ABSTRAK

Perkembangan teknologi jaringan komputer dewasa ini semakin pesat seiring dengan kebutuhan masyarakat akan layanan yang memanfaatkan jaringan komputer. Pada sistem jaringan komputer, protokol merupakan suatu bagian yang paling penting. Protokol jaringan yang umum digunakan adalah IPv4, yang masih terdapat beberapa kekurangan dalam menangani jumlah komputer dalam suatu jaringan yang semakin kompleks.

Telah dikembangkan protokol jaringan baru, yaitu IPv6 yang merupakan solusi dari masalah diatas. Protokol baru ini belum banyak diimplementasikan pada jaringan-jaringan di dunia.

Dalam tugas akhir ini akan di rancang sebuah system jaringan yang menggunakan protokol IPv6 dengan menggunakan sistem operasi *Linux*. Protokol IPv6 ini akan diimplementasikan dalam protokol *routing* (RIPng, OSPFv3), applikasi web server, mail server, proxy server dan applikasi server-client.

KATA PENGANTAR

Akhirnya selesai sudah Tugas Akhir dan laporan yang berjudul :
PERANCANGAN DAN IMPLEMENTASI JARINGAN IPV6 DI ITS-NET DENGAN SISTEM OPERASI *LINUX*.

Kami menyadari bahwa dalam penyusunan laporan tugas akhir ini tentunya masih terdapat kekurangan, sehingga kami sangat menghargai segala kritik dan masukan yang berguna dari pembaca. Semoga laporan ini dapat bermanfaat untuk pengembangan ilmu pengetahuan komputer, khususnya di bidang komputer paralel.

Sekali lagi penulis berharap semoga laporan tugas akhir ini dapat bermanfaat.

Surabaya, 9 Agustus 2002
Penyusun

UCAPAN TERIMA KASIH

Ucapan terima kasih dan rasa syukur yang tak terhingga ini penulis sampaikan kepada Allah SWT semata, hanya karena kasih sayang-Nya lah Tugas Akhir ini selesai, kemudian kepada berbagai pihak yang telah banyak membantu tugas akhir ini, secara khusus penulis ucapkan terima kasih kepada:

1. Bapak Supeno Mardi, SN. ST. MT, dan Bapak Surya Sumpeno ST. Selaku dosen pembimbing dan dosen wali yang dengan sabar memberikan pengarahan dan bimbingan untuk penulisan Tugas Akhir ini.
2. Bapak Ir. Yoyon Kusnendar, MSc, selaku Ketua Bidang Studi Komputer yang telah banyak membantu dan memberikan segala fasilitas yang diperlukan.
3. Bapak Ir. Katjuk Astrowulan, MSEE sebagai Kepala UPT Pusat Komputer ITS yang sudah memberikan izin atas ITS-Net sebagai tempat implementasi dari Tugas Akhir ini.
4. Dosen-dosen dan Pegawai di lingkungan jurusan Teknik Elektro.
5. Ayahanda (Prof. Dr. Ir. KRT. H. S. Sukardjono almarhum) mamah, dan keluarga tercinta Mbak Fitri, Mas Iman, Mas Njul, Mbak Tutik dan Yu Kar yang telah banyak memberikan dorongan moril, materiil dan doa agar dapat menyelesaikan pendidikan ini.
6. Om Heru Agustono, Bi Ati dan Krucil, terima kasih buku debiannya dan semua bantuannya selama ini.
7. ITS-Net crew: Mas Cahya, Mas Royyana, Mbak Mudji, Mbak Yeni, Wicak, Budi, Ninchi, Q-lon+Nuniek dan Pak Tulus atas segala bantuannya.
8. Lab-B201: Enthunk, Kunthil, Menthox, Cemetz, Makur, Codod, Gethug, Thebbbal, Menjezzz, Tuobhiel teman-teman satu lab dan adik kelas atas bantuannya.
9. Ghoank, Kakuz, Senggek, Dandang, Banzer, Akik, Rangsang, Ketip2, dan Joels(97) atas pelajaran *LINUX* nya.
10. Mas Yusuf dan Mas Zulkifli atas perhatiannya selama ini.

11. Khusus buat Kiky terima kasih atas bantuan, kesetiaan dan kesabarannya selama mendampingi penyelesaian Tugas Akhir ini.
12. Anak Kost B-2 yang bisa dikerjain, Yeni, Echi, Maya, Indri, Yeti, Butet, Iva, Eni.
13. M. Samik Ibrahim, ketua klan VLSM.ORG sehingga bisa memberikan koneksi ke ITB dan lainnya.
14. http://www.ipv6.itb.ac.id, http://www.freenet6.net dan http://www.ipv6tb.he.net atas *tunnel* IPv6
15. Debian GNU/*Linux*, http://www.debian.org
16. #indo*linux*, Dalnet IRC
17. Gemplo (Rikko, Koko, Arif, Kiky, Nurman) yang mau mengalah kalau penulis lagi sok sibuk.
18. Teman-teman seangkatan (e-37) di jurusan Teknik Elektro.
19. Dragon, Phoenix, Pegasus, Samba, Socks, Cerberus, Gryphon, Lexican, Experiment-warnet, Salwa, Pythagoras dan Kebo-Desperate para server *Linux* di ITS-Net.
20. Tux, Deby, Arale, Corel, Slacky, Little, Suse, Domba, dan Monmon para boneka yang menjaga para server.
21. Komik-komik yang ikut menghibur penulis kalau sedang jenuh.
22. Nokia+ProXL dan Telkom yang telah membantu dalam komunikasi.
23. Rekan-rekan sesama TA-er, Gethug, Gerdhuu, Kambil, Nglouncope, Ketip2, Linggis dan yang lainnya.
24. Dan spesial buat Mega.

Serta pihak-pihak yang belum penulis cantumkan diatas.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE

BAB I. PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi jaringan komputer dewasa ini semakin pesat seiring dengan kebutuhan masyarakat akan layanan yang memanfaatkan jaringan komputer. Pada sistem jaringan komputer, protokol merupakan suatu bagian yang paling penting. Protokol jaringan yang umum digunakan adalah IPv4 (Internet Protocol vesion 4.0), dimana masih terdapat beberapa kekurangan dalam menangani penambahan jumlah komputer dalam suatu jaringan yang semakin kompleks.

Telah dikembangkan protokol jaringan baru, yaitu IPv6 (Internet Protocol version 6.0) yang merupakan solusi dari masalah diatas. Protokol baru ini belum banyak diimplementasikan pada jaringan-jaringan komputer terbesar di dunia.

1.2. Tujuan

Tujuan dari Tugas Akhir ini adalah untuk mengimplementasikan protokol ipv6 tersebut pada jaringan ITS-NET dimana pada jaringan tersebut terdapat aplikasi-aplikasi server-client.

1.3. Permasalahan

Dalam penyelesain tugas akhir ini dirumuskan beberapa permasalahan yang dihadapi, yaitu :

1. Bagaimana merancang sebuah jaringan dengan menggunakan protokol IPv6 dengan menggunakan sumber daya dan jaringan yang sudah ada di ITS-NET.
2. Bagaimana mengimplementasikan protokol IPv6 ini dengan menggunakan *Linux* PC *router* dan protokol *router* seperti RIPng, OSPFv3.
3. Bagaimana mengimplementasikan aplikasi server *Linux* yang sudah ada di ITS-NET supaya dapat berjalan dengan menggunakan protokol IPv6.

1.4. Batasan Masalah

Batasan masalah dari tugas akhir ini adalah perancangan dan implementasi jaringan dengan protokol IPv6 menggunkan system operasi *Linux*, berikut langkah-langkah yang dilakukan:

1. Perancangan model jaringan dan migrasi ke IPv6 dengan memperhatikan model jaringan yang sudah ada.
2. Instalasi system operasi *Linux*.
3. Kompilasi kernel *Linux* supaya dapat menggunakan protokol IPv6.
4. PC *Linux* dan software "Zebra" sebagai *router* dan koneksi IPv6 ke internet.
5. PC *Linux* dengan aplikasi server antara lain : web server, DNS server, Mail server dan Proxy server.

1.5. Metodologi

Untuk mencapai tujuan diatas, dilakukan langkah-langkah sebagai berikut :

1. Pengumpulan bahan-bahan referensi, yang meliputi referensi protokol IPv6, membuat koneksi dengan pihak luar seperti ITB (Institut Teknologi Bandung), Freenet6 supaya mendapatkan koneksi IPv6 ke internet, referensi *Linux* Networking beserta aplikasi yang bersangkutan, mengikuti miling list yang bersangkutan dengan IPv6 dan *Linux*, serta referensi lainnya yang menunjang pencapaian tujuan tersebut.
2. Perancangan dan implementasi jaringan IPv6 di network ITS-Net, mengaktifkan koneksi IPv6 ke internet dan membuat service yang ada di ITS-Net mampu menggunakan protokol IPv6.
3. Penulisan laporan tugas akhir.

1.6. Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini dibagi dalam enam bab, masing-masing bab dapat diuraikan sebagai berikut :

1. BAB I merupakan pendahuluan yang berisi latar belakang, tujuan permasalahan, batasan masalah, metodologi dan sistematika penulisan.

2. BAB II membahas tentang dasar-dasar dan teori IPv6 serta membahas protokol-protokol penting yang digunakan pada IPv6.
3. BAB III memuat tutorial bagaimana mengimplementasikan IPv6 pada system operasi *Linux*.
4. BAB IV berisi tentang implementasi IPv6 di jaringan ITS-Net dengan menggunakan IPv6-in-IPv4. Implementasi *Router* IPv6 dengan menggunakan RadvD dan Zebra. Implementasi DNS, Web, Mail, Proxy, FTP, SSH, IPsec dan Firewall pada protokol IPv6.
5. BAB V membahas tentang ujicoba dari jaringan IPv6 ITS-Net
6. BAB VI merupakan penutup yang berisi kesimpulan dan saran.

Ini kosong !

BAB II. DASAR TEORI

2.1. Spesifikasi Dasar IPv6[1]

IP versi 6 (IPv6) adalah protokol Internet versi baru yang didesain sebagai pengganti dari Internet protocol versi 4 (IPv4) yang didefinisikan dalam RFC 791.

2.1.1. Pendahuluan

Perubahan dari IPv4 ke IPv6 pada dasarnya terjadi karena beberapa hal yang dikelompokkan dalam kategori berikut :

1. Kapasitas Perluasan Alamat
   IPv6 meningkatkan ukuran dan jumlah alamat yang mampu didukung oleh IPv4 dari 32bit menjadi 128bit. Peningkatan kapasitas alamat ini digunakan untuk mendukung peningkatan hirarki atau kelompok pengalamatan, peningkatan jumlah atau kapasitas alamat yang dapat dialokasikan dan diberikan pada *node* dan mempermudah konfigurasi alamat pada *node* sehingga dapat dilakukan secara otomatis. Peningkatan skalabilitas juga dilakukan pada *routing multicast* dengan meningkatkan cakupan dan jumlah pada alamat *multicast*. IPv6 ini selain meningkatkan jumlah kapasitas alamat yang dapat dialokasikan pada *node* juga mengenalkan jenis atau tipe alamat baru, yaitu alamat *anycast*. Tipe alamat *anycast* ini didefinisikan dan digunakan untuk mengirimkan paket ke salah satu dari kumpulan *node*.
2. Penyederhanaan Format *Header*
   Beberapa kolom pada *header* IPv4 telah dihilangkan atau dapat dibuat sebagai *header* pilihan. Hal ini digunakan untuk mengurangi biaya pemrosesan hal-hal yang umum pada penanganan paket IPv6 dan membatasi biaya *bandwidth* pada *header* IPv6. Dengan demikian, pemerosesan *header* pada paket IPv6 dapat dilakukan secara efisien.
3. Peningkatan dukungan untuk *header* pilihan dan *header* tambahan *(Options and extention header)*

[1] S. Deering, R. Hinden, 1998, **Internet Protocol, Version 6 (IPv6) Specification**, Request for Comments 2460

Perubahan yang terjadi pada *header-header* IP yaitu dengan adanya pengkodean *header Options* (pilihan) pada IP dimasukkan agar lebih efisien dalam penerusan paket (*packet forwarding*), agar tidak terlalu ketat dalam pembatasan panjang *header* pilihan yang terdapat dalam paket IPv6 dan sangat fleksibel/dimungkinkan untuk mengenalkan *header* pilihan baru pada masa akan dating.

4. Kemampuan pelabelan aliran paket
   Kemampuan atau fitur baru ditambahkan pada IPv6 ini adalah memungkinkan pelabelan paket atau pengklasifikasikan paket yang meminta penanganan khusus, seperti kualitas mutu layanan tertentu (*QoS*) atau *real-time.*
5. Autentifikasi dan kemampuan privasi
   Kemampuan tambahan untuk mendukung autentifikasi, integritas data dan data penting juga dispesifikasikan dalam alamat IPv6.

2.1.2. Terminologi

Node
Peralatan yang mengimplementasikan IPv6.

Router
Node yang melewatkan paket IPv6.

Host
Node lainnya yang tidak merupakan *router*.

Upper-layer
Layer protocol yang secara langsung berada di atas IPv6. Sebagai contoh adalah protokol transport seperti TCP dan UDP, protokol control seperti ICMP, protokol *routing* seperti OSPF dan Internet atau protokol level bawah di*tunnel* melalui IPv6 seperti IPX, Appletalk, dan IPv6 sendiri (IPX over IPv6, Appletalk over IPv6 dan IPv6 over IPv6).

Link
Fasilitas komunikasi atau medium, yaitu *node* dapat berkomunikasi pada *layer link*. *Layer link* ini yang secara langsung dibawah *layer*

IPv6. Sebagai contoh dari *link* adalah Ethernet (secara sederhana maupun menggunakan bridge); *link* PPP; X.25, Frame Relay, atau jaringan ATM, dan *layer* Internet *tunnel* seperti *tunnel* melalui IPv4 atau IPv6 sendiri.

Neighbors
Node lain yang dihubungkan dalam *link* yang sama

Interface
Media penghubung dari *node* (berada pada *node*) ke jaringan.

Address
Identifikasi pada *layer* IPv6 untuk *interface* atau sekumpulan *interface*.

Packet
Header IPv6 dan payload-nya (isi).

Link MTU
Maximum transmission unit. Ukuran maksimum paket dalam ukuran byte yang dapat disampaika melalui *link*.

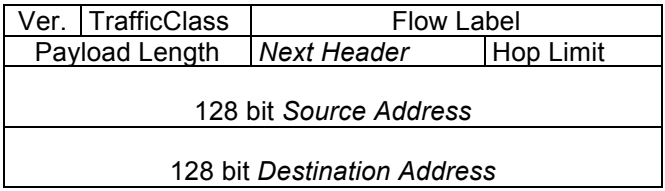
Path MTU
Link MTU yang paling kecil dari semua *link* dalam *path node* asal sampai *node* tujuan.

