP/smpntur
    rm -f $TMP/inpbaru
    rm -f $TMP/addrbr
    rm -f $TMP/addrrut
    rm -f $TMP/tur
    rm -f $TMP/cek
elif [[ $INPUT2 == del ]] && [[ $INPUT3 == ::/0 || $INPUT3 ==
default ]]
then
    ip route  del ::/0
elif [[ $INPUT2 == del && $INPUT4 == via ]]
then
    ip route del $INPUT3 via $INPUT5 2> $TMP/cek
    if [[ `cat $TMP/cek` == "" ]]
    then
      echo "$INPUT3.via.$INPUT5" > $TMP/parsing
      sed 's/\//\\\//' $TMP/parsing > $TMP/hasilparsing
      hasil=`cat $TMP/hasilparsing`
      sed '/'$hasil'/d' $ADDR_DIR/IPv6-route6 > $TMP/route6
      cp --reply=yes $TMP/route6 $ADDR_DIR/IPv6-route6
      rm -f $TMP/route6
      rm -f $TMP/rutdel
```

```
        rm -f $TMP/hasilparsing
        rm -f $TMP/parsing
      else
        echo "Perhatikan penulisan penghapusan tabel routing!!!"
      fi
      rm -f $TMP/cek

elif [[ $INPUT2 == show || $INPUT2 == list || \
        $INPUT2 == print ]]
then
      route -A inet6
else
      help_route
fi
}
```

Pada fungsi *routing* ini, program dapat digunakan untuk mensetting IPv6 *forwarding* agar *router* dapat melewatkan paket IPv6. Konfigurasi keadaan *forwarding* ini juga digunakan untuk fungsi otomatis IPv6 *forwarding* agar ketika komputer mengalami *restart*, IPv6 *forwarding* juga langsung menyala. Fungsi ini juga dapat digunakan untuk mematikan kondisi *forwarding* IPv6. Berikut adalah bagian program untuk mengaktifkan dan menonaktifkan IPv6 *forwarding*.

```
elif [[ $INPUT2 == up ]]
then
    echo "1" > /proc/sys/net/IPv6/conf/all/forwarding
    echo "1" > $ADDR_DIR/IPv6-forwarding
elif [[ $INPUT2 == down ]]
then
    echo "0" > /proc/sys/net/IPv6/conf/all/forwarding
    echo "0" > $ADDR_DIR/IPv6-forwarding
```

Sedangkan bagian untuk menambahkan tabel *routing*, program akan melihat isi dari tabel *routing* yang ada. Jika *routing* tambahan belum ada pada tabel *routing*, maka penambahan *routing* ini dimasukkan dalam tabel *routing* dan kemudian disimpan dalam file agar ketika *network* atau komputer mengalami *restart*, tabel *routing* dapat ditambahkan secara otomatis. Berikut adalah potongan program untuk menambahkan tabel *routing* dan fungsi penyimpanannya dalam sebuah file.

```
elif [[ $INPUT2 == add ]] && [[ $INPUT4 == via || $INPUT4 == gw ]]
then
    ip route add $INPUT3 via $INPUT5 2> $TMP/cek
    if [[ `cat $TMP/cek` == "" ]]
    then
      ip6 route show > $TMP/tur
      echo $INPUT3 > $TMP/addrrut
      sed 's/::...\//::\//' $TMP/addrrut > $TMP/addrbr
      sed 's/::..\//::\//' $TMP/addrbr > $TMP/inpbaru
      sed 's/::.\//::\//' $TMP/inpbaru > $TMP/addrrut
      tujuan=`cat $TMP/addrrut`
      cat /tmp/tur |grep $tujuan |grep $INPUT5 > $TMP/smpntur
      awk '{print $1, "via", $2}' $TMP/smpntur >> $TMP/unik
      uniq $TMP/unik > $TMP/antik
      cat $TMP/antik > $ADDR_DIR/IPv6-route6
    else
      echo "Perhatikan penulisan tabel routing!!!"
    fi
```

Program ini menyediakan fasilitas deteksi kesalahan dalam memasukkan tabel *routing* tambahan. Perintah untuk mengecek kesalahan tersebut adalah:

```
ip route add $INPUT3 via $INPUT5 2> $TMP/cek
if [[ `cat $TMP/cek` == "" ]]
```

Jika perintah dalam memasukkan *routing* benar, maka isi dari file cek akan kosong dan program akan menambahkan tabel *routing* ke sistem dan menyimpan file konfigurasi ke dalam file. Bagian program tersebut adalah:

```
ip6 route show > $TMP/tur
echo $INPUT3 > $TMP/addrrut
sed 's/::...\//::\//' $TMP/addrrut > $TMP/addrbr
sed 's/::..\//::\//' $TMP/addrbr > $TMP/inpbaru
sed 's/::.\//::\//' $TMP/inpbaru > $TMP/addrrut
tujuan=`cat $TMP/addrrut`
cat /tmp/tur |grep $tujuan |grep $INPUT5 > $TMP/smpntur
awk '{print $1, "via", $2}' $TMP/smpntur >> $TMP/unik
uniq $TMP/unik > $TMP/antik
cat $TMP/antik > $ADDR_DIR/IPv6-route6
```

Dalam bagian program diatas terdapat perintah yang digunakan untuk memparsing input tabel *routing* yang disimpan kedalam file agar sama seperti tabel *routing* yang ada dalam sistem.

```
sed 's/::...\//::\//' $TMP/addrrut > $TMP/addrbr
sed 's/::..\//::\//' $TMP/addrbr > $TMP/inpbaru
sed 's/::.\//::\//' $TMP/inpbaru > $TMP/addrrut
```

Selanjutnya setelah mengalami proses *parsing*, hasil parsing tersebut akan dibentuk mejadi sebuah baris konfigurasi dan ditambahkan dalam file tabel *routing*.

4.3.10. Fungsi bantuan

Bagian ini berisi tentang bantuan-bantuan program ip6. Hal ini dimaksudkan untuk menampilkan bantuan dalam menseting IPv6 dengan menggunakan program ip6.

```
Bantuan()
{
echo "gunakan : ip6 OPSI {command | help}"
echo "          OPSI := {add | del | tunnel | route}"
echo "          command := {up | down | list | help}"
echo " "
}

help_tambah()
{
echo " gunakan : ip6 add [alamat IPv6] [gw [gateway]] [dev]"
echo "           address := alamat IPv6"
echo "           gateway := alamat gateway (optional)"
echo "           device  := interface yang akan diberi IPv6"
}

help_hapus()
{
echo " gunakan : ip6 del [ alamat IPv6 ] [ interface ]"
echo "           address := alamat IPv6 yang akan dihapus"
echo "           device  := interface yang akan dihapus IPv6-
nya"
}

help_tunel()
{
echo "gunakan : ip6 tunnel [cmd] [dev] remote [IPv4 remote] \
local [IPv4 local]"
echo "          cmd := { add | del }"
echo "          dev := sit1, sit2, dst."
echo "          IPv4 remote := alamat IPv4 tujuan tunnel"
```

```
echo "          IPv4 local := alamat IPv4 sumber tunnel"
}

help_route()
{
echo "gunakan : ip6 route [ command ] [ destination ] via [
device ]"
echo "          command := add | del | show"
echo "          destination := alamat tujuan routing"
echo "          via := routing dilewatkan melalui alamat ini"
echo "          device := routing akan melalui interface ini"
}
```

4.3.11. Fungsi utama

Fungsi utama ini adalah identifikasi terhadap opsi yang akan dijalankan oleh program ip6. Berikut adalah kode program fungsi utama:

```
case $INPUT1 in
    up|start)
      aktif;;
    down|stop)
      deaktif;;
    add)
      tambah;;
    del)
      hapus;;
    list|show)
      tampil;;
    tunnel)
      tunel;;
    route)
      rute;;
    help|-h|--help)
      bantuan;;
    *)
      bantuan;;
esac
```

4.4. Implementasi Service IPv6

Dalam implementasi service ini program dibagi menjadi dua bagian yaitu untuk *start* dan untuk *stop*. Namum untuk mengaktifkan program ini agar menjadi *services* atau *daemon*, maka program perlu inisialisasi *chkconfig*. Program ini

menjadi *daemon* pada proses 2345, diaktifkan pada run level 95 dan dimatikan pada run level 92. Berikut adalah inisialisasi run level *daemon* IPv6:

```
#!/bin/bash
# Script Startup untuk menjalankan IPv6
#
# chkconfig: 2345 95 92
# description: untuk menjalankan IPv6 pada saat boot up
```

Dalam *daemon* ini alamat direktori juga didefinisikan untuk menunjukkan lokasi alamat file konfigurasi. Alamat direktori yang didefinisikan dalam *daemon* ini hanya pada direktori *temporary* dan direktori untuk file konfigurasi IPv6.

```
TMP=/tmp
ADDR_DIR=/etc/sysconfig/IPv6
```

Di dalam program ini juga ditambahkan fungsi untuk membuat pesan OK pada saat *start* dan *stop daemon* IPv6. Berikut adalah kode program fungsi pesan OK.

```
RES=60
LOCATION="echo -en \\033[${RES}G"
COLOR="echo -en \\033[1;32m"
NORMAL="echo -en \\033[0;39m"
TIME=300000

OK_CODE()
{
usleep $TIME
$LOCATION
echo -n "[  "
$COLOR
echo -n $"OK"
$NORMAL
echo -n "  ]"
echo ""
}
```

4.4.1. Fungsi mulai

Pada *daemon* ini memiliki fungsi untuk mengaktifkan semua konfigurasi IPv6 yang telah disimpan dalam file konfigurasi. Dalam fungsi ini

memiliki beberapa bagian yang digunakan untuk mengaktifkan IPv6 ke dalam *device*, mengaktifkan dan menambahkan *tunnel*, mengisikan tabel *routing* IPv6 pada sistem menurut tabel *routing* IPv6 yang tersimpan dalam file konfigurasi, dan mengaktifkan IPv6 *forwarding*.

```
mulai()
{
echo -n "Aktifkan IPv6: "
# untuk mengaktifkan alamat IPv6 ke device
i=0
while :
do
    if [[ -e "$ADDR_DIR/ip6cfg-eth$i" ]]
    then
      cat $ADDR_DIR/ip6cfg-eth$i |grep address > $TMP/addr
        awk '{print $2}' $TMP/addr > $TMP/bot6
      ip -f inet6 addr add `cat $TMP/bot6` dev "eth$i"
        rm -f $TMP/addr
        rm -f $TMP/bot6
# jika ditemukan gateway, maka gateway akan di aktifkan
      cat $ADDR_DIR/ip6cfg-eth$i |grep gateway > $TMP/gate
        gerbang=`awk '{print $2}' $TMP/gate`
        if [[ $gerbang != "" ]]
        then
          ip route add ::/0 via $gerbang
        fi
    fi
    if [[ ($i > 5) ]]
    then
        break
    fi
rm -f $TMP/gate
let i+=1
done
# aktifasi tunnel otomatis
q=0
while (( $q <= 5 ))
do
    let q+=1
    if [[ -e "$ADDR_DIR/sit$q" && -e "$ADDR_DIR/ip6cfg-sit$q" ]]
    then
      awk '{print $1," mode sit ",$2, $3, $4, $5, $6}'\
      "$ADDR_DIR/sit$q" > $TMP/situp
      siter=`cat $TMP/situp`
      ip tunnel add $siter
      UP="ip link set sit$q up"
      $UP
      ip addr add `cat $ADDR_DIR/ip6cfg-sit$q`
    fi
done
rm -f $TMP/situp
```

```
# digunakan untuk menghitung jumlah tabel routing
if [[ -e "$ADDR_DIR/IPv6-route6" ]]
then
    cat $ADDR_DIR/IPv6-route6 | wc -l > $TMP/counter6
    awk '{print $1}' $TMP/counter6 > $TMP/count6
# digunakan untuk mengaktifkan tabel routing
    if [[ `cat "$ADDR_DIR/IPv6-route6"` != "" ]]
    then
      awk '{print "rut" NR, $1, $2, $3, $4, $5}'\
          $ADDR_DIR/IPv6-route6 > $TMP/route6
      p=0
      while :
        do
          let p+=1
          awk '/rut'$p'/{print $2, $3, $4, $5, $6 }' \
            $TMP/route6 > $TMP/rut6
          ip route add `cat $TMP/rut6`
          if [[ $p == `cat $TMP/count6` ]]
          then
            break
          fi
      done
    rm -f $TMP/route6
    rm -f $TMP/rut6
    fi
fi
rm -f $TMP/count6
rm -f $TMP/counter6

# digunakan untuk forwarding otomatis
if [[ `cat $ADDR_DIR/IPv6-forwarding` == 1 ]]
then
    echo "1" > /proc/sys/net/IPv6/conf/all/forwarding
fi
# tampilkan tanda [ OK ]
OK_CODE
}
```

Fungsi mengaktifkan alamat IPv6 pada *interface* fisik pada *daemon* IPv6 ini adalah sebagai berikut:

```
# untuk mengaktifkan alamat IPv6 ke device
i=0
while :
do
    if [[ -e "$ADDR_DIR/ip6cfg-eth$i" ]]
    then
      cat $ADDR_DIR/ip6cfg-eth$i |grep address > $TMP/addr
        awk '{print $2}' $TMP/addr > $TMP/bot6
      ip -f inet6 addr add `cat $TMP/bot6` dev "eth$i"
        rm -f $TMP/addr
        rm -f $TMP/bot6
```

```
# jika ditemukan gateway, maka gateway akan di aktifkan
      cat $ADDR_DIR/ip6cfg-eth$i |grep gateway > $TMP/gate
        gerbang=`awk '{print $2}' $TMP/gate`
        if [[ $gerbang != "" ]]
        then
          ip route add ::/0 via $gerbang
        fi
    fi
    if [[ ($i > 5) ]]
    then
        break
    fi
rm -f $TMP/gate
let i+=1
done
```

Bagian ini adalah perulangan yang digunakan untuk mendeteksi file konfigurasi alamat IPv6 terhadap *device interface* tertentu. Jika ditemukan nama file konfigurasi alamat ip6cfg-eth$i, maka alamat IPv6 yang tersimpan dalam file tersebut akan dimasukkan dalam *interface* eth$i. Deteksi ada tidaknya file sampai bagian untuk menambahkan alamat IPv6 ke dalam *interface* tersebut adalah sebagai berikut:

```
if [[ -e "$ADDR_DIR/ip6cfg-eth$i" ]]
then
  cat $ADDR_DIR/ip6cfg-eth$i |grep address > $TMP/addr
    awk '{print $2}' $TMP/addr > $TMP/bot6
  ip -f inet6 addr add `cat $TMP/bot6` dev "eth$i"
    rm -f $TMP/addr
    rm -f $TMP/bot6
```

Jika ternyata dalam file tersebut juga ditemukan konfigurasi untuk *gateway*, maka program juga akan menambahkan *routing* sebagai *default gateway* dengan alamat *gateway* yang berada dalam file konfigurasi tersebut.

```
cat $ADDR_DIR/ip6cfg-eth$i |grep gateway > $TMP/gate
  gerbang=`awk '{print $2}' $TMP/gate`
  if [[ $gerbang != "" ]]
  then
    ip route add ::/0 via $gerbang
  fi
```

Konfigurasi penambahan *device tunnel* otomatis dilakukan sebagai upaya untuk membangun koneksi *point-to-point* dalam mekanisme *tunneling.* Konfigurasi *tunnel* juga telah disimpan dalam file sehingga ketika program menjalankan fungsi deteksi *tunnel*, program dapat mengetahui apakah sistem memiliki fungsi untuk koneksi *tunneling* atau tidak. Dalam fungsi ini terdapat dua fungsi utama untuk membangun mekanisme *tunneling* yaitu membangun koneksi *point-to-point* antar jaringan IPv4 kemudian menambahkan alamat IPv6 kedalam *sit device.*

```
q=0
while (( $q <= 5 ))
do
    let q+=1
    if [[ -e "$ADDR_DIR/sit$q" && -e "$ADDR_DIR/ip6cfg-sit$q" ]]
    then
      awk '{print $1," mode sit ",$2, $3, $4, $5, $6}'\
      "$ADDR_DIR/sit$q" > $TMP/situp
      siter=`cat $TMP/situp`
      ip tunnel add $siter
      UP="ip link set sit$q up"
      $UP
      ip addr add `cat $ADDR_DIR/ip6cfg-sit$q`
    fi
done
rm -f $TMP/situp
```

Dari kedua fungsi penambahan alamat diatas baik fungsi otomatis penambahan IPv6 kedalam *interface* fisik maupun fungsi otomatis *device ineterface* logika hanya dibatasi sampai dengan lima buah *interface.* Hal ini pengembang membatasi penggunaan *interface* fisik maupun *sit device* maksimal sebanyak lima buah.

Penambahan tabel *routing* IPv6 pada *daemon* ini tergantung dari tabel yang tersimpan dalam file konfigurasi tabel *routing* IPv6. Konfigurasi *routing*

IPv6 tersimpan dalam file IPv6-route6, dan pengembang tidak membatasi jumlah tabel *routing* yang akan ditambahkan.