2.1.3. Format *header* IPv6

Format *header* alamat IPv6 menyederhanakan format *header* pada alamat IPv4. Perbandingan antara format *header* IPv6 (Gambar II.1) dan IPv4 (Gambar II.2).

Ver.	*header*	TOS	Total length	
Identification			flag	*Fragment* offset
TTL		Protokol	Checksum	
32 bit *Source Address*				
32 bit *Destination Address*				

Gambar II.1Format Header IPv4



Gambar II.2 Format header IPv6

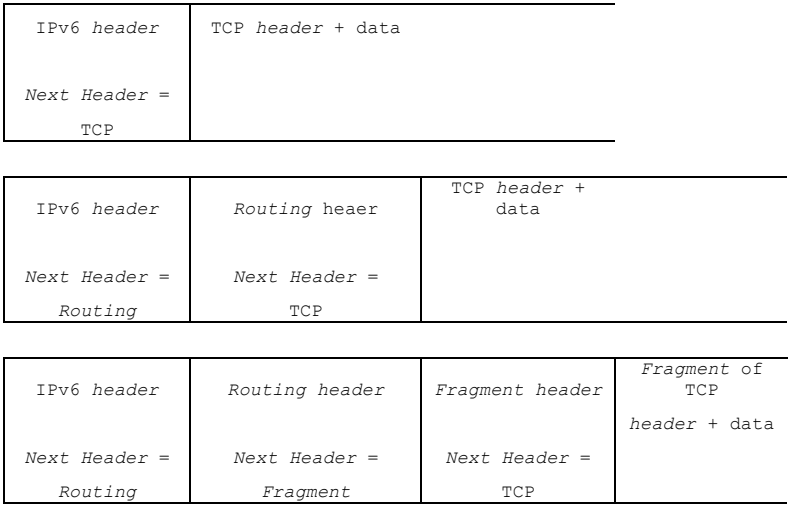
Keterangan

Version	4-bit nomor versi *Internet Protocol* = 6.
Traffic Class	8-bit *field traffic class*.
Flow Label	20-bit *flow label*
Payload Length	16-bit *unsigned* integer. Panjang dari payload IPv6, sebagai contoh, keseluruhan paket tersebut mengikuti *header* IPv6 ini, dalam oktet. (Perlu diperhatikan bahwa *header* ekstensi manapun yang ada merupakan bagian dari payload, termasuk dalam jumlah panjangnya)
Next Header	8-bit selector. Mengidentifikasi tipe *header* yang langsung mengikuti *header* IPv6. Menggunakan nilai yang sama seperti *field* protokol IPv4.
Hop Limit	8-bit *unsigned* integer. Dikurangi dengan 1 oleh setiap *node* yang meneruskan paket.
Source Address	128-bit alamat asal dari paket.
Destination Address	128-bit alamat penerima yang dituju dari paket (bisa jadi bukan penerima terakhir, jika terdapat *header routing*)

2.1.4. IPv6 *Extension Header* (*Header* Tambahan IPv6)

Dalam IPv6, pilihan informasi *internet-layer* di-encode dalam *header-header* yang terpisah yang mungkin diletakkan diantara *header* IPv6 dan *header* setingkat diatasnya dalam suatu paket. Ada sejumlah kecil *header* ekstensi yang serupa, setiap *header* tersebut diidentifikasi oleh suatu nilai *Next Header* yang pasti (fix). Sebagai

ilustrasi dalam gambar II.3 contoh berikut ini, suatu paket IPv6 mungkin membawa nol, satu, atau lebih *header* ekstensi, setiap paket tersebut diidentifikasi oleh *field Next Header* dari *header* yang mendahului :



Gambar II.3Contoh header tambahan pada IPv6

Dengan satu pengecualian, *header* ekstensi tidak diuji atau diproses oleh *node* manapun sepanjang *path* pengiriman dari suatu paket, hingga paket mencapai *node* tersebut (atau setiap *node* dari sekelompok *node*, dalam hal *multicast*) yang diidentifikasi dalam *field Destination Address* dari *header* IPv6. *Demultiplexing* normal pada *field Next Header* dari *header* IPv6 membutuhkan modul tersebut untuk memproses *header* ekstensi awal, atau *header* setingkat diatasnya jika tidak terdapat *header* ekstensi. Isi dan maksud dari setiap *header* ekstensi menentukan perlu atau tidak untuk meneruskan ke *header* selanjutnya.

Pengecualian di atas adalah *header Hop-by-Hop Options*, yang membawa informasi yang harus diuji dan diproses oleh setiap *node* sepanjang *path* pengiriman suatu paket, meliputi *node* sumber dan *node* tujuan. Saat terdapat *header Hop-by-Hop Options*, harus segera mengiktui paket *header* IPv6. Keberadaannya ditandai oleh nilai nol dalam *field Next Header* dari *header* IPv6.

Jika sesuai dengan hasil dari pemrosesan suatu *header*, suatu *node* diharuskan untuk meneruskan ke *header* selanjutnya tetapi nilai *Next Header* dalam *header* saat ini tidak dikenali oleh *node*, ini seharusnya membuang paket tersebut dan mengirim suatu pesan Parameter Problem ICMP ke sumber/alamat asal paket tersebut, dengan suatu nilai Kode ICMP 1 (*"unrecognized Next Header Type encountered"*) dan *field Pointer* ICMP berisi offset dari nilai yang tidak dikenali dalam paket yang asli/semula. Tindakan yang sama seharusnya diambil, jika suatu *node* mendapati suatu *Next Header* bernilai nol dalam *header* IPv6.

Suatu implementasi penuh dari IPv6 meliputi implementasi dari *header* ekstensi berikut ini :

1. *Hop-by-Hop Options*
2. *Routing (Type 0)*
3. *Fragment*
4. *Destination Options*
5. *Authentication*
6. *Encapsulating Security Payload*

2.1.5. *Extension Header* Order (Urutan *Header* Tambahan)

Ketika lebih dari satu *extension header* digunakan dalam paket yang sama, seharusnya *header* tersebut muncul sebagai berikut:

1. *IPv6 header*
2. *Hop-by-Hop Options header*
3. *Destination Options header (note 1)*
4. *Routing header*
5. *Fragment header*
6. *Authentication header (note 2)*
7. *Encapsulating Security Payload header (note 2)*
8. *Destination Options header (note 3)*
9. *upper-layer header*

Note 1 : Untuk opsi yang diproses dalam *field Destination Address* IPv6.

Note 2 : Rekomendasi tambahan yang berkaitan dengan urutan relatif dari *header* Authentication dan *Encapsulating Security Payload* yang terdapat dalam RFC-2406

Note 3 : Untuk opsi yang diproses hanya oleh *Destination* terakhir dari paket.

Setiap *extension header* seharusnya terjadi hanya sekali, kecuali untuk *header Destination Options* yang seharusnya terjadi dua kali (sekali sebelum *header Routing* dan sekali sebelum *header* upper-*layer*).

Jika *header* upper-*layer* adalah *header* IPv6 yang lain (dalam hal ini adalah IPv6 yang disalurkan melalui atau dienkapsulasi dalam IPv6), ini mungkin diikuti oleh *extension header*-nya sendiri, yang merupakan subyek terpisah pada rekomendasi pengurutan yang sama.

Ketika *extension header* yang lain didefinisikan, batasan pengurutannya yang relatif pada *header* yang terdaftar diatasnya harus ditentukan.

Node-node IPv6 harus menerima dan mencoba untuk memproses *extension header* dalam urutan apapun dan membuat terjadi berapa kali pun dalam paket yang sama, kecuali untuk *header Hop-by-Hop Options* yang tiba-tiba muncul tiba-tiba setelah *header* IPv6 saja. Meskipun demikian, sangat disarankan agar *source* dari paket IPv6 menempel diatas urutan yang direkomendasikan sampai dan kecuali kalau spesifikasi berikutnya merevisi rekomendasi tersebut.

2.1.6. *Option*

Dua dari *extension header* yang terdefinisi saat ini – *header Hop-by-Hop Options* dan *header Destination Options* – membawa variabel jumlah dari *Type*-length-value (TLV) yang meng-encode "*Options*", dari format berikut ini:

Option Type	8-bit pengidentifikasi dari tipe opsi.
Opt Data Len	8-bit *unsigned* integer. Panjang *field Option* Data dari opsi ini, dalam oktet.
Option Data	*Field Variable-length*. Data *Option-Type-spesific*.

Rangkaian opsi dalam suatu *header* harus diproses secara cepat dalam urutan dimana opsi tersebut muncul dalam *header* tersebut; penerima tidak harus demikian, sebagai contoh, mendeteksi

sepanjang *header* tersebut untuk mencari jenis opsi dasar dan proses opsi itu sebelum pemrosesan semua opsi sebelumnya.

Pengidentifikasi *Option Type* di-encode secara internal sedemikian hingga dua bit urutan tertinggi-nya menentukan aksi yang harus diambil jika pemrosesan *node* IPv6 tidak mengenali *Option Type* tersebut :

00 - Menghiraukan/meloncati opsi tersebut dan melanjutkan pemrosesan *header*.
01 - Membuang paket.
10 - Membuang paket dan, tanpa menghiraukan apakah *Destination Address* dari paket tersebut adalah *Address multicast*, mengirim suatu ICMP Parameter Problem, Code 2, memberitahu *Source Address* dari paket, menunjuk pada *Option Type* yang tidak dikenali.
11 - Membuang paket tersebut dan, hanya jika *Destination Address* dari paket bukan merupakan suatu alamat *multicast*, mengirim suatu ICMP Parameter Problem, Code 2, memberitahu *Source Address* dari paket, menunjuk pada *Option Type* yang tidak dikenali.

Bit urutan tertinggi ketiga dari *Option Type* menentukan apakah *Option* Data dari opsi tersebut dapat merubah *route* pada *Destination* terakhir dari paket. Ketika terdapat suatu *header Authentication* dalam paket tersebut, untuk opsi manapun yang memiliki data yang dapat mengubah en-route, keseluruhan *field Option* Data-nya harus perlakukan sebagai nilai nol oktet ketika menghitung atau memverifikasi nilai autentikasi dari paket.

0 - *Option* Data tidak merubah route
1 - *Option* Data dapat merubah route

Bit urutan tertinggi ketiga yang dijelaskan diatas diperlakukan sebagai bagian dari *Option Type*, bukan di luar *Option Type*. Karena itu, suatu opsi utama yang diidentifikasi oleh 8-bit penuh *Option Type*, tidak hanya 5-bit urutan terendah dari *Option Type*.

Penomoran *Option Type* yang sama digunakan baik untuk *header Hop-by-Hop Option* dan *header Destination Options*.

Bagaimanapun juga, Spesifikasi suatu opsi utama dapat membatasi fungsinya hanya pada satu dari dua *header* tersebut.

Opsi individual dapat memiliki syarat pensejajaran yang spesifik, untuk memastikan bahwa nilai multi-octet dalam *field Option* Data yang terletak pada batasan alami. Syarat pensejajaran dari suatu opsi ditentukan menggunakan notasi xn+y, berarti *Option Type* harus muncul pada suatu perkalian integer x oktet dari awal *header*, ditambah y oktet. Sebagai contoh :

2n	Artinya offset 2-oktet manapun dari awal *header*.
8n+2	Artinya offset 8-oktet manapun dari awal *header*, plus 2-oktet

Ada dua *padding Option* yang digunakan jika perlu untuk mensejajarkan *Option* berikutnya dan untuk membagi *header* menjadi 8-oktet ganda. Padding *Option* ini harus dikenali oleh seluruh implementasi IPv6.

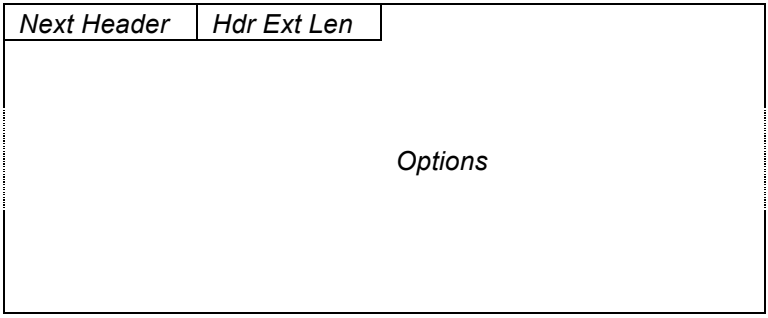
CATATAN! Format dari Pad1 *Option* adalah kasus khusus – tidak memiliki *field* length dan value.

Pad1 *Option* digunakan untuk menyisipkan satu oktet padding ke dalam area *Option* dari suatu *header*. Jika lebih dari satu oktet padding diharuskan, PadB *Option*, yang akan dijelaskan selanjutnya, seharusnya digunakan dibanding Pad1 *Option* ganda.

PadN *Option* digunakan untuk menyisipkan dua atau lebih oktet padding ke dalam area *Option* dari suatu *header*. Untuk N oktet padding, *field* Opt Data Len berisi nilai N-2, dan *Option* Data terdiri atas N-2 nilai nol oktet.

2.1.7. *Hop-by-Hop Option Header* (*Header* Pilihan *Hop-by-Hop*)

Header Hop-by-Hop Option digunakan untuk membawa informasi opsional yang harus dicek oleh setiap *node* sepanjang *path* pengiriman dari suatu paket. *Header Hop-by-Hop Option* dikenali dengan nilai 0 *Next Header* dalam *header* Ipv6, dan memiliki format sebagai berikut:



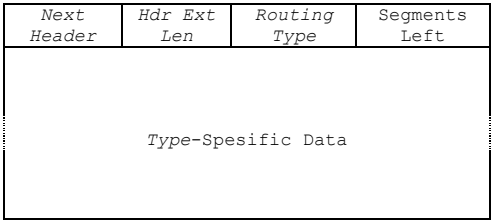
Gambar II.4 Format Header Hop-by-Hop

Keterangan gambar:

Next Header	Selektor 8-bit. Mengenali tipe *header* tepat setelah *header Hop-by-Hop Option*. Menggunakan nilai yang sama seperti *field* Protokol Ipv4.
Hdr Ext Len	8-bit *unsigned* integer. Panjang dari *header Hop-by-Hop Option* dalam satuan 8-oktet, tidak termasuk 8 oktet awal.
Options	*Field variable-length*, dari panjang dimana *header Hop-by-Hop Option* yang lengkap adalah suatu integer dengan panjang 8 oktet ganda. Berisi satu atau lebih opsi TLV-encoded, seperti yang dijelaskan pada bagian 4.2.

2.1.8. *Routing Header*

Header Routing digunakan oleh suatu IPv6 *source* untuk mendaftar satu atau lebih *node* intermediate yang dikunjungi dalam perjalanan menuju *Destination* dari paket. Fungsi ini hampir sama dengan opsi *Loose Source* dan Record Route pada Ipv4. *Header Routing* dikenali dengan nilai 43 pada *Next Header* tepat pada *header* sebelumnya, dan memiliki format sebagai berikut:



Gambar II.5 Format Header Routing

Keterangan gambar:

Next Header	8-bit selektor. Mengidentifikasi tipe *header* tepat sebelum *header Routing*. Menggunakan nilai yang sama seperti *field* Protokol IPv4.
Hdr Ext Len	8-bit *unsigned* integer. Panjang dari *header Routing* dalam satuan 8-oktet, tidak termasuk 8 oktet pertama.
Routing Type	8-bit pengenal dari varian *header Routing* utama.
Segment Left	8-bit *unsigned* integer. Jumlah segmen route berikutnya, sebagai contoh, jumlah intermediate *node* yang terdaftar secara eksplisit masih akan dilewati sebelum mencapai *Destination* akhir.
Type-specific data	*Field Variable-length*, dari format ditentukan oleh *Routing Type*, dan dari panjang dimana *header Routing* lengkap merupakan suatu integer dengan panjang 8 oktet ganda.

Jika saat memproses paket yang diterima, suatu *node* menangani *header Routing* dengan nilai *Routing Type* yang tidak dikenali, tingkah laku yang diharuskan dari *node* tersebut bergantung pada nilai dari *field Segment Left*, sebagai berikut :

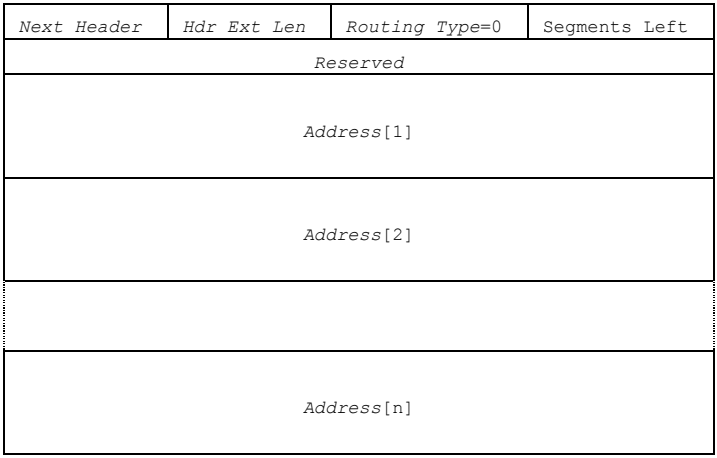
Jika Segment Left sama dengan nol, *node* tersebut harus mengabaikan *header Routing* dan lebih dahulu memproses *Next*

Header dalam paket tersebut, yang tipenya dikenali oleh *field Next Header* dalam *header Routing*.

Jika Segment Left tidak sama dengan nol, *node* tersebut harus menolak/membuang paket tersebut dan mengirim suatu pesan ICMP Parameter Problem, Code 0, ke *Source Address* dari paket tersebut, mengacu/menunjuk pada *Routing Type* yang tidak dikenali.

Jika, setelah memproses suatu *header Routing* dari paket yang diterima, intermediate *node* menentukan bahwa paket tersebut akan diteruskan ke suatu *link* yang *link* MTU-nya lebih kecil dari ukuran paket tersebut, *node* tersebut harus membuang paket dan mengirim pesan ICMP Packet Too Big ke *Source Address* dari paket tersebut.

Tipe 0 *header Routing* memiliki format sebagai berikut:



Gambar II.6 Format Header Routing Nol

Keterangan gambar:

Next Header 8-bit selektor. Mengidentifikasi tipe *header* tepat sebelum *header Routing*. Menggunakan nilai yang sama seperti *field* Protokol IPv4.

Hdr Ext Len	8-bit *unsigned* integer. Panjang dari *header Routing* dalam satuan 8-oktet, tidak termasuk 8 oktet pertama. Untuk *Type* 0 *header Routing*, *Hdr Ext Len* sama dengan dua kali jumlah alamat-alamat dalam *header* tersebut.
Routing Type	0.
Segment Left	8-bit *unsigned* integer. Jumlah segmen route berikutnya, sebagai contoh, jumlah intermediate *node* yang terdaftar secara eksplisit masih akan dilewati sebelum mencapai *Destination* akhir.
Reserved	32-bit *field reserved*. Diinisialisasi dengan nol untuk transmisi; diabaikan pada penerimaan.
Address[1..n]	Vektor dari 128-bit alamat, dinomori 1 s/d n.

Alamat-alamat *multicast* harus tidak muncul dalam suatu *header Routing* dari *Type* 0, atau dalam *field* IPv6 *Destination Address* dari suatu paket pembawa *header Routing* dari *Type* 0.

Suatu *header Routing* tidak diperiksa atau diproses sampai *header* tersebut mencapai *node* yang diidentifikasi dalam *field Destination Address* dari *header* IPv6. Dalam *node* tersebut, pengiriman pada *field Next Header* dari *header* tepat sebelumnya menyebabkan modul *header Routing* yang diminta, dalam hal ini *Routing Type* 0, melakukan algoritma sebagai berikut:

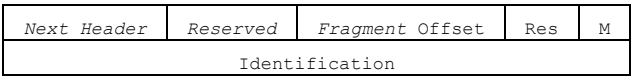
```
If Segments Left = 0 {
            memulai untuk memproses next header dalam paket
            tersebut, yang memiliki tipe yang diidentifikasi oleh
            field Next Header dalam header Routing
}
else if Hdr Ext Len adalah ganjil {
            mengirim suatu pesan ICMP Parameter Problem, Code 0,
            ke Source Address, menunjuk pada field Hdr Ext Len,
            dan membuang paket tersebut
}
else {
            hitung n, jumlah dari alamat-alamat dalam header
            Routing, dengan membagi Hdr Ext Len dengan 2

            if Segments Left lebih besar dari n {
                  mengirim suatu pesan ICMP Parameter Problem,
                  Code 0, ke Source Address, menunjuk pada field
                  Segments Left, dan membuang paket tersebut
            }
            else {
                  kurangi Segments Left dengan 1;
                  hitung i, index dari alamat selanjutnya yang
                  akan dilewati/dikunjungi dalam vektor alamat,
                  dengan mengurangkan Segments Left dari n
```

```
                    if Address [i] atau IPv6 Destination Address
                    adalah multicast {
                        membuang paket tersebut
                    }
                    else {
                        kosongkan IPv6 Destination Address dan
                        Addres [i]

                        if IPv6 Hop Limit kurang dari atau sama
                        dengan 1 {
                              mengirim suatu pesan ICMP Time
                              Exceeded – Hop Limit Exceeded in
                              Transit ke Source Address dan
                              membuang paket
                        }
                        else {
                              mengurangi Hop Limit dengan 1

                              mensubmit kembali paket tersebut
                              pada modul IPv6 untuk transmisi ke
                              Destination baru
                        }
                    }
               }
}
```

Kode II-1.Algoritma Dasar Routing

2.1.9. *Fragment Header*

Header Fragment digunakan oleh suatu *source* IPv6 untuk mengirim suatu paket yang lebih besar dari yang akan berada pada *path* MTU ke *Destination*-nya. (Catatan: tidak seperti IPv4, *Fragment*asi dalam IPv6 hanya dilakukan oleh *node source*, bukan oleh *router* sepanjang delivery *path* dari suatu paket) *Header Fragment* diidentifikasi dengan nilai 44 *Next Header* pada tepat sebelum *header*, dan memiliki format sebagai berikut :



Gambar II.7 Format Fragment Header

Keterangan gambar:

Next Header — Selektor 8-bit. Mengidentifikasi inisial tipe *header Fragment*able Part dari paket asli/awal (yang didefinisikan sebelumnya). Menggunakan nilai yang sama dengan *field* IPv4 Protocol.

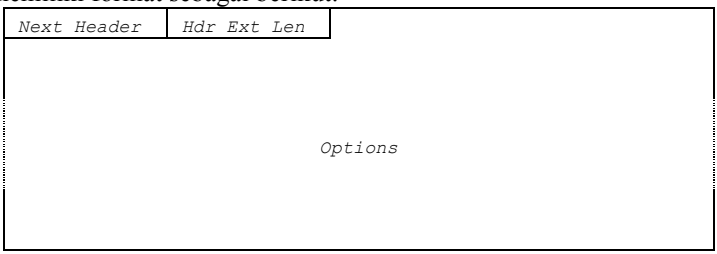
Reserved	*Field reserved* 8-bit. Diinisialisasi dengan nol untuk transmisi; diabaikan pada penerimaan.
Fragment Offset	13-bit *unsigned* integer. Offset, dalam satuan 8-oktet, dari data yang mengikuti/setelah *header* ini, relatif pada awal *Fragment*able Part dari paket mula-mula.
Res	*Field reserved* 2-bit. Diinisialisasi dengan nol untuk transmisi; diabaikan pada penerimaan.
M flag	1 = more *Fragment*s; 0 = last *Fragment*.
Identification	32 bit. Lihat deskripsi di atas.

Untuk mengirim suatu paket yang terlalu besar yang sesuai dengan *path* MTU ke *Destination*-nya, suatu *node source* boleh membagi paket tersebut menjadi *Fragment-Fragment* dan mengirim masing-masing *Fragment* sebagai paket terpisah, kemudian disatukan kembali pada penerima.

Untuk setiap paket yang akan dipecah-pecah, *node* asal harus membangkitkan nilai yang digunakan untuk mengidentifikasi paket yang dipecah tersebut. Dalam satu paket yang dipecah-pecah tersebut harus mempunyai nilai identifikasi yang berbeda. Jika *header routing* terdapat pada paket, tujuan alamat dikonsentrasikan pada tujuan terakhir.

2.1.10. *Destination Option Header* (*Header* Pilihan Tujuan)

Header Destination Options digunakan untuk membawa informasi opsional yang dibutuhkan hanya untuk diuji oleh *node Destination* dari suatu paket. *Header Destination Options* dikenali dengan nilai 60 pada *Next Header* dalam tepat sebelum *header*, dan memiliki format sebagai berikut:



Gambar II.8 Format Header Destination

Keterangan gambar:

Next Header	Selektor 8-bit. Mengidentifikasi tipe dari *header* tepat setelah *header Destination Options*. Menggunakan nilai yang sama dengan *field* IPv4 Protocol
Hdr Ext Len	8-bit *unsigned* integer. Panjang dari *header Destination Options* dalam satuan 8-oktet, tidak termasuk 8 oktet pertama/awal.
Options	*Field variable-length*, dari panjang seperti *header Destination Options* lengkap merupakan suatu integer yang panjangnya 8 oktet ganda. Terdiri dari satu atau lebih opsi TLV-encoded.

Perlu dicatat bahwa ada dua kemungkinan cara untuk meng-enkode informasi *Destination* opsional dalam suatu paket IPv6 baik seperti suatu opsi dalam *header Destination Options*, atau seperti *extension header* yang terpisah. *Header Fragment* dan *header* Authentication adalah contoh dari pendekatan akhir. Pendekatan mana yang dapat digunakan bergantung pada tindakan apa yang diinginkan dari *node Destination* yang tidak mengetahui informasi opsional :

a. Jika tindakan yang diinginkan adalah agar *node Destination* membuang paket tersebut dan , hanya jika *Destination Address* dari paket tersebut bukan *multicast Address*, mengirim suatu pesan ICMP Unrecognized *Type* pada *Source Address* dari paket tersebut, lalu informasinya mungkin di-enkode baik sebagai suatu *header* yang terpisah atau sebagai suatu opsi dalam *header Destination Options* yang memiliki *Option Type* bernilai 11 pada dua bit urutan teratasnya. Pilihan tersebut boleh tergantung pada faktor-faktor serupa sesuai yang mengambil oktet lebih sedikit, atau yang menghasilkan penjajaran yang lebih baik atau penguraian yang lebih efisien.
b. Jika aksi lain yang diinginkan, informasinya harus dienkode sebagai suatu opsi dalam *header Destination Options* yang memiliki *Option Type* bernilai 00, 01, atau 10 pada dua bit urutan teratas, yang menunjukkan aksi yang diinginkan.

2.1.11. *No Next Header*

Nilai 59 dalam *field Next Header* dari suatu *header* IPv6 atau *extension header* manapun menandakan bahwa tidak ada yang mengikuti *header* tersebut. Jika *field* Payload Length dari *header* IPv6 menandakan keberadaan dari oktet-oktet sebelum akhir dari suatu *header* yang memiliki *field Next Header* berisi 59, oktet-oktet tersebut harus diabaikan, dan dibiarkan tidak berubah jika paket tersebut diteruskan.

2.1.12. *Packet Size Issues* (Isu Ukuran Paket)

IPv6 mensyaratkan bahwa setiap *link* dalam Internet harus mempunyai MTU (*Maximum Transfer Unit*) 1280 oktet atau lebih. PAda setiap *link* yang tidak dapat membawa paket sebesar 1280 oktet dalam satu buah paket, spesifikasi *link* untuk *Fragment*asi dan penyatuan kembali harus disediakan oleh *layer* sebelum IPv6.

Link yang mempunyai MTU yang dapat dikonfigurasi (sebagai contoh, *link* PPP [RFC 1661]) harus dikonfigurasi untuk mempunyai MTU minimal 1280 oktet; dan direkomendasikan untuk dikonfigurasi dengan MTU 1500 oktet atau lebih, dimana hal ini untuk mengakomodasikan kemungkinan adanya enkapsulasi atau *tunneling* tanpa adanya *Fragment*asi pada *layer* IPv6.

Dari setiap *link* tempat *node* terhubung secara langsung *node* harus mampu menerima paket sebesar *link* MTU. Hal ini sangat direkomendasikan bahwa *node* IPv6 mengimplementasikan *discovery Path* MTU (RFC 1981), dalam rangka untuk mencari dan mengambil keuntungan dari adanya *path* MTU yang lebih besar dari 1280 oktet. Namn, implementasi minimum IPv6 (misalkan pada boot ROM) boleh melakukan pembatasan dengan membatasi pengiriman paket yang besarnya tidak lebih dari 1280 oktet, dan menghindari implementasi dari *discovery path* MTU.

Pada pengiriman paket yang lebih besar dari *path* MTU, *node* boleh menggunakan *header Fragment* IPv6 untuk paket memecah paket sudah sampai pada alamat tujuan. Bagaimanapun juga, penggunaan *Fragment*asi ini menggagalkan setiap aplikasi yang mampu menepatkan ukuran paket sesuai dengan *path* MTU.