```
if [[ -e "$ADDR_DIR/IPv6-route6" ]]
then
    cat $ADDR_DIR/IPv6-route6 | wc -l > $TMP/counter6
    awk '{print $1}' $TMP/counter6 > $TMP/count6
# digunakan untuk mengaktifkan tabel routing
    if [[ `cat "$ADDR_DIR/IPv6-route6"` != "" ]]
    then
      awk '{print "rut" NR, $1, $2, $3, $4, $5}'\
          $ADDR_DIR/IPv6-route6 > $TMP/route6
      p=0
      while :
        do
          let p+=1
          awk '/rut'$p'/{print $2, $3, $4, $5, $6 }' \
            $TMP/route6 > $TMP/rut6
          ip route add `cat $TMP/rut6`
          if [[ $p == `cat $TMP/count6` ]]
          then
            break
          fi
      done
    rm -f $TMP/route6
    rm -f $TMP/rut6
    fi
fi
rm -f $TMP/count6
rm -f $TMP/counter6
```

Potongan program diatas adalah bagian untuk menghitung banyaknya tabel *routing* IPv6 yang ada pada file konfigurasi *routing* IPv6 dan yang akan ditambahkan ke dalam sistem.

Sedangkan bagian yang digunakan untuk mendeteksi IPv6 *forwarding* adalah dengan melihat status IPv6 *forwarding* dan membandingkan apakah status *forwading* IPv6 bernilai satu atau tidak. Jika status bernilai satu maka *forwarding* IPv6 juga akan di *enable* secara otomatis. Berikut adalah potongan program untuk *enable forwarding* IPv6.

```
if [[ `cat $ADDR_DIR/IPv6-forwarding` == 1 ]]
then
    echo "1" > /proc/sys/net/IPv6/conf/all/forwarding
fi
```

4.4.2. Fungsi selesai

Fungsi ini adalah fungsi untuk menghapus tabel *routing* dari sistem, menghapus *sit device* dan me-*reset interface* fisik agar alamat IPv6 hilang dari *interface*. Berikut adalah kode program fungsi selesai:

```
selesai()
{
echo -n "Nonaktifkan IPv6: "
# hapus routing tabelnya dulu
ip6 route show > $TMP/ruteasli
cat $TMP/ruteasli |wc -l > $TMP/jmlrut
awk '{print $1}' $TMP/jmlrut > $TMP/ttlrut
awk '{print "rut" NR, $1}' $TMP/ruteasli > $TMP/hpsrut
p=2
while :
do
    let p+=1
    awk '/rut'$p'/{print $2}' $TMP/hpsrut > $TMP/del
    ip route del `cat $TMP/del`
    if [[ $p == `cat $TMP/ttlrut` ]]
    then
      break
    fi
done
rm -f $TMP/ruteasli
rm -f $TMP/jmlrut
rm -f $TMP/ttlrut
rm -f $TMP/hpsrut
rm -f $TMP/del

# hapus sit device
q=5
while (( $q >=1 ))
do
    ip6 list |grep sit$q > $TMP/ceksit
    if [[ `cat $TMP/ceksit` != "" ]]
    then
      ip tunnel del sit$q
    fi
let q-=1
done
rm -f $TMP/ceksit

# mereset interface eth0
i=5
while (( $i >= 0 ))
do
    if [ -e "$ADDR_DIR/ip6cfg-eth$i" ]
    then
      ifconfig eth$i down
```

```
      ifconfig eth$i up
    fi
    if [[ $i == 0 ]]
    then
      break
    fi
let i-=1
done

# tampilkan [ OK ]
OK_CODE
}
```

Pada bagian awal fungsi selesai ini adalah program untuk menghapus seluruh tabel *routing* IPv6 yang telah tertanam di sistem. Proses penghapusan tabel *routing* IPv6 ini adalah dengan melihat terlebih dahulu seluruh tabel *routing* IPv6 yang ada dalam sistem kemudian menyimpannya kedalam file, dan setelah melalui proses parsing terhadap alamat tujuan *routing* barulah tabel di hapus satu per satu.

```
ip6 route show > $TMP/ruteasli
cat $TMP/ruteasli |wc -l > $TMP/jmlrut
awk '{print $1}' $TMP/jmlrut > $TMP/ttlrut
awk '{print "rut" NR, $1}' $TMP/ruteasli > $TMP/hpsrut
p=2
while :
do
    let p+=1
    awk '/rut'$p'/{print $2}' $TMP/hpsrut > $TMP/del
    ip route del `cat $TMP/del`
    if [[ $p == `cat $TMP/ttlrut` ]]
    then
      break
    fi
done
rm -f $TMP/ruteasli
rm -f $TMP/jmlrut
rm -f $TMP/ttlrut
rm -f $TMP/hpsrut
rm -f $TMP/del
```

Setelah selesai proses penghapusan tabel *routing* IPv6, maka dilakukan penghapusan *sit device* dari sistem. Dalam proses ini ketika *sit device* dihapus, seluruh alamat baik definisi alamat *tunnel* dan alamat IPv6 *tunnel* didalam *sit*

device ini akan ikut terhapus. Oleh karena itu, proses ini hanya melakukan test apakah device *sit?* ada didalam sistem atau tidak. Jika ditemukan, maka *device* tersebut akan di hapus dari sistem.

```
q=5
while (( $q >=1 ))
do
    ip6 list |grep sit$q > $TMP/ceksit
    if [[ `cat $TMP/ceksit` != "" ]]
    then
      ip tunnel del sit$q
    fi
let q-=1
done
rm -f $TMP/ceksit
```

Proses terakhir dari fungsi selesai ini adalah mereset *interface* fisik. Hal ini digunakan untuk menghapus alamat IPv6 dari *interface* dalam sistem. Karena program ip tidak dapat menangani uninstall modul *interface* maka pengembang menggunakan program bantu lain yang dapat menguninstall modul *interface* yaitu program *ifconfig*. Dengan mereset *interface* ini, seluruh konfigurasi IPv6 pada *interface* akan hilang dari sistem.