Node harus mampu menerima paket yang ter*Fragment*asi, yang setelah disusun kembali, sebesar 1500 oktet. *Node* diijinkan untuk menerima paket yang ter*Fragment*asi dan telah disusun kembali lebih besar dari 1500 oktet. Protokol atau aplikasi pada level

diatas level IPv6 yang bergantung pada Fragmentasi IPv6 untuk mengirimkan paket yang lebih besar dari 1500 oktet kecuali *node* tersebut memastikan bahwa *node* tujuan mampu menyusun kembali paket yang mempunyai ukuran lebih besar.

2.1.13. *Flow Labels*

Kolom pelabelan aliran sepanjang 20 bit dalam *header* IPv6 mungkin digunakan oleh alamat sumber untuk menandai/mengklasifikasikan sekumpulan paket yang meminta penanganan khusus oleh *router* IPv6, seperti kualitas layanan (QoS) yang tidak sesuai standar atau layanan real-time. Aspek ini dalam IPv6, pada saat tulisan ini ditulis, masih dalam percobaan dan perubahan pokok yang sesuai dengan yang diperlukan untuk dukungan flow dalam Internet menjadi diperjelas. *Router* atau *host* yang tidak mendukung fungsi dari kolom label flow diminta untuk mengeset kolom tersebut menjadi nol ketika menjadi asal dari paket, dan melewatkan paket dengan ridak ada perubahan pada kolom, dan mengabaikan kolom tersebut ketika menerima paket.

2.1.14. Pengkelasan Trafik

Kolom kelas trafik 8 bit dalam *header* IPv6 tersedia untuk digunakan oleh *node* asal dan atau *routing* yang melewatkan paket untuk mengidentifikasikan dan membedakan antara kelas atau prioritas yang berbeda dari paket IPv6. Pada waktu tulisan ini ditulis, ada sejumlah eksperimen yang dilaksanakan dengan menggunakan tipe layanan dari IPv4 atau diawali dengan menyediakan bermacam-macam bentuk perbedaan layanan untuk paket IP. Kolom kelas trafik dalam *header* IPv6 diharapkan menyediakan fungsi yang sama untuk didukung dalam IPv6.

2.2. Ar*site*ktur Pengalamatan IP versi 6[2]

Alamat IPv6 adalah pengindetifikasi sepanjang 128 bit untuk *interface* dan sekumpulan *interface*. Ada tuga tipe dari alamat IPv6 :

a. *Unicast* : Pengidentifikasi untuk *interface* tunggal. Paket yang dikirimkan ke alamat unocast adalah paket yang dikirimkan ke sebuah *interface* yang diidentifikasi oleh alamat tersebut.

---

[2] R. Hinden, S. Deering, 1998, **IP Version 6 Addressing Architecture**, Request for Comments 2373

b. *Anycast* : Pengidentifikasi untuk sekumpulan *interface* (umumnya milik *node* yang berbeda). Paket yang dikirimkan ke alamat *anycast* adalah paket yang dikirimkan ke salah satu dari sekumpulan *interface* yang diidentifikasi oleh alamat tersebut (alamat yang paling dekat, mengacu pada pengukuran jarak dari protokol *routing*).
c. *Multicast* : pengidentifikasi untuk sekumpulan *interface* (umumnya milik *node* yang berbeda). Paket yang dikirimkan ke alamat *multicast* adalah paket yang dikirimkan ke semua *interface* yang diidentifikasi oleh alamt tersebut.

Tidak ada alamat broadcast dalam IPv6, fungsi alamat broadcast digantikan oleh alamat *multicast*.

2.2.1. Model Pengalamatan

Alamat-alamat IPv6 dari semua tipe diberikan pada *interface*, tidak pada *node*. Alamat *unicast* IPv6 mengacu pada *interface* tunggal. Karena setiap *interface* milik *node* tunggal, alamat *unicast* yang diberikan pada *node* tersebit juga digunakan untuk mengidentifikasi *node* tersebut.

Semua *interface* diharuskan untuk mempunyai setidaknya satu alamat *unicast link-local*. Satu buah *interface* dapat diberikan atau dialokasikan alamat IPv6 lebih dari satu dengan berbagai macam tipe alamat atau *scope.* Alamat *unicast* dengan *scope* lebih besar dari *link-scope* tidak diperlukan untuk *interface* yang tidak digunakan sebagai alamat asal atau tujuan dari paket IPv6. Hal ini kadang-kadang tepat untuk *interface* point-to-point, atau dalam bentuk *link* point-to-point, tidak perlu adanya pemberian alamat *unicast* pada kedua *interface* tersebut. Ada satu pengecualian pada model pengalamatan ini, yaitu alamat *unicast* atau sekumpulan ala‚at *unicast* mungkin diberikan ke *interface* fisik yang banyak jika implementasi tersebut menganggap *interface* yang banyak tersebut sebagai satu kesatuan *interface* ketika dihadapkan pada *layer* internet. Hal ini sangat berguna untuk load-sharing melalui *interface* fisik yang banyak.

Saat ini IPv6 melanjutkan model IPv4 dimana prefix subnet diasosiasikan dengan satu *link* (*link* tunggal). Prefix subnet yang mungkin diberikan pada *link* yang sama dapat lebih dari satu.

2.2.2. Representasi Teks dari Alamat

Ada tiga jenis bentuk konversional untuk merepresentasikan alamat IPv6 sebagai string teks :

1. Bentuk yang disukai adalah x:x:x:x:x:x:x:x, x adalah nilai heksadesimal dari 8 satuan yang mana setiap satuan terdiri atas 16 bit

   Contoh :

   FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
   1080:0:0:0:8:800:200C:417A

   Catatan :

   Tidak perlu menulis permulaan nilai nol dalam setiap kolom (dipisahkan dengan tanda ":"), misalkan 0008 cukup dapat dituli 8 saja. Namun, setidaknya harus ada satu dalam setiap kolom jika semuanya berupa 0.

2. Ada beberapa metode dalam pengalokasian gaya tertentu dari alamat IPv6, hal ini khususnya untuk alamat yang berisi string nol bit yang panjang. Dalam rangka untuk membuat mudah penulisan alamat yang berisi bit nol, special sintaks tersedia untuk memadatkan kumpulan dari tiap-tiap nilai nol sepanjang 16 bit yng berurutan. Tanda "::" hanya dapat tampil sekali dalam sebuah alamat. Tanda "::" juga dapat digunakan untuk memadatkan kumpulan nilai 16 bit yang terdapat pada awal alamat.

   Contoh :

1080:0:0:0:8:800:200C:417A	alamat *unicast*
FF01:0:0:0:0:0:0:101	alamat *multicast*
0:0:0:0:0:0:0:1	alamat *loopback*
0:0:0:0:0:0:0:0	alamat tak terdefinisi

   mungkin direpresentasikan menjadi:

1080::8:800:200C:417A	alamat *unicast*
FF01::101	alamat *multicast*
::1	alamat *loopback*
::	alamat tak terdefinisi

3. Bentuk alternative yang kadang-kadang lebih tepat ketika dihadapkan dengan lingkungan gabungan dari IPv4 dan IPv6 adalah x:x:x:x:x:x:d.d.d.d dimana x menandakan nilai heksadesimal dari enam satuan yang masing-masing terdiri atas 16 bit, dan d adalah nilai decimal dari empat satuan yang

masing-masing terdiri dari 7 bit (standar representasi IPv4). Contoh :

0:0:0:0:0:0:202.154.63.9
0:0:0:0:0:FFFF:10.122.1.77
atau dalam bentuk dipadatkan :
::202.154.63.9
::FFFF:10.122.1.77

2.2.3. Representasi Teks dari Alamat Prefix

Representasi teks dari alamat prefix sama dengan alamat prefix pada IPv4 yang ditulis dalam notasi CIDR (Classless Inter Domain *Routing*), alamat prefix IPv6 direpresentasikan degnan notasi berikut:

IPv6-*Address*/Prefix-length

IPv6-*Address* adalah alamat IPv6 dengan ketentuan notasi pengalamatan.
Prefix-length adalah nilai decimal yang menspesifikasikan berapa banyak bit yang berurutan disebelah kiri mulai dari awal bit yang termasuk dalam prefix.

Sebagai contoh, berikut ini representasi yang benar dari 60 bit prefix 12AB00000000CD3 (dalam heksa decimal) :
12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60

Berikut ini adalah representasi yang salah dari prefix diatas:
12AB:0:0:CD3/60

menghilangkan nilai nol yang berada di depan tiap kolom, namun tidak mencantumkan nol yang berada di belakang.
12AB::CD30/60

alamat di sebelah kiri "/" jika diperlukan akan menjadi 12AB:0000:0000:0000:0000:0000:0000:CD30
12AB::CD3/60

alamat disebelah kiri "/" jika diperlukan akan menjadi 12AB:0000:0000:0000:0000:0000:0000:CD3
Ketika menulis alamat *node* dan prefix dari alamat *node* tersebut, keduanya dapat dikombinasikan sebagai berikut:
Alamat *node* : 12AB:0:0:CD30:123:4567:89AB:CDEF

Nomer Subnet : 12AB:0:0:CD30::/60
Dapat disingkat sebagai :
12AB:0:0:CD30:123:4567:89AB:CDEF/60

2.2.4. Representasi Tipe Alamat

Tipe spesifik dari alamat IPv6 diindikasikan dengan bit-bit awal yang berada dalam alamat. Pangjang bit-bit awal yang bervariabel ini disebut format prefix (FP). Inisialisasi alokasi dari prefix-prefix ini ada sebagai berikut:

Alokasi	Prefix (biner)	Fraction of *Address* Space
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Aggregatable global *Unicast Address*	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Unassigned	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512
Link-local *Unicast Address*	1111 1110 10	1/1024
Site-local *Unicast Address*	1111 1110 11	1/1024

Multicast Address	1111 1111	1/256

Gambar II.9 Tabel Inisialisasi Alamat IPv6

Catatan:

1. "unspecified *Address*" di atas adalah alamat *loopback* dan alamat IPv6 yang digabungkan dengan alamat IPv4. Alamat-alamat tersebut diberikan alokasi dengan spasi format prefix 0000 0000.
2. Format prefix-prefix dari 001 sampai 111, kecuali alamat *multicast* (111 1111), semuanya distaratkan harus mempunyai pengudentifikasi *interface* 64 bit dalam format EUI-64 (panjang prefix maksumum adalah 64 bit dan 64 bit selanjutnya adalah pengidentifikasi *interface*).

Alamat *unicast* dibedakan dari alamat *multicast* dengan nilai octet berorder tinggi dalam alamat. Nilai FF (1111 1111) mengidentifikasikan alamat sebagai alamat *multicast*, nilai lain dari alamat adalah alamat *unicast*. Alamat *anycast* diambil dari spasi alamat *unicast*, dan secara sintaks tidak berbeda dari alamat *unicast*.

2.2.5. Alamat *Unicast*

Alamat *Unicast* merupakan alamat yang dapat diagregasi dengan masking pada bit-bit seperti pada alamat IPv4 di bawah CIDR (Class-less InterDomain *Routing*).

Ada beberapa bentuk pemberian alamat *unicast* dakam IPv6, termasuk alamat *unicast* global teragregatisasi, alamat NSAP, alamat hierarki IPX, alamat *site*-local, alamat *link*-local, dan alamat dengan gabungan IPv4. Tipe alamat tambahan dapat didefinisikan pada masa depan. *Node* IPv6 boleh mempunyai sedikit pengetahuan dari struktur internal dari alamat IPv6, tergantung pada peran tersebut dilaksanakan (misal di *host* atau *router*). PAda minimumnya, *node* boleh menganggap bahwa alamat *unicast* tidak mempunyai internal struktur:

Host yag sedikit pintar (tapi masih cukup sederhana) dapat sedikit perhatian pada prefix subnet pada *link* tempat *host* tersebut tersambungkan. Alamat yang berbeda mempunyai nilai-nilai yang berbeda, missal untuk n di bawah ini :

Host yang lebih pintar lagi mungkin perhatian pada batasan hirarki dalam alamat *unicast*. Meskipun pada *router* yang kecil tidak

mengetahui struktur internal alamat IPv6, *router* pada umumnya mempunyai pengetahuan satu atau lebih batasan structural alamat *unicast* untuk operasi dari protokol *routing*.

2.2.5.1. Pengidentifikasi *Interface*

Pengidentifikasi *interface* dalam alamat *unicast* IPv6 digunakan untuk mengidentifikasi *interface* pada *link*. Mereka diharuskan untuk unik pada *link* tersebut. Mereka mungkin juga unik pada cope yang besar. Pada banyak kasus pengidentifikasi *interface* akan sama dengan alamat *interface* pada *layer link* atau *layer* 2. Pengidentifikasi *interface* yang sama mungkin digunakan pada *interface-interface* pada *node* tunggal.

Dalam sejumlah format prefix, *interface* id disyaratkan mempunyai panjang 64 bit dan disusun dalam format IEEE EUI-64. Pnegidentifikasi *interface* berbasis EUI-64 boleh mempunyai scope global ketika terdapat token global (misalnya MAC, Media Access Control, IEEE 48 bit atau biasa disebut alamat MAC) atau boleh mempunyai scope local ketika tidak terdapat token global (seperti serial *link tunnel* end-point).

2.2.5.2. Alamat Tak Terspesifikai

Alamat 0:0:0:0:0:0:0:0 disebut alamat yang tak terspesifikasi. Alamat tersebut tidak boleh diberikan pada setiap *node*. Alamat tersebut mengidentifikasikan kehadiran alamat. Salah satu contoh penggunaannya adalah di dalam kolom alamat asal dari setiap paket IPv6 yang dikirim oleh *host* penginisialisasi sebelum *host* tersebut telah mempelajari alamatnya sendiri.

Alamat yang tak terspesifikasikan tidak boleh digunakan sebagai alamat tujuan dari paket IPv6 atau dalam *header routing*.

2.2.5.3. Alamat *Loopback*

Alamat *unicast* 0:0:0:0:0:0:0:1 disebut alamat *loopback*. Alamat tersebut mungkin digunakan oleh *node* untuk mengirimkan paket IPv6 ke dirinya sendiri. Alamat tersebut tidak pernah diberikan pada setiap *interface* fisik. Hal ini mungkin dihubungkan dengan intergace virtual (misalkan *interface loopback*).

Alamat *loopback* tidak boleh digunakan sebagai alamat asal dalam paket IPv6 yang dikirimkan keluar *host*. Paket IPv6 dengan

alamat tujuan *loopback* tidak boleh dikirim keluar dari *host* dan tidak dapat diforwardkan/dilewatkan oleh *router* IPv6.

2.2.5.4. Alamat IPv6 Digabung engan Alamat IPv4

Mekanisme transisi IPv6 memasukkan teknik untuk *host* dan *router* secara dinamik men*tunnel* paket IPv6 melalui infrastruktur *routing* IPv4. *Node* IPv6 yang menggunakan teknik ini diberikan alamat *unicast* IPv6 spesial yang membawa alamat IPv4 dalam low order 32-bit. Tipe alamat ini diistilahkan "IPv4-compatible IPv6 *Address*".

Tipe kedua dari alamat IPv6 dimana terdapat gabungan dengan alamat IPv4 juga didefinisikan. Alamat ini diginakan untuk merepresentasikan *node* dengan alamat IPv4 saja (dimana tidak mendukung IPv6) sebagai alamat IPv6. Tipe alamat ini diistilahkan "IPv4-mapped IPv6 *Address*".

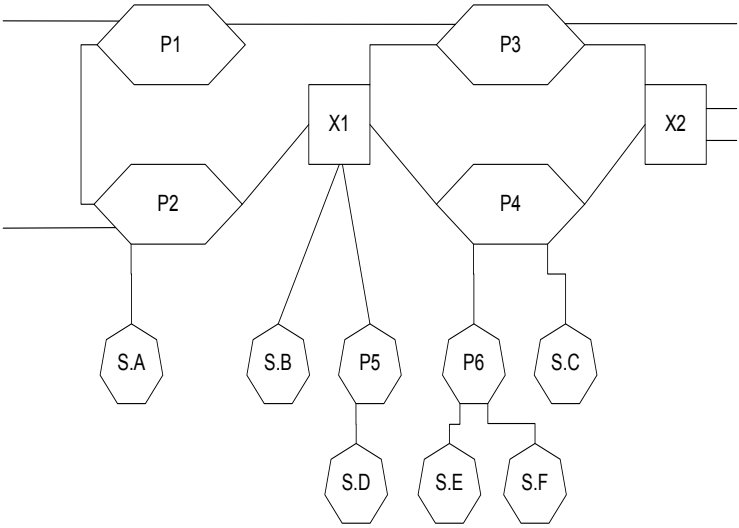
2.2.5.5. Alamt Global *Unicast* yang Dapat Teragregasi

Format alamat ini didesain untuk mendukung dua tipe agregasi, yaitu tipe agregasi yang saat ini digunakan oleh provider dan tipe baru dari agregasi yang disebut *exchanges.*Kombinasi akan membuat agregasi *routing* lebih efisien untuk kedia *site* yang terkoneksi secara langsung ke provider dan yang terkoneksi ke exchanges. *Site* akan mempunyai pilihan untuk melakukan koneksi ke entitas agregasi manapun (provider maupun exchanges).

Alamt yang dapat diagregasi ke dalam tiga tingkat hirarki:

a. *Public Topology*
b. *Site Topology*
c. *Interface Identifier*

Topologi public adalah koleksi dari provider dan exchange yang menyediakan layanan transit pada internet public. *Topologi site* adalah spesifik *site* local atau organisasi yang tidak menyediakan layanan transit Internet pada *node* keluar dari *site*nya. Pengidentifikasi *interface* mengidentifikasi *interface* pada *link*.



Gambar II.10 Format alamat yang dapat diagregasi

Seperti yang ditunjukkan pada gambar diatas, format alamat yang dapat diagregasi didesain untk mendukung provider besar (ditunjukkan pada P1, P2, P3, dan P4), exchange (ditunjukkan sebagai X1 dan X2), provider berbagai tingkat (ditunjukkan pada P5 dan P6) dan subscriber (ditunjukkan sebagai S.x).

Format Alamat Global *Unicast* yang dapat teragregasi adalah sebagai berikut:

3	13	6	24	16	64 bits
FP	TLA ID	RES	NLA ID	SLA ID	*Interface* ID

Public Topology — *Site* Topology — *Interface* Identifier

Gambar II.11 Format Agregasi pada Alamat Global Unicast

Dimana :

001

Format prefix(3 bit) untuk alamat global *unicast* yang dapat teragregasi

LA ID

Pengidentifikasi agregasi Top-Level

RES

Dicadangkan untuk penggunaan masa yang akan datang

NLA ID

Pengidentifikasi agregasi *Next* Level

SLA ID

Pengidentifikasi agregasi *Site*-Level

Interface ID

Pengidentifikasi *interface*

***Top-Level Agregation Identifier* (TLA ID)**

Top-Level Agregation Identifier (TLA ID) adalah tingkat tertinggi dalam hirarki *routing*, *Router* yang tanpa adanya rute default harus mempunyai entri tabel *routing* untuk setiap TLA ID yang aktif dan meungkin akan mempunyai entri tambahan yang menyediakan informasi *routing* untuk TLA ID yempay mereka berada. *Router*-*Router* yersebut boleh mempunyai entri tambahan untuk mengoptimalisasi *routing* pada semua level harus didesain untuk meminimalkan jumlah entri tambahan yang ada pada tabel *routing* yang bebas rute default (tidak adanya entri rute default pada *router*).

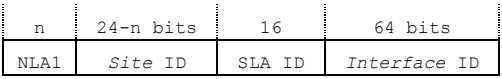
Dengan metode pengalamatan ini, TLA ID yang terdapat pada tabel *routing*, setiap *router* terutama *router* exchange hanya sampai 8.192(2^13) entri. Jika dibandingkan dengan tabel *routing* yang terdapat pada *router*-*router* exchange untuk Ipv4 saat ini, jumlah 8.192 sangatkan kecil karena tabel *routing* pada *router* exchange saat ini sudah mencapai 100.000 entri. Tambahan TLA ID mungkin ditambahkan ketika jumlah yang saat ini sudah dipakai dan penuh dengan menggunakan kolom yang dicadangkan.

Reserved

Kolom *reserved* dicadangkan untuk penggunaan masa depan dan harus di set ke nol. Kolom ini dapat digunakan untuk pertumbuhan kedepan dari kolom TLA dan NLA yang memerlukan tambahan bit.

Next-Level Agregation Identifier

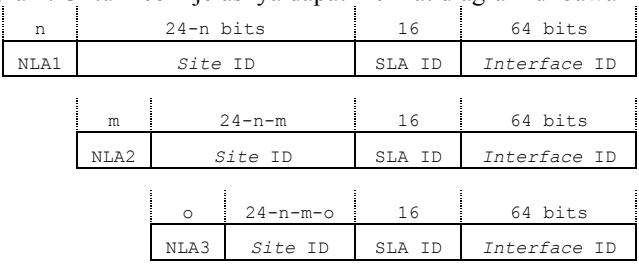
Next-Level Agregation Identifier digunakan oleh organisasi yang diberikan TLA ID untuk membuat hirarki pengalamatan dan untuk mengidentifikasi *site*nya. Organisasi dapat memberikan bagian tertinggi dari NLA ID dengan membuat hirarki pengalamatan yang tepat ke networknya. Organisasi dapat menggunakan sisa bit dalam kolom NLA-ID untuk mengidentifikasi *site* yang ingin dilayani. Hal ini ditunjukkan sebagai berikut:



Gambar II.12 Format Next-Level Agregation Identifier

Setiap organisasi yang diberikan oleh TLA ID menerima spasi 24 bit NLA id. Spasi NLA ID yang didapat oleh organisasi ini mengizinkan setiap organisasi tersebut dapat untuk menyediakan layanan kira-kira sebanyak semua organisasi yang saat ini dilayani oleh Internet dengan IPv4.

Organisasi yang diberikan oleh TLA ID boleh juga mendukung NLA ID dalam spasi *Site* ID mereka sendiri. Hal ini membuat organisasi yang diberikan oleh TLA ID dapat menyediakan layanan ke organisasi yang menyediakan layanan transit publik dan untuk mengorganisasi siap saja yang tidak menyediakan layanan transit publik. Organisasi-organisasi yang menerima NLA ID ini boleh juga menggunakan spasi *site* ID yang diberikan ke NLA ID yang lain. Untuk lebih jelasnya dapat melihat diagram di bawah ini:



Gambar II.13 Contoh Pengalokasian NLA-ID

Desain layout bit dari spasi NLA ID untuk TLA ID tertentu diserahkan kepada organisasi yang telah diberikan atau memiliki

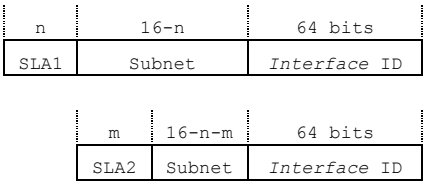
TLA ID. Demikian juga desain layout bir dari NLA ID berikutnya (misal NLA2 atau NLA3) adalah tanggung jawab dari level NLA ID sebelumnya.

Desain rencana alokasi NLA ID adalah perbandingan antara efisiensi agregasi *routing* dan fleksibelitas. Dengan membuat modek hirarki, akan mengizinkan kemampuan agregasi *routing* yang besar sehingga menghasilkan tabel *routing* yang kecil. Pemberian NLA ID ke provider secara flat akan membuat kemudahan dalam alokasi dan lebih fleksibel tapi menghasilkan banyak entri dalam *routing* tabel.

Site-Level Agregation Identifier

Kolom SLA ID digunakan oleh organisasi individual untuk membuat hirarki pengalamatan lokalnya dan untuk mengidentifikasi subnet-subnet. Hal ini sama dengan subnet dalam Ipv4 kecuali setiap organisasi mempunyai jumlah subnet yang sangat banyak daripada Ipv4. Kolom 16 bit pada SLA ID mampu membuat 65.535 subnet.

Organisasi dapat memilih pembagian SLA ID apakah dengan sistem flat/merata (dengan tidak membuat beberapa sub SLA ID sehingga mengakibatkan tabel *routing* yang besar) atau dengan membuat dua atau lebih tingkatan hirarki dalam kolom SLA ID (yang menghasilkan tabek *routing* yang kecil). Pilihan yang terkahir daoat dilihat berikut:



Gambar II.14 Contoh Pengalokasian SLA-ID

Pilihan strukturisasi kolom SLA ID yang akan digunakan adalah tanggung jawab dari organisasi-organisasi tersbut secara individual yang mendalatkan otoritas atas SLA tersebut.

Jumlah subnet yang didukung dalam format alamat ini harus cukup untuk semua termasuk organisasi yang terbesar. Oraganisasi yang membutuhkan subnet-subnet provider dimana mereka mendapatkan layanan untuk memndapatkan *site* identifier

tambahan dan menggunakan *site* identifier tersebut untuk membuat subnet tambahan.

2.2.5.6. Penggunaan Secara Lokal Alamat *Unicast* IPv6

Ada dua tipe pengguanaan secara lokal alamat *unicast* IPv6 digunakan. Tipe itu adalah *link*-local dan *site*-local. *Link* local adalah untuk pengguanaan pada *link* tunggal dan *Site*-local adalah untuk penggunaan dalam *site* tunggal. Alamat-alamat *link*-local mengukuti format berikut:

Alamat-alamat *link*-local didesain untuk digunakan untuk pengalamatan pada *link* tunggal untuk tujuan seperti konfigurasi pengalamatan secara otomatis, pencarian tetangga, atau ketika tidak terdapat *router*.

Router harus tidak menforwardkan semua paket yang berasal dari *link*-local atau paket dengan alamat tujuan *link*-local ke *link* lain.

Alamat-alamat *site*-local mempunyai format berikut :

Alamat-alamat *site*-local didesain untuk digunakan untuk pengalamatan dalam *site* yang tapa memerlukan global prefix. *Router* jarus tidak melewatkan semua paket yang berasal dari *site*-local atau paket dengan alamat tujuan *site*-local ke *site* lain.

2.2.6. Alamat *Anycast*

Alamat *Anycast* IPv6 adalah alamat yang diberikan kepada lebih dari satu *interface* (biasanya milik *node* yang berbeda), paket yang dikirim ke alamat *anycast* dirutekan ke *interface* terdekat yang mempunyai alamat tersebut, mengacu pada pengukuran jarak dari protokol *routing*.

Alamat *anycast* dialokasikan dari spasi alamat *unicast*, menggunakan berbagai macam definisi format alamat berbeda dengan alamat *unicast*. Ketika alamat *unicast* berubah ke alamat *anycast*, *node* yang diberikan alamat tersebut harus secara eksplisit tahu bahwa alamat tersebut adalah alamat *anycast*

Satu yang diharapkan dari penggunaan alamat-alamat *anycast* adalah untuk mengidentifikasi sekumpulan *router* yang menjadi milik organisasi yang menyediakan layanan Internet. Alamat tersebut dapat digunakan sebagai alamat pertengahan dalam *header routing* IPv6, untuk menyebabkan paket dikirim melalui agregasi khusus atau urutan untuk mengidentifikasi sekumpulan

router yang terkoneksi dalam subnet tertentu atau mengidentifikasi sekumpulan *router* yang menyediakan entri ke dalam domain *routing* tertentu.

Ada beberapa larangan yang diberikan pada alamat *anycast*:

a. Alamat *anycast* harus tidak digunakan sebagai alamat asal dari paket IPv6.
b. Alamat *anycast* harus tidak diberikan kepada *host* IPv6 dengan demikian mungkin diberikan ke *router* IPv6 saja

2.2.7. Alamat *Multicast*

Alamat *multicast* IPv6 adalah pengidentifikasi untuk sekumpulan *node*. *Node* dapat menhadi anggora sejumlah group *multicast*. Alamat *multicast* mempunyai format :

11111111 pada awal alamat tersebut menunjukkan jika alamat tersebut adalah alamat multicst. Flags adalah sekumpulan dari 4 bit flags yang mempunyai format berikut

Tiga urutan terdepan dicadangkan, dan harus diberi niali nol.

T=0 mengidentifikasikan alamat*multicast* yang diberikan secara permanen (“well-known”). Pemberian alamat ini dilakukan ileh badan Internet global yang berwenang pada masalah penomoran

T=1 menandakan alamat *multicast* yang diberikan secara non-permanen(transien)

Scope adalah nilai jangkuan*multicast* sepanjang 4 bit digunakan untuk membatasi jangkauan group *multicast*.

Nilai ini adalah:

1. 0 dicadangkan
2. 1 jangkaunan *node*-lokal
3. 2. Jangkuan *link* local
4. 3 belum diberikan
5. 4 belum diberikan
6. 5 Jangkauan *Site* local
7. 6 belum diberikan
8. 7 belum diberikan
9. 8 Jangkuan organizaziton-local
10. 9 Belum diberikan
11. A belum diberikan
12. B belum diberikan

13. C Belum diberikan
14. D Belum diberikan
15. E Jangkuan global
16. F dicadangkan

Group ID mengidentifikasi group *multicast*, apakah diberikan secara permanen atau transien, dalam jangkuan yang diberikan.