```
i=5
while (( $i >= 0 ))
do
    if [ -e "$ADDR_DIR/ip6cfg-eth$i" ]
    then
      ifconfig eth$i down
      ifconfig eth$i up
    fi
    if [[ $i == 0 ]]
    then
      break
    fi
let i-=1
done
```

Seperti halnya pada fungsi start, di dalam fungsi ini pengembang juga membatasi jumlah *interface* yang di *restart* dan *sit device* yang di hapus.

Pengembang hanya mengijinkan penggunaan *interface* maupun *sit device* sebanyak maksimal lima buah.

4.4.3. Fungsi utama

Fungsi utama ini adalah identifikasi terhadap opsi yang dapat dijalankan oleh *daemon* IPv6. Opsi tersebut adalah *start*, *stop* dan *restart*. Berikut adalah kode program fungsi utama:

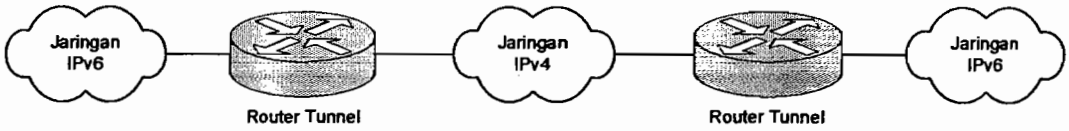
```
case "$1" in
    start)
      mulai ;;
    stop)
      selesai ;;
    restart)
      # proses yang dipakai adalah stop kemudian start
      selesai
      mulai;;
    *)
      echo "Gunakan : $0 { start | stop | restart }"
      exit 1
esac
exit 0
```

4.5. Implementasi Jaringan

Bagian ini menjelaskan iplementasi jaringan IPv6 terhadap fungsi *tunnel* yang dibangun pada jaringan IPv4.

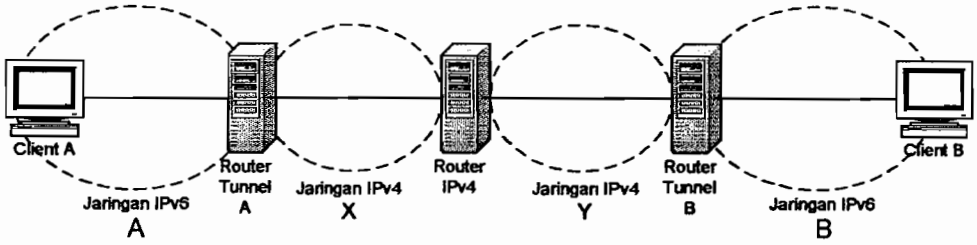
4.5.1. Implementasi jaringan fisik

Implementasi jaringan adalah mencoba mekanisme *tunneling* jaringan IPv6 melalui jaringan IPv4. Desain logika implementasi yang dibuat adalah sebagai berikut:

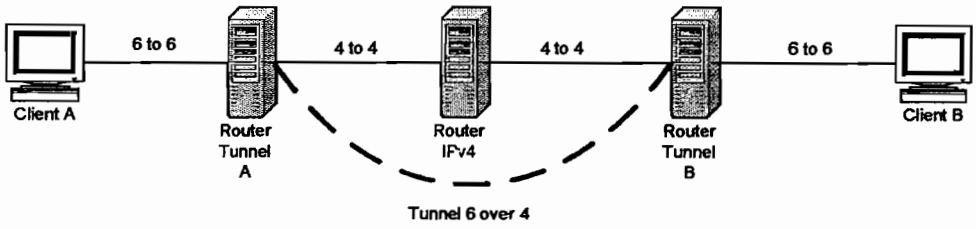


Gambar 4.1. Implementasi secara logika

Dengan menggunakan desain ini, pengembang mengimplementasikan desain ini kedalam bentuk simulasi jaringan komputer dengan memanfaatkan 5 buah komputer yang tersusun menurut bentuk desain yang ada. Bentuk desain jaringan fisik yang digunakan adalah sebagai berikut:



Gambar 4.2. Implementasi secara fisik



Gambar 4.3. Koneksi tunnel

Dalam kasus ini setiap segmen jaringan memiliki alokasi alamat yang berbeda-beda. Asumsi alokasi alamat setiap segmen jaringan adalah sebagai berikut:

a. Jaringan A : 2002:a122::/96

b. Jaringan B : 2002:b201::/96

c. Jaringan X : 202.158.142.0/29

d. Jaringan Y : 202.169.21.0/29

e. Jaringan tunnel : 2001:f1a::/120

Dengan alokasi alamat seperti diatas, maka tiap tiap komputer akan diberi alamat sesuai dengan segmen jaringannya. Spesifikasi pengalamatan pada masing-masing komputer adalah sebagai berikut:

a. Client A

Alamat IPv6 : 2002:a122::2/96

Gateway IPv6 : Router tunnel A

b. Router Tunnel A

Alamat IPv6 : 2002:a122::1/96

Alamat tunnel : 2002:f1a::1/120

Alamat IPv4 : 202.158.142.1/29

Gateway IPv4 : Router IPv4

c. Router IPv4

Alamat IPv4 : 202.158.142.2/29

202.169.21.2/29

d. Router Tunnel B

Alamat IPv6 : 2002:b201::1/96

Alamat tunnel :2002:f1a::2/120

Alamat IPv4 : 202.169.21.1/29

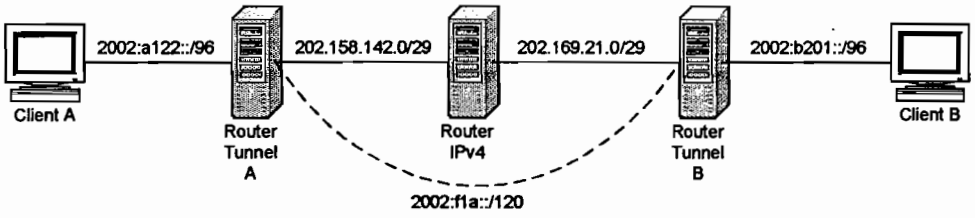
Gateway IPv4 : Router IPv4

e. Client B

Alamat IPv6 : 2002:b201::2/96

Gateway : Router tunnel B

Sehingga konfigurasi alamat setiap segmen pada jaringan akan berbentuk seperti berikut:



Gambar 4.4. Implementasi alamat ip pada setiap segmen

Dalam jaringan seperti ini mekanisme *tunneling* belum terbentuk. Oleh karena itu pada *router tunnel* A ditambah identifikasi *tunneling* dan alamat IPv6 untuk membuat jalur *tunnel* yang seolah-olah menghubungkan langsung antara *router tunnel* A dengan *router tunnel* B.

4.5.2. Penggunaan Program ip6

Program ip6 digunakan untuk mengisikan alamat IPv6 dan setting IPv6 lainnya pada masing-masing komputer yang menggunakan protokol IPv6. Pada implementasi ke dalam jaringan, komputer yang menggunakan program ip6 adalah komputer *client* A, *router tunnel* A, *router tunnel* B dan *client* B.

4.5.2.1. Client A

Pada *client* A setting IPv6 yang digunakan adalah menginstall modul IPv6, menambahkan alamat IPv6 pada *interface* dan menambahkan *default gateway*. Perintah yang digunakan adalah:

```
# ip6 up
# ip6 add 2002:a122::2/96 gw 2002:a122::1 eth0
```

Untuk mengetahui apakah alamat IPv6 sudah terinstall pada *interface* eth0 dapat di cek dengan menggunakan perintah ip6 list. Sedangkan untuk mengetahui *default gateway* dapat dilihat dengan perintah ip6 route show

```
# ip6 list
# ip6 route show
```

4.5.2.2. Router Tunnel A

Interface pada *router tunnel* A yang terhubung dengan jaringan *client* A adalah eth0. Jadi *interface* eth0 hanya mempunyai konfigurasi alamat IPv6 saja. Sehingga jaringan yang terbentuk dalam segmen ini adalah murni jaringan yang menggunakan protokol IPv6. *Interface* eth1 disetting dengan IPv4 dan terhubung dengan jaringan *router* IPv4. Dalam implementasi ini diasumsikan bahwa jaringan IPv4 dari *router tunnel* A sampai *router tunnel* B sudah terbentuk dengan konfigurasi seperti gambar dan penjelasan diatas. *Interface* eth0 pada *router tunnel* A dikonfigurasi dengan menambahkan alamat IPv6 2002:a122::1/96.

```
# ip6 up
# ip6 add 2002:a122::1/96 eth0
```

Setelah setting alamat IPv6 terhadap *interface* eth0 pada *router tunnel* A, langkah selanjutnya adalah melihat apakah alamat IPv6 telah tertanam pada *interface* eth0 atau tidak, dengan mengetikan perintah ip6 list. Jika alamat IPv6 sudah benar maka langkah selanjutnya adalah melakukan test koneksi antara *client* A dengan *router tunnel* A dengan perintah ping6 dari *client* A dengan tujuan *router tunnel* A, atau sebaliknya. Berikut adalah perintah test koneksi dari *router tunnel* A.