Maksud dari alamat *multicast* yang diberikan secara permanen adalah nilai jangkuan yang independen. Sebagai contoh, jika group server NTP diberikan alamat *multicast* permanen dengan ID group 101 (dalam heksa), maka:

FF01:0:0:0:0:0:0:101 berarti semua server NTP pada *node* yang sama sebagai pengirim.

FF02:0:0:0:0:0:0:101 berarti semua server NTP pada *link* yang sama sebagai pengirim.

FF05:0:0:0:0:0:0:101 berarti semua server NTP pada *site* yang sama sebagai pengirim.

FF0E:0:0:0:0:0:0:101 baerarti semua server NTP dalam internet.

Alamat *multicast* yang diberikan secara nonpermanen berarti hanya dalam jangkauan yang diberikan. Sebagai contoh, group yang diidentifikasikan oleh alamat *multicast site*-local nonpermanen FF15:0:0:0:0:0:0:101 pada suatu *site* tidak berhubungan dengan group yang menggunakan alamat yang sama pada *site* yang berbeda, atau juga tidak berhubungan terhadap group non permanen yang berada pada jangkauan yang berbeda, atau juga tidak berhubungan dengan group permanen dengan group ID yang sama.

Alamat *multicast* harus tidak digunakan sebagai alamat asal dalam paket IPv6 atau terdapat dalam semua *header routing*.

Berikut ini adalah alamat *multicast* well-konwn yang sudah didefinisikan :

*Reserved Multicast Address*es:

FF00:0:0:0:0:0:0:0
FF01:0:0:0:0:0:0:0
FF02:0:0:0:0:0:0:0
FF03:0:0:0:0:0:0:0
FF04:0:0:0:0:0:0:0
FF05:0:0:0:0:0:0:0
FF06:0:0:0:0:0:0:0
FF07:0:0:0:0:0:0:0

FF08:0:0:0:0:0:0:0
FF09:0:0:0:0:0:0:0
FF0A:0:0:0:0:0:0:0
FF0B:0:0:0:0:0:0:0
FF0C:0:0:0:0:0:0:0
FF0D:0:0:0:0:0:0:0
FF0E:0:0:0:0:0:0:0
FF0F:0:0:0:0:0:0:0

Alamat *multicast* ini dicadangkan dan tidak boleh diberikan pada semua group *multicast*

All *Node*s *Address*es: FF01:0:0:0:0:0:0:1
FF02:0:0:0:0:0:0:1

Alamat *multicast* ini mengidentifikasi group dari semua *node* IPv6, dalam scope 1 (*node*-local) atau 2 (*link*-local).

All *Router*s *Address*es: FF01:0:0:0:0:0:0:2
FF02:0:0:0:0:0:0:2

Alamat *multicast* ini mengidentifikasi group dari semua *router*, dalam scope 1 (*node*-local), atau 2 (*link*-local) atau 5(*site*-local)

DHCP Server/Relay-Agent: FF02:0:0:0:0:0:0:C

Alamat *multicast* ini mengidentifikasi groups dari semua IPv6 DHCP server dan Relay Agent, dalam scope 2 (*link*-local)

Solicited-Node Address: FF02:0:0:0:0:1:FFXX:XXXX

Solicited-node Address adalah alamat yang diperhitungkan sebagai fungsi alamat *unicast* dan *anycast node*. Alamat ini dibentuk dengan mengambil 24 bit terakhir dari alamat (*unicast* atau *anycast*) dan menambahkan 24 bit terakhir yang diambil tersebut pada prefix FF02:0:0:0:0:1:FF00::/104.

Prefix FF02:0:0:0:0:1:FF00::/104 ini menghasilkan alamat *multicast* dalam jangkauan FF02:0:0:0:0:1:FF00:0000 sampai FF02:0:0:0:0:1:FFFF:FFFF

Sebagai contoh, alamat *multicast* model ini yang berhubungan dengan alamat IPv6 2001:200:830::200E:8C6C adalah FF02::1:FF0E:8C6C

2.2.7.1. Alamat yang Diperlukan *Node*

Host diharuskan untuk mengenali alamat-alamat berikut sebagai perngidentifikasi untuk diri sendiri:

a. Alamat *link*-local untuk setiap *interface*
b. Alamat *unicast* yang diberikan
c. Alamat *loopback*
d. Alamat *multicast solicited-node* untuk setiap alamat *unicast* dan *anycast* yang diberikan
e. Alamat *multicast* dari semua group tempat *host* menjadi anggota

Router diharuskan untuk mengenali semua alamat seperti yang diperlukan pada *host*, ditambah alamat-alamat berikut sebagai pengidentifikasi untuk diri sendiri:

a. Alamat subnet *anycast router* untuk *interface-interface* yang dikonfigurasi untuk bertindak sebagai *router*.
b. Semua alamat *anycast* lain yang sudah dikonfigurasi pada *router*.
c. Alamat *multicast* semua *router*
d. Alamat *multicast* dari semua group lain tempat *router* menjadi salah satu anggotanya

Alamt prefix-prefix yang harus didefinisikan sebelumnya pada implementasi IPv6 adalah:

a. Unspecified *Address*
b. Alamat *loopback*
c. Prefix *Multicast* (FF)
d. Prefix local-use (*link*-local dan *site*-local)
e. Pre-defined *multicast Address*es
f. IPv4-Compatible prefix

Implementasi harus mengasumsikan semua alamat sebagai *unicast* kecuali ada konfigurasi yang sudah dispesifikasikan.

2.3. Spesifikasi Internet Control Message Protocol (ICMPv6) untuk Internet Protocol Version 6 (IPv6)[3]

Internet Protocol (IP) versi 6 (IPv6) adalah versi baru dari IP. IPv6 menggunakan Internet Control Message Protocol (ICMP)

---

[3] A. Conta, S. Deering, 1998, **Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,** Request for Comments 2463

seperti yang didefinisikan untuk IPv4 dalam RFC 793 dengan sejumlah perubahan. Hasil dari protokol ini disebut ICMPv6 dan mempunyai nilai 58 pada *next header* IPv6.

2.3.1. ICMPv6 (ICMP untuk IPv6)

ICMPv6 digunakan oleh *node* IPv6 untuk melaporkan kesalahan dalam pengolahan paket dan untuk memberikan pada fungsi *layer* Internet lainnya, seperti untuk diagnosa (ICMPv6 "ping"). ICMPv6 adalah bagian yang utuh dari IPv6 dan harus diimplementasikan penuh oleh setiap *node* IPv6.

2.3.1.1. Format Umum Pesan

Pesan-pesan ICMPv6 dikelompokkan dalam dua kelas yaitu pesan kesalahan dan pesan informasional. Pesan kesalahan diidentifikasi dengan mempunyai nilai nol pada bit-bit awal dari kolom tipe pesan. Sehingga, pesan kesalahan mempunyai nilai tipe-tipe pesan dari 0 sampai 127; pesan informasi mempunyai nilai tipe-tipe pesan 128 sampai 255.

Format untuk pesan ICMPv6 :

Pesan kesalahan ICMPv6:

1. *Destination unreachable*
2. *Packet Too Big*
3. *Time Exceeded*
4. *Parameter Problem*

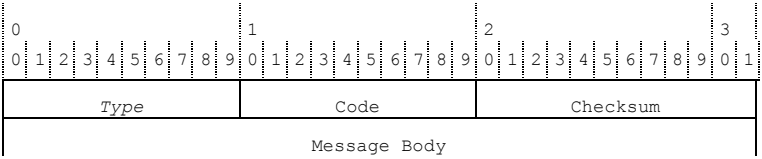
Pesan Informasioanl ICMPv6:

128 Echo Request

129 Echo Reply

Setiap pesan ICMPv6 diawali oleh *header* IPv6 dan nol atau lebih *header* tambahan. *Header* ICMPv6 diidentifikasi oleh *next header* dengan nilai 58 dalam yang terdapat pada *header* sebelum *header* ICMPv6.

Pesan-pesan ICMPv6 mempunyai format umum seperti pada gambar II.15:



Gambar II.15 Format Pesan ICMPv6

Kolom tipe mengidentifikasikan tipe dari pesan. Nilai ini menentukan format sisa data.
Kolom kode tergantung pada tipe pesan. Ini digunakan untuk membuat tingkatan tambahan ukuran pesan.
Kolom checksum digunakan untuk mendeteksi data yang hilang/rusak dalam pesan ICMPv6 dan bagian dari *header* IPv6.

2.3.1.2. Penentuan Asal Alamat Pesan

Node yang mengirimkan pesan ICMPv6 harus menentukan kedua alamat IPv6 asal dan tujuan dalam *header* IPv6 *header* sebelumnya menghitung checksum. Jika *node* mempunyai lebih dari satu alamat *unicast*, *node* tersebut harus memilih alamat asal dari pesan sebagai berikut:

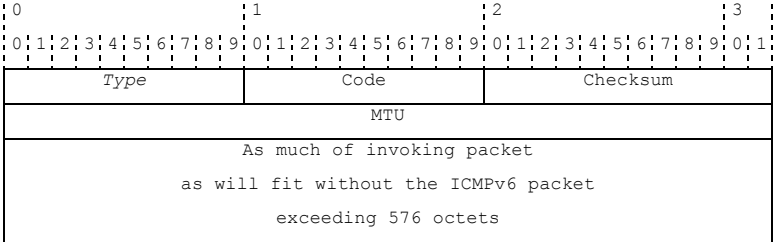
1. Jika pesan yang akan dikirim adalah respon ke pesan yang dikirim ke salah satu dari alamat *unicast node*, alamat asal dari pesan yang akan dibalaskan harus alamat yang sama dengan alamat tujuan dimana pesan tesebut dikirim (ke alamat *unicast node*). Contoh, jika pesan ICMPv6 dikirim ke a, maka alamat asal dari pesan balasan adalah a.
2. Jika pesan yang akan dikirim adalah respon dari pesan yang dikirim ke group *multicast* atau *anycast* yang terdapat *node-node* sebagai anggora, alamat asal dari pesan balasan harus alamat *unicast* milik *interface node* yang menerima paket *multicast* atau *anycast*.
3. Jika pesan adalah respon pesan yang dikirim ke alamat yang tidak milik dari *node*, alamat asal harus alamat *unicast* milik *node* yang akan sangat membantu dalam mempelajari kesalahan. Sebagai contoh, jika pesan adalah respon ke

suatu tindakan penerusan paket (dalam *router*) yang tidak dapat berlangsung secara sukses, alamat asal harus alamat *unicast* milik *interface* dimana tindakan penerusan paket gagal.

4. Jika tidak, tabel *routing* dari *node* harus dipelajari guna menentukan *interface* ana akan digunakan untuk mengirimkan pesan ke tujuannya dan alamat *unicast* milik *interface* tersebut harus digunakan sebagai alamat asal dari pesan.

2.3.2. Pesan Error ICMPv6

2.3.2.1. Pesan *Destination* Unreachable



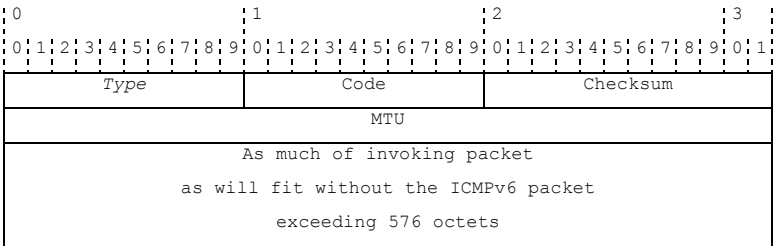
Gambar II.16. Format Pesan Destination Unreachable

Kolom IPv6 : alamat tujuan dilihat dari kolom alamat asal yang terdapat dari paket
Kolom ICMPv6:

Tipe	1
Kode	0 - tidak ada rute ke tujuan 1 - komunikasi dengan tujuan secara adminitrasi tidak diperbolehkan 2 - (belum diberikan) 3 - *Address* unreachable 4 - port unreacheable
Unused	kolom ini tidak digunakan untuk semua nilai kode. Kolom ini harus diinisialisasi nol oleh pengirim dan diabaikan oleh penerima

Deskripsi:
Pesan *Destination* unreachable seharusnya dihasilkan oleh *router* atau *layer* IPv6 dalam *node* asal, untuk memberitahu bahwa paket tidak dapat dikirimkan ke alamat tujuan untuk asalan selain kongesti. Pesan ICMPv6 tidak harus dihasilkan jika paket drop/hilang karena kongesti.
Jika alasan kesalahan pengiriman adalah karena ketidaktepatan antara alamat tujuan dengan tabel *routing* dalam perutean ke arah *node*, kolom kode diset ke 0.
Jika alasan untuk kesalahan pengirim adalah larangan secara administratif seperti filter firewall, kolom kode diset 1.
Jika terdapat alasan lain dalam kesalahan pengiriman seperti ketidakmampuan untuk memutuskan alamat tujuan IPv6 ke dalam alamat *link* yang sesuai, atau *link* yang secara spesifik digunakan mengalami masalah yang singkat maka kolom kode diset ke 3.
Node tujaun harus mengirim pesan *Destination* unreachable dengan kode 4 untuk merespon jika paket dengan tujuan protokol transport (misal UDP) tidak mendengarkan atau merespon, atau jika protokol transport tersebut tidak mempunyai alternatif lain untuk menginformasikan pengirim.
Node yang menerima pesan ICMP *Destination* unreachable harus memberitahu pada proses *layer* di atasnya.

2.3.2.2. Pesan Packet Too Big



Gambar II.17 Format Pesan ICMPv6 Packet Too Big

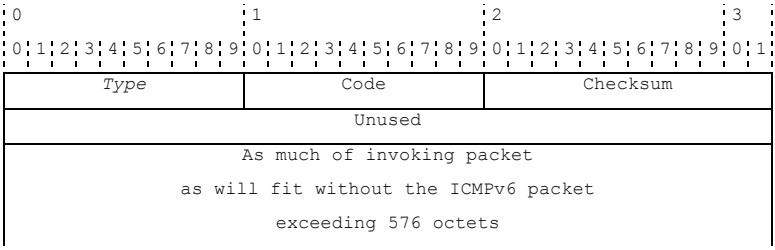
Kolom IPv6: Alamat tujuan, dilihat dari kolom alamat asal yang terdapat dari paket.