```
# ping6 2002:a122::2
```

Jika dari test koneksi dengan perintah ping6 ini mendapat status *reply*, maka koneksi antara *client* A dan *router tunnel* A sudah benar. Langkah selanjutnya adalah dengan membuat koneksi *tunnel* antara *router tunnel* A dengan *router tunnel* B dan menambahkan alamat IPv6 pada *device tunnel*. Perintah yang digunakan untuk membangun koneksi ini adalah sebagai berikut:

```
# ip6 tunnel add sit1 remote 202.169.21.1 local 202.158.142.1
# ip6 add 2001:f1a::1/120 sit1
```

Untuk melihat apakah alamat IPv6 sudah tertanam pada *device sit1* atau belum dapat dilakukan dengan mengetik perintah ip6 list sit1. Jika *device* dan alamat IPv6 sudah ada, maka *device tunnel* sudah terbuat dan sudah siap. Untuk mengaktifkan *router* agar dapat meneruskan paket IPv6 maka fungsi *forwarding router* harus diaktifkan dengan perintah berikut:

```
# ip6 route up
```

Untuk memeriksa fungsi *forwarding* IPv6 pada *router tunnel* A adalah dengan melakukan ping6 ke *client* A dari alamat *sit device*. Perintah yang digunakan adalah:

```
# ping6 2002:a122::2 -I 2001:f1a::1
```

Perintah diatas dapat diartikan sebagai "lakukan ping6 ke alamat tujuan 2002:a122::2 melalui *interface* yang memiliki alamat 2001:f1a::1".

4.5.2.3. Client B

Pada client B cara setting IPv6 sama seperti pada *client* A yaitu dengan menambahkan alamat IPv6 pada *interface* dan menambahkan *default gateway*. Perintah yang digunakan adalah:

```
# ip6 up
# ip6 add 2002:b201::2/96 gw 2002:b201::1 eth0
```

Untuk mengetahui apakah alamat IPv6 sudah terinstall pada *interface* eth0 dapat di cek dengan menggunakan perintah ip6 list. Sedangkan untuk mengetahui *default gateway* dapat dilihat dengan perintah ip6 route show

```
# ip6 list
# ip6 route show
```

4.5.2.4. Router Tunnel B

Prinsip yang digunakan pada *router tunnel* B sama dengan *router tunnel* A. *Interface* pada *router tunnel* B yang terhubung dengan jaringan *client* B adalah eth0. *Interface* eth0 pada *router tunnel* B dikonfigurasi dengan menambahkan alamat IPv6 2002:b201::1/96.

```
# ip6 up
# ip6 add 2002:b201::1/96 eth0
```

Langkah selanjutnya adalah melakukan test koneksi antara *client* B dengan *router tunnel* B dengan perintah ping6 dari *client* B dengan tujuan *router tunnel* B, atau sebaliknya. Berikut adalah perintah test koneksi dari *router tunnel* B.

```
# ping6 2002:b201::2
```

Jika dari test koneksi dengan perintah ping6 ini mendapat status *reply*, maka koneksi antara *client* B dan *router tunnel* B sudah benar. Langkah

selanjutnya adalah dengan membuat koneksi *tunnel* antara *router tunnel* B dengan *router tunnel* A dan menambahkan alamat IPv6 pada *device tunnel*. Perintah yang digunakan untuk membangun koneksi ini adalah sebagai berikut:

```
# ip6 tunnel add sit1 remote 202.158.142.1 local 202.169.21.1
# ip6 add 2001:f1a::2/120 sit1
```

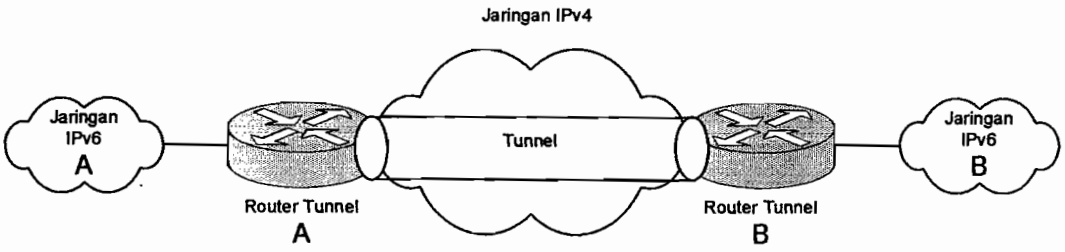
Sama seperti *router tunnel* A, fungsi *forwarding router* ini juga harus diaktifkan dengan perintah:

```
# ip6 route up
```

Untuk memeriksa fungsi *forwarding* IPv6 pada *router tunnel* B adalah dengan melakukan ping6 ke *client* B dari alamat *sit device*. Perintah yang digunakan adalah:

```
# ping6 2002:b201::2 -I 2001:f1a::2
```

Jika mendapat *reply*, maka fungsi *forwading* pada *router* ini sudah aktif. Melakukan ping6 dari *sit device router tunnel* A dengan tujuan *sit device router tunnel* B adalah cara untuk memastikan koneksi *tunnel* yang sudah dibuat. Jika perintah ini mendapat *reply*, maka koneksi *tunnel* sudah terbentuk.



Gambar 4.5. Jaringan dengan koneksi tunnel

Dengan adanya koneksi *tunnel* seolah-olah dari *router tunnel* A adalah jaringan satu segmen dengan *router tunnel* B. Oleh karena itu agar *router tunnel* A dapat mengenali alamat tujuan paket pada jaringan IPv6 B, maka pada *router*

tunnel A ditambah tabel *routing* yang tujuan alamat jaringan IPv6 B akan dilewatkan alamat *sit device router tunnel* B.

```
# ip6 route add 2002:b201::/96 via 2001:f1a::2
```

Demikian pula pada *router tunnel* B agar dapat mengenali jaringan IPv6 A, maka para *router tunnel* B juga ditambah tabel *routing* tujuan jaringan IPv6 A akan dilewatkan alamat *sit device router tunnel* A.

```
# ip6 route add 2002:a122::/96 via 2001:f1a::1
```

Kemudian untuk mengetahui hasil koneksi *tunneling* ini dapat dilakukan perintah ping6 dari *client* A dengan tujuan *client* B. Jika koneksi sudah benar, maka *client* A akan mendapat *reply* dari *client* B. Untuk memastikan paket yang terkirim dari client A telah mengalami proses *tunnel* di *router* tunnel A, di *router* IPv4 dapat dijalankan program *tcpdump* yang akan melihat paket-paket yang lewat.

```
# ping6 2002:b201::2
```

Untuk menguji lebih jauh lagi, aplikasi ssh dapat dijalankan dari sisi *client* A dengan tujuan *client* B atau sebaliknya.

```
# ssh -l root 2002:b201::2
```

Jika client B difungsikan sebagai web server dengan server berupa apache 2.0 dan *client* A memiliki *xwindows* maka pengujian koneksi dapat dilakukan dengan melakukan *browsing* terhadap *web server*. Pada *web browser*, ketikkan perintah:

```
http://[2002:b201::2]
```

Jika web server sudah support dengan IPv6, maka tampilan web akan muncul pada web browser.

BAB V

ANALISIS HASIL IMPLEMENTASI

Pada bab ini akan dibahas mengenai analisis jaringan IPv6 dan penggunaan piranti-piranti yang ada dalam bahasa permograman Bash.

5.1. Analisis Jaringan IPv6

Pada bagian ini akan dibahas mengenai efektivitas dan efisiensi penggunaan metode yang digunakan dalam membangun atau mengembangkan jaringan IPv6 dalam masa transisi.

Metode yang digunakan dalam implementasi jaringan IPv6 ini adalah metode *tunneling*, dimana metode ini memiliki tahapan-tahapan, yaitu:

1. Pengumpulan kebutuhan
2. Analisis sistem
3. Perancangan
4. Implementasi

Dengan metode ini pengembang dapat membangun sebuah jaringan IPv6 yang dilewatkan pada jaringan IPv4, karena memiliki tahapan-tahapan yang terstruktur, sehingga dapat diikuti oleh pengembang. Dengan mengikuti tahapan-tahapan tersebut pengembang dapat merancang jaringan IPv6 dari hasil kebutuhan yang terkumpul.

5.2. Analisis Bahasa Pemrograman

Bahasa pemrograman Bash dapat digunakan untuk membangun aplikasi dan *daemon* yang dapat digunakan untuk otomatisasi setting IPv6 pada jaringan, karena:

1. Bash merupakan bahasa pemrograman yang sudah *built-in* di dalam sistem operasi linux sehingga bahasa pemrograman Bash sangat kompatibel dengan Linux.
2. Bash dapat digunakan untuk memanggil dan menjalankan aplikasi-aplikasi yang ada dalam linux, sehingga cakupan tools yang dapat dibuat dengan bahasa pemrograman Bash lebih luas.
3. Fasilitas dan fungsi yang dimiliki oleh bahasa pemrograman ini sudah cukup lengkap. Dalam pembuatan tools ini pengembang diharuskan memparsing data yang ada agar dapat digunakan untuk otomatisasi setting IPv6 dan kemampuan memparsing data telah ada dalam Bash sehingga mempermudah pengembang dalam membuat tools.

5.3. Hasil Uji Coba Program dan Jaringan

Setelah merancang sistem dan mengimplementasikan program, maka penulis mengadakan uji coba sehubungan dengan mekanisme *tunneling* jaringan IPv6 melalui jaringan IPv4. Uji coba tersebut sangat berguna bagi penulis dan pengembang selanjutnya.

Penulis melakukan uji coba mekanisme *tunneling* jaringan IPv6 melalui jaringan IPv6 pada lingkup lokal. Hasil yang penulis dapatkan tidak diukur dalam

besaran tertentu karena belum ada alat ukur (*benchmark*) yang dapat membandingkan hasil *koneksi* antara jaringan IPv4 dengan IPv6. Namun dalam uji coba, penulis mendapatkan beberapa hasil yang dapat ditunjukkan untuk menilai kinerja mekanisme *tunneling*. Berikut ini hasil yang penulis dapatkan berdasarkan hasil uji coba yang telah dilakukan adalah sebagai berikut:

1. Kinerja *tunneling* terpengaruh oleh konektivitas jaringan IPv4 yang dilaluinya.
2. Koneksi *tunneling* tidak begitu stabil. Hal ini ditunjukkan dengan adanya pesan "*destination unreacheble*" secara terus menerus pada client ketika melakukan ping6 ke *client* lain yang beda segmen dan melewati jalur *tunnel* ketika *router tunnel* sempat mengalami restart. Meskipun network pada *router tunnel* sudah stabil kembali, terkadang *client* masih menampilkan pesan ini sampai user meng-*cancel* ping6 pada *client*. Setelah menunggu beberapa saat, dengan menjalankan kembali perintah ping6, baru akan didapat pesan *reply* pada *client*.
3. Mekanisme *tunneling* berfungsi membentuk *virtual private network*.
4. Program dapat melakukan fungsi otomatis terhadap setting IPv6.

5.4. Kelebihan dan kekurangan program

Setelah melaukukan uji coba tools ke dalam model jaringan yang dibuat, penulis memperoleh beberapa kelebihan dan kekurangan dari tools yang dibuat.

5.4.1. Kelebihan

Program untuk otomatisasi setting IPv6 ini mempunyai beberapa kelebihan, antara lain:

1. Mampu menyisip kedalam fungsi *daemon* network. Ketika *daemon* network di *restart*, program ini juga akan me-*restart* seluruh setting IPv6.
2. Konfigurasi IPv6 dapat diubah dengan mengubah isi pada file konfigurasi.
3. Menyediakan sebuah *daemon* tersendiri terhadap otomatisasi setting IPv6.

5.4.2. Kekurangan

Dan program yang telah dibuat, mempunyai beberapa kelemahan, antara lain:

1. Program masih belum dapat mengantisipasi duplikasi alamat IPv6 dalam satu interface.
2. Program hanya dapat menginstall modul IPv6 tetapi tidak dapat menguninstall modul IPv6. Untuk mengunistall modul IPv6, komputer harus di *reboot*.

BAB VI

PENUTUP

Pada bagian akhir dari penulisan skripsi ini dicantumkan beberapa kesimpulan dan saran dari hal-hal yang terkait dengan implementasi jaringan IPv6 dan pengembangan tools IPv6 berdasarkan komposisinya dari bab-bab sebelumnya.

6.1. Kesimpulan

Dari implementasi jaringan IPv6 dan pembuatan tools IPv6 dapat disimpulkan:

1. Jaringan IPv6 telah dapat diimplementasikan untuk menggantikan jaringan IPv4. Namun dalam penerapannya akan membutuhkan banyak perubahan terhadap sistem operasi agar mendukung IPv6.
2. Dua jaringan IPv6 yang tepisah jarak yang cukup jauh dan hanya memiliki koneksi IPv4 ke internet, dapat saling berhubungan antar jaringan dengan memanfaatkan mekanisme tunneling terhadap IPv4.
3. Sistem Operasi Linux telah mendukung setting dan koneksi dengan menggunakan protokol IPv6, tetapi dengan menggunakan syntax yang panjang.

6.2. Saran

Saran yang dapat diberikan penulis dalam pengembangan tools IPv6 lebih lanjut:

1. Dalam tools IPv6 ini, file-file konfigurasi sudah tersimpan dalam satu direktori. Namun struktur isi file konfigurasi masih belum tertata secara rapi, maka penulis menyarankan untuk merapikan penulisan ini konfigurasi ke dalam file.
2. Tools IPv6 ini hanya dapat digunakan untuk mekanisme *tunneling* saja, maka penulis menyarankan untuk mengembangkan tools agar dapat digunakan untuk mekanisme *translasi*.

DAFTAR PUSTAKA

Yuliardi, Rofiq, "*Bash Scripting untuk Administrasi Sistem Linux*", Elex Media Komputindo, Jakarta 2003.

Susanto, Budi, "*Unix & Pemrograman Script*", J&J Learning, Yogyakarta 2001.

Syukri, Muhammad, "*PC Router dengan GNU/Linux*", Elex Media Komputindo, Jakarta 2003.

Wijaya, Hendra, "*Cisco Router*", Elex Media Komputindo, Jakarta 2002.

Wijaya, Hendra, "*Cisco Switch*", Elex Media Komputindo, Jakarta 2003.

Arifin, Zaenal, "*Langkah Mudah Mengkonfigurasi Router Cisco*", Penerbit Andi, Yogyakarta 2003.

Wahana, "*Panduan Lengkap Pengembangan Jaringan Linux*", Penerbit Andi, Yogyakarta 2003.

Wahana, "*Mari Mengenal Linux*", Penerbit Andi, Yogyakarta 2001.

Stallings, William, "*Komunikasi Data dan Komputer Jaringan Komputer*", Salemba Teknika, 2000.

http://en.wikipedia.org/wiki

http://www.yale.edu

ftp://ftp.isi.edu/in-notes/rfc2373.txt

LAMPIRAN

Program IP6

```
#!/bin/bash
#
# tools untuk mensetting ipv6 dalam mesin linux
#
# oleh
#
#   antonius arief widiyatmoko
#     (ant_arief@yahoo.com)
#
#

# inisialisasi lokasi alamat ipv6
ADDR_MOD=/etc/sysconfig
ADDR_DIR=/etc/sysconfig/ipv6
ADDR_NET=/etc/init.d
TMP=/tmp

# inisialisasi passing parameter
INPUT1=$1
INPUT2=$2
INPUT3=$3
INPUT4=$4
INPUT5=$5
INPUT6=$6
INPUT7=$7


# fungsi untuk mengaktifkan otomatis modul ipv6
aktif()
{
# jika direktori /etc/sysconfig/ipv6 ada
if [[ -d $ADDR_DIR ]]
then
# cek apakah sudah aktif atau belum
    if [[ -a $ADDR_DIR/ipv6 && `cat $ADDR_DIR/ipv6` == 1 ]]
    then
      echo "Modul IPv6 dalam kondisi otomatis"
    else
# jika belum
# Mengaktifkan fungsi otomatis IPv6 dalam sistem
      echo "1" > $ADDR_DIR/ipv6
      modprobe ipv6
      sed '/NETWORKING/a NETWORKING_IPV6=yes' \
      $ADDR_MOD/network > $TMP/temp6
      cp --reply=yes $TMP/temp6 $ADDR_MOD/network
      rm -f $TMP/temp6