Kolom ICMPv6:

Tipe	2
Kode	Di set ke 0 (nol) oleh pengirim dan diabaikan oleh penerima
MTU	Maximum Transmission Unit dari *link* hop selanjutnya

Deskripsi:
Pesan ICMPv6 paket terlalu besar harus dikirim oleh *router* dalam merespon paket yang tidak dapat diteruskan karena paket lebih besar dari MTU dari *link* yang akan dilaluinya. Informasi dalam pesan ini digunakan sebagai bagian dari proses penemuan jalut MTU.
Pengirim pesan paket terlalu besar membuat saru perkecualian ke salah satu aturan yang terdapat pada pengiriman pesan kesalahan ICMPv6. Tidak seperti pesan lain, pesan ini dikirim dalam merespon paket yang diterima dengan alamat tujuan *multicast*, atau alamat *layer layer link multicast* atau broadcast.
Pesan paket yang terlalu besar yang dapat harus diteruskan ke proses di *layer* yang lebih atas.

2.3.2.3. Pesan Waktu Habis



Gambar II.18 Format Pesan ICMPv6 Time Exceeded

Kolom IPv6: Alamat tujuan dilihat dari kolom alamat asal yang terdapat dari paket.

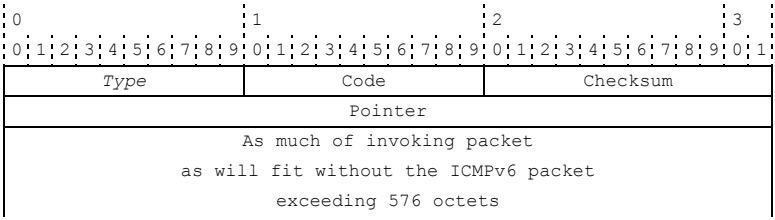
Kolom ICMPv6:

Tipe	3
Kode	0 - batasan hop terlampau dalam transit

	1 - waktu penyusun *Fragment* terlampaui
Unused	Kolom ini tidak digunakan untuk semua nilai kode. Kolom ini harus diinisialisasi nol oleh pengirim dan diabaikan oleh penerima

Deskripsi:
Jika *router* menerima paket dengan batas hop sama dengan nol, atau *router* mengurangi limit hop sama dengan nol, *router* harus membuang paket tersebut dan mengirimkan pesan ICMPv6 bahwa batasan waktu telah terlampau dengan kode 0 ke alamat asal dari paket tersebut. Hal ini menandakan terjadinya *routing* yang berputar atau inisialisasi nilai batas hop terlalu kecil. Pesan ICMPv6 batas waktu terlampaui yang datang harus diteruskan ke proses pada *layer* di atasnya.

2.3.2.4. Pesan Masalah Parameter



Gambar II.19 Format Pesan ICMPv6 Masalah Parameter

Kolom IPv6: alamat tujuan, dilihat dari kolom alamat asal yang terdapat dari paket.
Kolom ICMPv6

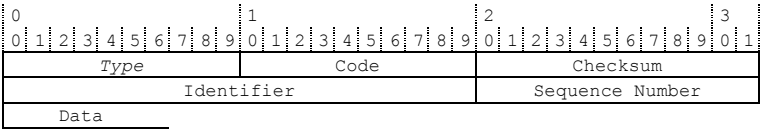
Tipe	4
Kode	0 - ditemukan kolom *header* yang keliru 1 - ditemukan tipe *next header* yang tidak dikenali 2 - ditemukan *Option* IPv6 yang tidak dikenali
Pointer	Mengidentifikasi penggantian oktet dalam paket yang terlihat ketika ditemui error

Deskripsi:
Jika dalam pemrosesan paket IPv6, *node* menemukan masalah dengan kolom dalam *header* IPv6 atau pada *header* tambahan,

misalkan masalah ini adalah *node* tidak dapat melakukan pemrisesan paket sampai selesai. *Node* harus membuang paket dan sebaiknya mengirimkan pesan ICMPv6 masalah parameter ke asal paket, untuk memberitahu tempat dan tipe masalah yang ada.

2.3.3. Pesan-Pesan Informational ICMPv6

2.3.3.1. Pesan Echo Request



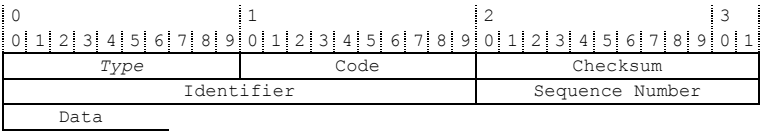
Gambar II.20 Format ICMPv6 Echo Request

Kolom IPv6: alamat tujuan semua alamat IPv6 yang legal
Kolom ICMPv6:

Tipe	128
Kode	0
Pointer	Pengidentifikasi untuk membantu menempatkan pengembalian echo ke permintaan echo ini, Mungkin nol
Sequence	Rangkainan nomer untuk membantu dalam penempatan pengembalian echo ke permintaan echo. Mungkin nol
Data	Nol atau beberapa oktet dari data yang berubah ubah

Deskripsi:
Setiap *node* harus mengimplementasikan fungsi pengresponan echo ICMPv6 ketika menerima permintaan echo dengan mengirimkan pengembalian echo yang diminta. *Node* dapat juga mengimplementasikan permintaan echo dan menerima pengembalian echo untuk tujuan diagnosa.

2.3.3.2. Pesan Echo Reply



Gambar II.21 Format ICMPv6 Echo Reply

Kolom IPv6: Alamat tujuan dilihar dari kolom alamat asal yang terdapat dari paket.

Kolom ICMPv6:

Tipe	129
Kode	0
Pointer	Pengidentifikasi untuk membantu menempatkan pengembalian echo ke permintaan echo ini, Mungkin nol
Sequence	Rangkainan nomer untuk membantu dalam penempatan pengembalian echo ke permintaan echo. Mungkin nol
Data	Nol atau beberapa oktet dari data yang berubah ubah

Deskripsi:
Setiap *node* yang menerima permintaan echo harus mengimplementasi fungsi peresponan echo ICMPv6 dan mengirimkan pengembalian echo yang sesuai. *Node* juga harus mengimplementasikan *interface layer* aplikasi untuk pengiriman request echo dan menerima pengembalian echo, untuk keperluan diagnostik.
Alamat asal dari Echo reply yang digunakan untuk merespon permintaan pesan Echo request harus sama dengan alamat tujuan dari pesan echo dan menerima pengembalian echo, untuk keperluan diagnostik.
Pesan Echo reply harus dikirim dalam merespon pesan echo request yang dikirim ke alamat *multicast* IPv6. Alamat asal dari pengembalian harus alamat *unicast* milik *interface* yang menerima pesan permintaan echo *multicast*.
Data yang diterima dalam pesan permintaan echo ICMPv6 harus dikembalikan semua dan tidak diubah-ubah sedikitpun dalam pesan pengembalian echo ICMPv6.
Pesan pengembalian echo harus dilewatkan ke proses yang merupakan asal dari pesan permintaan echo. Hal ini mungkin juga diteruskan ke proses yang tidak menjadi asal pesan permintaan echo.

2.3.4. Pertimbangan Keamanan

2.3.4.1. Autentifikasi dan Enkripsi dari Pesan ICMP

Pertukaran paket protokol ICMP dapat diautentifikasi menggunakan autentifikasi *header* IP. *Node* harus memasukkan *header* autentifikasi pada waktu mengirimkan pesan ICMP jika asosiasi keamanan untuk alamat tujuan tersebut. Asosiasi keamanan mungkin dibuat melalui manual konfigurasi atau melalui operasi dari beberapa protokol manajemen kunci.

Header autentifikasi dalam *header* ICMP yang diterima harus diperiksa dulu kebenarannya dan paket dengan autentifikasi yang tidak sesuai harus diabaikan dan dibuang.

Ada kemungkinan bagi administrator system untuk mengkonfigurasi *node* untuk mengabaikan semua pesan ICMP yang tidak terautentifikasi menggunakan *header* autentifikasi atau muatan encapsualsi keamanan. Switch-switch harusnya secara default mengizinkan pesan yang tidak menggunakan autentifikasi.

2.3.4.2. Penyerangan ICMP

Pesan-pesan ICMP mungkin menjadi sasaran dari beberapa penyerangan. Berikut ini beberapa penyerangan dan pencegahannya:

1. Pesan-pesan ICMP mungkin menjadi sasaran terhadap aksi yang dapat menyebabkan penerima percaya bahwa pesan datang dari sumber yang berbeda dibandingkan dengan asal pesan. Proteksi terhadap pesan ini dapat dilakukan dengan menerapkan mekanisme autentifikasi IPv6.
2. Pesan-pesan ICMP mungkin menjadi sasaran terhadap aksi yang dapat menyebabkan pesan atau pesan balasan pergi ke tujuan yang berbeda dari tujuan yang sebenarnya. Kalkulasi checksum ICMP menyediakan mekanisme proteksi serangan yang mengubah kolom alamat IP tujuan dan asal dalam paket yang membawa pesan. Kolom checksum ICMP yang mencegah adanya serangan diganti oleh autentifikasi atau enkripsi dari pesan ICMP.
3. Pesan-pesan ICMP mungkin menjadi sasaran terhadap perubahan kolom pesan atau payload. Autentifikasi atau enkripsi dari pesan ICMP adalah proteksi dari serangan tersebut.
4. Pesan-pesan ICMP mungkin digunakan untuk mencoba melakukan serangan DoS (Denial of Service) dengan saling

megirimkan kembali paket-paket yang salah. Proteksi dari serangn ini adalah dengan mekanisme pembatasan error rate ICMP.

2.4. Neighbor *Discovery* (ND) untuk IPversi6 (IPv6)[4]

Node (*host* dan *router*) menggunakan ND untuk mencari alamat-alamat *layer link* (misalkan alamat MAC pada ethernet) guna mengetahui *node-node* tetangga yang berada pada *link* yang sama dan secara cepat menghapus alamat-alamat tersebut pada cache jika alamat tersebut sudah tidak berlaku lagi. *Host-host* juga menggunakan ND untuk mencari *router-router* yang menjadi tetangganya yang bersedia melewatkan paket-paketnya untuk kepentingan *host* itu sendiri. Dan juga, *node-node* menggunakan protokol ini untuk mendeteksi perubahan pada alamat *layer link* (*layer* 2). Ketika *router* atau jalur ke *router* gagal/rudak, *host* akan secara aktif mencari alternatif yang aktif.

Alamat *multicast* untuk semua *node* adalah FF02::1, yang merupakan jangkauan alamat *link-link* untuk dapat mencapai semua *node*. Sedangkan alamat *multicast* untuk semua *router* adalah FF02::2, yang merupakan jangkauan alamat *link*-local untuk mencapai semua *router*.

2.4.1. Overview Protocol

Protokol ini memecahkan sejumlah masalah sebelumnya yang berhubungan dengan interaksi antara *node-node* yang terhubung dalam *link* yang sama. Protokol ini mendefinisikan mekanisme untuk memecahkan setiap masalah berikut:

1. *Router Discovery*, Bagaimana *host-host* mencari *router* yang berkoneksi pada *link*
2. Prefix *Discovery*, Bagaimana *host-host* menemukan alamat prefix yang merupakan alamat atau pengidentifikasi *link* tempat *host-host* tersut saling terinterkoneksi (*node-node* menggunakan prefix untuk membedakan apakah *node* yang akan tersbut beradapada *link* yang sama dengan *node* asal atau *node* yang akan dituju tersebut yang hanya dapat dijangkau melalui *router*).

[4] A. Narten, E. Nordmark, W. Simpson, 1998, **Neighbor Discovery for IP Version 6 (IPv6)**, Request for Comments 2461

3. Parameter *Discovery*, Bagaimana *node* mempelajari parameter-parameter pada *link* seperti *link* MTU atau parameter-parameter internet seperti jumlah batasan hop yang akan ditempatkan pada peket yang akan dikirim.
4. *Address* Autoconfiguration, Bagaimana *node-node* secara otomatis mengkonfigurasi alamat IPv6 untuk *interface*nya.
5. *Address* Resolution, Bagaimana *node-node* mencari alamat *link layer* dari *node* yang akan dituju yang masih berada pada *link* yang sama (misalnya *node* tetangga) hanya dengan diberikan alamat IP *node* tujugannya saja.
6. *Next*-Hop determination, Algoritma untuk memerakan alamat IPv6 dari *node* tujuan ke dalam alamat IPv6 *node* tetangga. Trafik untuk *node* tujuan tersebut akan dikirimkan ke *node* tetangga tersbut. *Next*-Hop ini dapat berupa *router* atau *host* tujuan itu sendiri (bergantung pada ke mana trafik itu akan dikirim, jika ke tujuan yang masih dalam satu *link* yang sama maka *next* hop adalah *node* tujuan itu sendiri, jika tujuannya sudah berbeda *link*/prefix maka *next* hop tersbut adalah *router*).
7. Neighbor Unreachability Detection, Bagaimana *node* mempelajari bahwa salah saru tetangga sudah tidak aktif lagi. Untuk *node* tetangga yang digunakan sebagai *router*, *node* tersebut dapat mencoba rute default alternatif. Untuk kedua *router* dan *host*, *Address* resolution dapat dilakukan kembali.
8. Duplicate *Address* Detection, Bagaimana *node* mempelajari bahwa alamat yang ingin digunakan sedang tidak digunakan oleh *node* lain.
9. Redirect, Bagaimana *router* memberitahu *host* tentang *node* pertama mana yang baik sebagai *next* hop untuk mencapai tujuan tertentu.

Neighbor *Discovery* mendefinisikan lima tipe paket ICMP yang berbeda, yaitu sepasang yang terdiri atas *router* solocotation dan *router* advertisement messages, sepasang yang terdiri atas Neighbor Solicitation dan Neighbor Advertisement Messages dan terakhir pesan Redirect. Pesan-pesan tersebut melayani beberapa tujuan berikut:

a. ***Router* Solocitation**: ketika sebuah *interface* pada *host* menjadi diaktifkan, *host-host* boleh mengirimkan keluar *router*

solicitation yang meminta *router* untuk mengirimkan *router* advertisement sesegera mungkin ke *host* tersbut.

b. ***Router* Advertisement** : *router* mengumumkan keberadaan mereka dengan berbagai macam *link* dan parameter internet secara periodik atau dalam merespon pesan *router* solicitation. *Router* advertisement ini dikirim kepada *host-host* yang berada di sekitar *router*. *Router* advertisement berisi prefix-prefix yang digunakan untuk pengenalan/pencarian *link* dan/atau untuk konfigurasi alamat, nilai perkiraan batasan hop dan sebagainya.
c. **Neighbor Solicitation** dikirim oleh *node* untuk mencari alamat *layer link node* tetangga-tetangganya, atau untuk memeriksa apakah tetangga tersbut masih dapat dijangkau dengan alamat *layer link* yang berada dalam memori cache-nya atau tidak Neighbor Solicitation juga digunakan untuk mendeteksi alamat yang ganda.
d. **Neighbor Advertisement** adalah sebuah respon kepada pesan neighbor solicitation. *Node* boleh juga mengirimkan neighbor advertisemnt tanpa didahului oleh neighbor solicitation yang digunakan untuk mengumumkan perubahan alamat *layer link* pada tetangga tersbut.
e. **Redirect** Digunakan oelh *router-router* untuk menginformasikan *host-host* tentang hop pertama yang paling baik untuk sebuah tujuan tertentu.