# Menyisipkan fungsi IPv6 ke dalam fungsi service network
# agar IPv6 dapat diangkat secara otomatis
      sed '/touch/a #aktifkan ipv6' $ADDR_NET/network > $TMP/net
```

```
        sed '/#aktifkan/a service ipv6 start' $TMP/net > $TMP/net1
        sed '/# and/a #deaktif ipv6' $TMP/net1 > $TMP/net
        sed '/#deaktif/a service ipv6 stop' $TMP/net > $TMP/net1
        cp --reply=yes $TMP/net1 $ADDR_NET/network
        rm -f $TMP/net
          rm -f $TMP/net1
    fi
else
# jika folder ipv6 tidak ada, maka buat dulu
    mkdir $ADDR_MOD/ipv6
    aktif
fi
}

# fungsi untuk menonaktifkan otomatis modul ipv6
deaktif()
{
# cek kondisi ipv6 apakah sudah dalam kondisi off
if [[ `cat $ADDR_DIR/ipv6` == 0 ]]
then
    echo "Modul IPv6 dalam kondisi manual"
else
# Menghapus fungsi otomatis IPv6 dari sistem
    echo "0" > $ADDR_DIR/ipv6
    sed '/NETWORKING_IPV6/d' $ADDR_MOD/network > $TMP/temp6
    cp --reply=yes $TMP/temp6 $ADDR_MOD/network
    rm -f $TMP/temp6

# Menghapus fungsi IPv6 dari fungsi service network
    sed '/#aktifkan ipv6/d' $ADDR_NET/network > $TMP/net
    sed '/service ipv6 start/d' $TMP/net > $TMP/net1
    sed '/#deaktif ipv6/d' $TMP/net1 > $TMP/net
    sed '/service ipv6 stop/d' $TMP/net > $TMP/net1
    cp --reply=yes $TMP/net1 $ADDR_NET/network
    rm -f $TMP/net
    rm -f $TMP/net1
fi
}

# menambahkan alamat ipv6 ke dalam interface --> add
tambah()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_tambah
elif [[ $INPUT3 == eth? ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT3"
elif [[ $INPUT3 == gw ]]
then
    ip -f inet6 addr add $INPUT2 dev $INPUT5
    ip route add ::/0 via $INPUT4
    echo "address= $INPUT2" > "$ADDR_DIR/ip6cfg-$INPUT5"
    echo "gateway= $INPUT4" >> "$ADDR_DIR/ip6cfg-$INPUT5"
elif [[ $INPUT3 == sit? ]]
```

```
then
    ip -f inet6 addr add $INPUT2 dev $INPUT3
    echo "$INPUT2 dev $INPUT3" > $ADDR_DIR/ip6cfg-$INPUT3
else
    help_tambah
fi
}

# menghapus alamat ipv6 dari interface --> del
hapus()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_hapus
elif [[ $INPUT3 == eth? ]]
then
# Untuk membandingkan IPv6 yang aktif dengan yang ada di tabel
IPv6
    ip6 list $INPUT3 > $TMP/list
    cat $TMP/list |grep inet6|grep global > $TMP/list6
    cat "$ADDR_DIR/ip6cfg-$INPUT3"|grep address > $TMP/addr
    asli=`awk '{print $2}' "$TMP/addr"`
    scan=`awk '{print $2}' "$TMP/list6"`
# Untuk menghapus IPv6 dari sistem
    if [[ $asli == $scan ]]
    then
      ip -f inet6 addr del $INPUT2 dev $INPUT3
      rm -f $ADDR_DIR/ip6cfg-$INPUT3
    fi
    rm -f $TMP/list
    rm -f $TMP/list6
    rm -f $TMP/addr
elif [[ $INPUT3 == sit? ]]
then
    ip6 list $INPUT3 |grep global > $TMP/hpssit
    asli=`awk '{print $2}' "$TMP/hpssit"`
    scan=`awk '{print $1}' "$ADDR_DIR/ip6cfg-$INPUT3"`
    if [[ $asli == $scan ]]
    then
        ip tunnel del $INPUT2 dev INPUT3
      rm -f $ADDR_DIR/ip6cfg-$INPUT3
    fi
    rm -f $TMP/hpssit
else
    help_hapus
fi
}

# menampilkan daftar alamat ipv6 dari setiap interface --> list
tampil()
{
if [[ $INPUT2 == eth? || $INPUT2 == sit? ]]
then
    ip -f inet6 addr show dev $INPUT2
else
    ip -f inet6 addr
```

```
fi
}

# menjalankan fungsi interface tunnel --> tunnel
tunel()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_tunel
elif [[ $INPUT2 == add && $INPUT3 == sit? && \
      $INPUT4 == remote && $INPUT6 == local ]]
then
    ip tunnel add $INPUT3 mode sit remote $INPUT5 local \
    $INPUT7 2> $TMP/ceksit

    if [[ `cat $TMP/ceksit` == "" ]]
    then
      UP="ip link set $INPUT3 up"
      $UP
      echo "$INPUT3 remote $INPUT5 local $INPUT7" > \
      $ADDR_DIR/$INPUT3
    else
      echo "Perhatikan penulisan tunneling"
    fi
    rm -f $TMP/ceksit
elif [[ $INPUT2 == del && $INPUT3 ]]
then
    if [[ -e $ADDR_DIR/$INPUT3 ]]
    then
      DOWN="ip link set $INPUT3 down"
      ip tunnel del $INPUT3
      rm -f $ADDR_DIR/$INPUT3
    fi
else
    help_tunel
fi
}

# menjalankan fungsi route -->route
rute()
{
if [[ $INPUT2 == help || $INPUT2 == "" ]]
then
    help_route
elif [[ $INPUT2 == up ]]
then
    echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
    echo "1" > $ADDR_DIR/ipv6-forwarding
elif [[ $INPUT2 == down ]]
then
    echo "0" > /proc/sys/net/ipv6/conf/all/forwarding
    echo "0" > $ADDR_DIR/ipv6-forwarding
elif [[ $INPUT2 == add ]] && [[ $INPUT4 == via || $INPUT4 == gw ]]
then
    ip route add $INPUT3 via $INPUT5 2> $TMP/cek
    if [[ `cat $TMP/cek` == "" ]]
```

```
    then
      ip6 route show > $TMP/tur
      echo $INPUT3 > $TMP/addrrut
      sed 's/::...\//::\//' $TMP/addrrut > $TMP/addrbr
      sed 's/::..\//::\//' $TMP/addrbr > $TMP/inpbaru
      sed 's/::.\//::\//' $TMP/inpbaru > $TMP/addrrut
      tujuan=`cat $TMP/addrrut`
      cat /tmp/tur |grep $tujuan |grep $INPUT5 > $TMP/smpntur
      awk '{print $1, "via", $2}' $TMP/smpntur >> $TMP/unik
      uniq $TMP/unik > $TMP/antik
      cat $TMP/antik > $ADDR_DIR/ipv6-route6
    else
      echo "Perhatikan penulisan tabel routing!!!"
    fi
    rm -f $TMP/antik
    rm -f $TMP/unik
    rm -f $TMP/smpntur
    rm -f $TMP/inpbaru
    rm -f $TMP/addrbr
    rm -f $TMP/addrrut
    rm -f $TMP/tur
    rm -f $TMP/cek
elif [[ $INPUT2 == del ]] && [[ $INPUT3 == ::/0 || $INPUT3 ==
default ]]
then
    ip route  del ::/0
elif [[ $INPUT2 == del && $INPUT4 == via ]]
then
    ip route del $INPUT3 via $INPUT5 2> $TMP/cek
    if [[ `cat $TMP/cek` == "" ]]
    then
      echo "$INPUT3.via.$INPUT5" > $TMP/parsing
      sed 's/\//\\\//' $TMP/parsing > $TMP/hasilparsing
      hasil=`cat $TMP/hasilparsing`
      sed '/'$hasil'/d' $ADDR_DIR/ipv6-route6 > $TMP/route6
      cp --reply=yes $TMP/route6 $ADDR_DIR/ipv6-route6
      rm -f $TMP/route6
      rm -f $TMP/rutdel
      rm -f $TMP/hasilparsing
      rm -f $TMP/parsing
    else
      echo "Perhatikan penulisan penghapusan tabel routing!!!"
    fi
    rm -f $TMP/cek

## PERLAJARI ULANG

#elif [[ $INPUT2 == add && $INPUT4 == dev && $INPUT5 == sit? ]]
#then
#    ip route add $INPUT3 dev $INPUT5 2> $TMP/insit
#    if [[ `cat $TMP/insit` == "" ]]
#    then
#     echo "$INPUT3 dev $INPUT5" > $ADDR_DIR/route-sit
#    fi
```

```
elif [[ $INPUT2 == show || $INPUT2 == list || \
      $INPUT2 == print ]]
then
    route -A inet6
else
    help_route
fi
}

# bantuan untuk menjalankan program
bantuan()
{
echo "gunakan : ip6 OPSI {command | help}"
echo "          OPSI := {add | del | tunnel | route}"
echo "          command := {up | down | list | help}"
echo " "
}

help_tambah()
{
echo " gunakan : ip6 add [alamat ipv6] [gw [gateway]] [dev]"
echo "           address := alamat ipv6"
echo "           gateway := alamat gateway (optional)"
echo "           device  := interface yang akan diberi ipv6"
}

help_hapus()
{
echo " gunakan : ip6 del [ alamat ipv6 ] [ interface ]"
echo "           address := alamat ipv6 yang akan dihapus"
echo "           device  := interface yang akan dihapus ipv6-nya"
}

help_tunel()
{
echo "gunakan : ip6 tunnel [command] [dev] remote [ipv4 remote] \
local [ipv4 local]"
echo "          command := { add | del }"
echo "          dev := sit1, sit2, dst."
echo "          ipv4 remote := alamat ipv4 tujuan tunnel"
echo "          ipv4 local := alamat ipv4 sumber tunnel"
}

help_route()
{
echo "gunakan : ip6 route [ command ] [ destination ] [ via ] [
device ]"
echo "          command := add | del | show"
echo "          destination := alamat tujuan routing"
echo "          via := routing dilewatkan melalui alamat ini"
echo "          device := routing akan melalui interface ini"
}

case $INPUT1 in
    up|start)
      aktif;;
```

```
    down|stop)
      deaktif;;
    add)
      tambah;;
    del)
      hapus;;
    list|show)
      tampil;;
    tunnel)
      tunel;;
    route)
      rute;;
    help|-h|--help)
      bantuan;;
    *)
      bantuan;;
esac
```

Daemon ipv6

```
#!/bin/bash
# Script Startup untuk menjalankan IPv6
#
# chkconfig: 2345 95 92
# description: untuk menjalankan IPv6 pada saat boot up
#
# by Antonius Arief Widiyatmoko
# ant_arief@yahoo.com

# definisikan lokasi direktori
TMP=/tmp
ADDR_DIR=/etc/sysconfig/ipv6

# definisikan lokasi dan warna tanda [ OK ]
RES=60
LOCATION="echo -en \\033[${RES}G"
COLOR="echo -en \\033[1;32m"
NORMAL="echo -en \\033[0;39m"
TIME=300000

# fungsi untuk menampilkan tanda [ OK ]
OK_CODE()
{
usleep $TIME
$LOCATION
echo -n "[  "
$COLOR
echo -n $"OK"
$NORMAL
echo -n "  ]"
echo ""
}

# menjalankan service ipv6
mulai()
{
echo -n "Aktifkan ipv6: "
# untuk mengaktifkan alamat IPv6 ke device
i=0
while :
do
    if [[ -e "$ADDR_DIR/ip6cfg-eth$i" ]]
    then
      cat $ADDR_DIR/ip6cfg-eth$i |grep address > $TMP/addr
        awk '{print $2}' $TMP/addr > $TMP/bot6
      ip -f inet6 addr add `cat $TMP/bot6` dev "eth$i"
        rm -f $TMP/addr
        rm -f $TMP/bot6
# jika ditemukan gateway, maka gateway akan di aktifkan
      cat $ADDR_DIR/ip6cfg-eth$i |grep gateway > $TMP/gate
        gerbang=`awk '{print $2}' $TMP/gate`
        if [[ $gerbang != "" ]]
        then
          ip route add ::/0 via $gerbang
```

```
        fi
    fi
    if [[ ($i > 5) ]]
    then
        break
    fi
rm -f $TMP/gate
let i+=1
done

# aktifasi tunnel otomatis
q=0
while (( $q <= 5 ))
do
    let q+=1
    if [[ -e "$ADDR_DIR/sit$q" && -e "$ADDR_DIR/ip6cfg-sit$q" ]]
    then
      awk '{print $1," mode sit ",$2, $3, $4, $5, $6}'\
      "$ADDR_DIR/sit$q" > $TMP/situp
      siter=`cat $TMP/situp`
      ip tunnel add $siter
      UP="ip link set sit$q up"
      $UP
        ip addr add `cat $ADDR_DIR/ip6cfg-sit$q`
    fi
done
rm -f $TMP/situp

# digunakan untuk menghitung jumlah tabel routing
if [[ -e "$ADDR_DIR/ipv6-route6" ]]
then
    cat $ADDR_DIR/ipv6-route6 | wc -l > $TMP/counter6
    awk '{print $1}' $TMP/counter6 > $TMP/count6
# digunakan untuk mengaktifkan tabel routing
    if [[ `cat "$ADDR_DIR/ipv6-route6"` != "" ]]
    then
      awk '{print "rut" NR, $1, $2, $3, $4, $5}'\
          $ADDR_DIR/ipv6-route6 > $TMP/route6
      p=0
      while :
        do
          let p+=1
          awk '/rut'$p'/{print $2, $3, $4, $5, $6 }' \
            $TMP/route6 > $TMP/rut6
          ip route add `cat $TMP/rut6`
          if [[ $p == `cat $TMP/count6` ]]
          then
            break
          fi
      done
    rm -f $TMP/route6
    rm -f $TMP/rut6
    fi
fi
rm -f $TMP/count6
rm -f $TMP/counter6
```

```
# digunakan untuk forwarding otomatis
if [[ -e $ADDR_DIR/ipv6-forwarding && `cat $ADDR_DIR/ipv6-
forwarding` == 1 ]]
then
    echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
fi

# tampilkan tanda [ OK ]
OK_CODE

}

# matikan service ipv6 dan hapus alamat ipv6
selesai()
{
echo -n "Nonaktifkan ipv6: "
# hapus routing tabelnya dulu
ip6 route show > $TMP/ruteasli
cat $TMP/ruteasli |wc -l > $TMP/jmlrut
awk '{print $1}' $TMP/jmlrut > $TMP/ttlrut
awk '{print "rut" NR, $1}' $TMP/ruteasli > $TMP/hpsrut
p=2
while :
do
    let p+=1
    awk '/rut'$p'/{print $2}' $TMP/hpsrut > $TMP/del
    ip route del `cat $TMP/del`
    if [[ $p == `cat $TMP/ttlrut` ]]
    then
      break
    fi
done
rm -f $TMP/ruteasli
rm -f $TMP/jmlrut
rm -f $TMP/ttlrut
rm -f $TMP/hpsrut
rm -f $TMP/del

# hapus alamat sit beserta devicenya
q=5
while (( $q >=1 ))
do
    ip6 list |grep sit$q > $TMP/ceksit
    if [[ `cat $TMP/ceksit` != "" ]]
    then
      ip tunnel del sit$q
    fi
let q-=1
done
rm -f $TMP/ceksit

# ini baru menghapus alamat IPv6 dari eth
# dan mematikan interface eth0
i=5
while (( $i >= 0 ))
```

```
do
    if [ -e "$ADDR_DIR/ip6cfg-eth$i" ]
    then
      ifconfig eth$i down
      ifconfig eth$i up
    fi
    if [[ $i == 0 ]]
    then
      break
    fi
let i-=1
done

# tampilkan [ OK ]
OK_CODE
}

case "$1" in
    start)
      mulai ;;
    stop)
      selesai ;;
    restart)
      # proses yang dipakai adalah stop kemudian start
      selesai
      mulai;;
    *)
      echo "Gunakan : $0 { start | stop | restart }"
      exit 1
esac
exit 0
```