Dengan kemampuan *multicast* pada *link*, setiap *router* secara periodik mengirimkan secara *multicast* paket *router* advertisement yang mengumumkan keberadaanya. *Host* menerima *router* advertisement dari semua *router*, dan kemudian membuat daftar *router* default. *Router* mengirimkan pesan *router* advertisement ini cukup sering sehingga *host* dapat mempelajari keberadaannya mereka dalam beberapa menut namun juga tidak cukup sering untuk mendeteksi adanya kesalahan pada *router* dengan menggunakan tidak adanya *router* advertisement yang diterima. Algoritma deteksi ketidakterjangkauannya tetangga ini dibuat terpisah dan algoritma ini menyediakan deteksi kesalahan.

Router advertisement berisi daftar dari prefix-prefix yang digunakan untuk penentuan alamat *link* yang digunakan dan/atau konfigurasi alamat pada *host* secara mandiri. *Host* menggunakan prefix-prefix yang diumumkan oleh *router* untuk membuat dan memelihara daftar yang akan digunakan untuk memutuskan apakah paket tujuan ini

masih dalam *link* yang sama atau harus dijangkau dengan *router*. Perlu diingat bahwa tujuan dapat juga benda dalam *link* yang sama meskipun tudak diikutsertakan dalam pengumuman prefix-prefix yang berada pada saru *link* yang sama. Pada kasus ini *router* dapat mengirimkan pesan redirect yang menginformasikan pengirim bahwa tujuan redirect yang menginformasikan pengirim bahwa tujuan tersebut adalah masih metupakan tetangga.

Router Advertisement mengizinkan *router-router* untk menginformasikan *host-host* di bawahnya bagaimana melaukan konfigurasi alamat otomatis. Sebagai contoh *router* dapat menspesifikasikan apakah *host-host* tersbut sebaiknya menggunakan konfigurasi alamat secara stateful (DHCPv6) dan/atau secara autonomous/sendiri (stateless). Pesan *router* advertisement juga berisi parameter-parameter internet seperti batas hop yang *host-host* sebaiknya gunakan dalam paket yang akan ditransmisikan dan secara opsional dapat berisi pula parameter seperti *link* MTU. Fasilitas ini memusatkan administrasi dari parameter-parameter yang penting yang dapat diset di *router-router* dan secara otomatis dipropagasikan ke semua *host-host* yang terkoneksi di bawah *router*.

Node melaksanakan resolusi/pencarian alamat dengan mengirimkan balik alamat *layer link* nya. Pesan neighbor solicitation adalah pesan yang di*multicast*kan (dikirim ke group/kelompok *node* tertentu) ke alamat *multicast* tertentu tempat *node* yang merupakan target tersebut menjadi salah satu anggotanya. *Node* yang nerupakan target mengirimkan alamat *layer link* nya ke *node* yang meminta dalam bentuk pesan neighbor advertisement yang *unicast*. Satu pasang tanya-jawab dari paket ini sudah cukup untuk kedua *node* yang melakukan tanya-jawab untuk saling mengetahui alamat *link layer*. Hal ini karena *node* yang merupakan inisiator juga mengirimkan alamat *layer link* dalam pesan neighbor solicitaion. Pesan neighbor solicitation dapat juga digunakan untuk mempelajari jika lebih dari satu *node* dialokasikan alamat *unicast* yang sama.

Neighbor Unreachability Detection mendeteksi adanya masalah pada *node* tetangga atau masalah pada jalur pelewatan paket ke tetangga. Metode ini memerlukan konfirmasi positif yang menyatakan bahwa paket yang dikirimkan ke tetangga telah benar-benar sampai dan telah diproses dengan baik oleh *layer* IP. Neighbor Unreachability Detection menggunakan konfirmasi dari dua sumber. Ketika memungkinkan, protokol pada *layer* yang lebih tinggi menyediakan

konfirmasi positif yang menyatakan bahwa koneksi telah mencapai kemajuan yang berarti data yang dikirimkan sebelumnya telah sampai dengan benar. Ketika knfirmasi positif tidak menunjukkan tanda-tanda kedatangannya , *node* mengirimkan pesan *unicast* neighbor solicitation yang meminta neighbor advertisement sebagai konfirmasi keterjangkauan *node* yang dituju pada *node* yang menhadi hop berikutnya. Untuk mengurangi trafik jaringan yang tidak perlu, pesan untuk penyelidikan hanya dikirim ke tetangga tempat *node* secara aktif mengirimkan paket-paket.

Sebagai tambahan terhadap masalah umum di atas, Neighbor *Discovery* juga menangani situasi berikut ini:

Perubahan alamat *link-layer node* yang tahu alamat *link-layer*nya berubah dapat me*multicast*kan pesan dengan mengirimkan beberapa paket Neighbor Advertisement ke semua *node* untuk secara cepat mengupdate alamat *link-layer* dalam memori cache yang sidah tidak berlaku lagi.

Inbound load balancaing – *node* yang mempunyai *interface* ganda ataupun lebih dari satu mungkin ingin melakukan load balancing penerimaan paket-paket yang dapat melalui *interface* jaringan yang lebih dari satu pada *link* yang sama. Seperti *node* yang mempunyai alamat *link-layer* yang lebih dari satu yang diberikan pada *interface* yang sama. Sebagai contoh dari driver jaringan yang tunggal (software) dapat merepresentasikan card *interface* jaringan yang lebih dari satu sebagai *interface* logikal tunggal yang mempunyai alamat *layer link* yang lebih dari satu. Load balancing ditangani oleh *router* dengan menghilangkan alamat *link-layer* asal dari paket *Router* Advertisement, dengan demikian memaksa tetangga untuk menggunkan pesan Neighbor Solicitation untuk mempelajari alamat *layer link router*. Pesan neighbor advertisement yang kembali dapat berisi alamat *layer link* yang berbeda.

Alamat *anycast* – alamat *anycast* mengidentifikasi satu dari sekumpulan nide yang menyediakan layanan yang sama, dan *node-node* pada *link* yang sama dapat dikonfigurasi untuk mengenal alamat *anycast* dengan menerima Neighbor Advertisement lebih dari satu untuk target yang sama. Semua advertisement yang diterima untuk alamat *anycast* ditandai sebagai advertisement yang tidak dapat ditimpa. Dengan mengguakan peraturan-peraturan yang spesifik maka akan dipelajari advertisement yang mana yang akan digunakan.

2.5. Transmisi Paket IPv6 melalui Ethernet[5]

2.5.1. MTU (maximum Transmission Unit)

Ukuran MTU secara default untuk paket IPv6 pada ethernet adalah 1500 oktet. Ukuran ini mungkin dikurangi oleh *router* advertisement yang berisi pilihan MTU yang menspesifikasikan MTU yang lebih kecil, atau oleh konfigurasi manual setiap *node* nya. Jika *router* advertisement diterima pada *interface* yang mempunyai pilihan MTU yang menspesifikasikan MTU lebih besar dari 1500, atau lebih besar dari nilai yang dikonfigurasi secara manual, pilihan MTU tersebut akan dicatat pada manajemen sistem dan harus diabaikan

2.5.2. Format Frame

Paket IPv6 ditransmisikan dalam frame standar ethernet. *Header* ethernet berisi alamat ethernet tujuan dan asal serta berisi kode tipe ethernet, dimana harus berisi alamat hexadesimal dengan nilai 86DD. Kolom data berisi *header* IPv6 diikuti secara langsung oleh payload dan oktet padding untuk memenuhi ukuran frame minimum pada *link* ehternet.

2.5.3. Stateless Autoconfiguration

Node yang pertama kali tersambung ke jaringan akan secara otomatis mengkonfigurasi alamat IPv6 *site*-local dan global tanpa memerlukan manual konfigurasi atau bantuan dari server seperti server DHCP (Dynamic *Host* Configuration Protocol). Dengan IPv6, *router* akan mengirimkan pesan *router* advertisement yang berisi prefix global dan *site*-local.

Pesan *router* advertisement ini akan dikirim secara periodik atau dapat dikirm sewaktu-waktu untuk merespon pesan *router* solicitation yang dikirim oleh *host* pada waktu startup sistem. Alamat global IPv6 dibentuk oleh prefix yang diberikan oleh pesan *router* advertisement dan pengidentifikasi unterface (EUI-64). Alamat prefix IPv6 digunakan untuk stateless autoconfiguration dari *interface* erhernet harus mempunyai panjang 64 bit.

---

[5] M. Crawford, 1998, **Transmission of IPv6 Packets over Ethernet Networks,** Request for Comments 2464

Pengidentifikasi *interface* untuk *interface* ethernet berbasis pada pengidentifikasi EUI-64 yang diperoleh dari alamat *interface* IEEE 802 dengan pangjang 48 bit yang sudah ada pada tiap *interface* etheternet. Deskripsi EUI-64 sebagai berikut:

Tiga oktet pertama dari lamat ehternet menjadi perusahaan ID dari EUI-64 (tiga oktet pertama). Oktet keemapat dan kelima dari EUI diset tetap dengan nilai FFFE dalam hexadesimal. Tiga oktet terakhir dari alamat ethernet menjadi tiga alamat terakhir dari EUI-64.

Pengidentifikasi *interface* kemudia dibentuk dari EUI-64 dengan mengkomplemenkan bit "Universal/Local" (U/L), yang mana merupakan bit order terendah dari oktet pertama EUI-64. Mengkomplemenkan hal ini akan berubah dari nilai 0 menjadi 1, karena alamat yang sudah terdapat pada *interface* diharapkan dari alamat yang dialokasikan secara universal dan mempunyai nilai yang unik. Alamat IEEE 802 yang diaokasikan secara universal atau EUI-64 ditandai dengan 0 pada posisi U/L bitnya, sedangkan pengidentifikasi *interface* IPV6 secra global ditandai dengan nilai 1 pada posisi yang sama.

Sebagai contoh, pengidentifikasi *interface* untuk *interface* ethernet yang mempunyai alamat yang sudah berada didalamnya, dalam hexadesimal:
34-56-78-9A-BC-DE akan menjadi 36-56-78-FF-FE-9A-BC-DE.

2.5.4. Alamat *link*-local

Alamat *link* local IPv6 untuk *interface* ethernet dibentuk oleh dengan menambahkan pengidentifikasi *interface* dengan prefix FE80::/64.

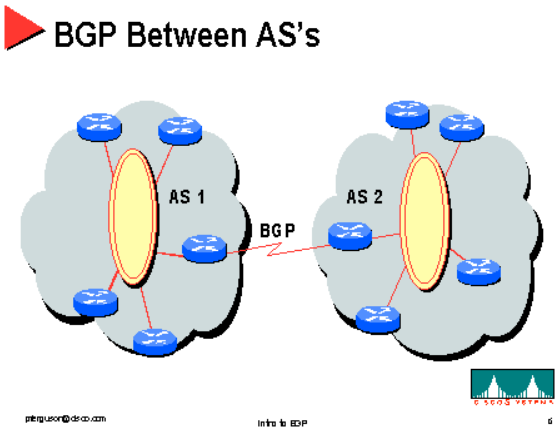
2.6. Protokol *Routing* pada IPv6

Protokol *routing* yang digunakan pad a IPv6 adalah BGP4+ untuk external *routing* dan OSPFv6, RIPng untuk internal *routing*.

2.6.1. BGP4+

Border Gateway Protokol adalah *routing* protokol yang memakai system autonomous. Fungsi utama dari BGP adalah untuk saling tukar-menukar informasi konektivitas jaringan antar BGP sistem. Informasi konektifitas ini antara lain adalah daftar dari Autonomous System (ASs). Informasi ini digunakan untuk membuat daftar *routing* sehingga terjadi suatu koneksi.

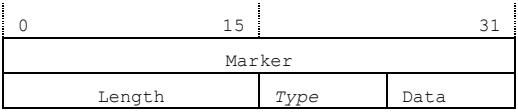
BGP4 mampu melakukan suatu advertaisement dan IP-prefix serta menghilangkan keterbatasan tentang network class. BGP memakai pola *Hop-by-Hop* yang artinya hanya meggunakan jalur yang berikutnya yang terdaftar dalam Autonomous System.
BGP menggunakan TCP sebagai media transport. BGP menggunakan port 179 untuk koneksi BGP. BGP mendukung CIDR.



Gambar II.22 Model BGP

BGP mampu mempelajari jalur internet malalui internal atau eksternal BGP dan dapat memilih jalur terbaik dan memasukkannya dalam ip forwarding.

BGP dapat digunakan pada dual maupun multi-homed, dengan syarat memiliki nilai AS. BGP tidak dapat digunakan pada single-homed.



Gambar II.23 Format BGP Header

Type dari BGP:

OPEN, tipe pesan yang diterima sewaktu koneksi antar BGP tersambungkan.

UPDATE, tipe pesan yang dikirimkan untuk mengirimkan informasi *routing* antar BGP.

KEEPALIVE, tipe pesan yang dikirimkan untuk mengetahui apakah pasangan BGP masih hidup

NOTIFICATION, tipe pesan yang dikirimkan apabila terjadi error.

Atribut yang dimiliki oleh BGP:

AS_*path*, adalah jalur yang dilalui dan dicatat dalam data BGP route, dan dapat mendeteksi loop.

*Next*_Hop, adalah jalur berikutnya yang akan dilalui dalam *routing* BGP, biasanya adalah local network dalam eBGP. Selain itu bisa didapat dari iBGP.

Local Preference, penanda untuk AS BGP local

Multi-Exit Discriminator (MED), bersifat non-transitif digunakan apabila memiliki eBGP yang lebih dari 1.

Community, adalah sekumpulan BGP yang berada dalam satu AS.

Perbandingan BGP-4[6] antara yang digunakan untuk IPv4 dan IPv6 adalah kemampuan dari BGP yang dapat mengenali *scope* dari IPv6, yaitu *global, site-local, link-local*. Apabila IPv6 masih menggunakan IPv4 sebagai transport maka alamat *peer* pada BGP yang lainnya harus diikutkan pada konfigurasi.

2.6.2. RIPng

Routing Information Protocol *Next* Generation adalah protokol *routing* yang berdasarkan protokol *routing* RIP di IPv4 yang sudah mendukung IPv6.

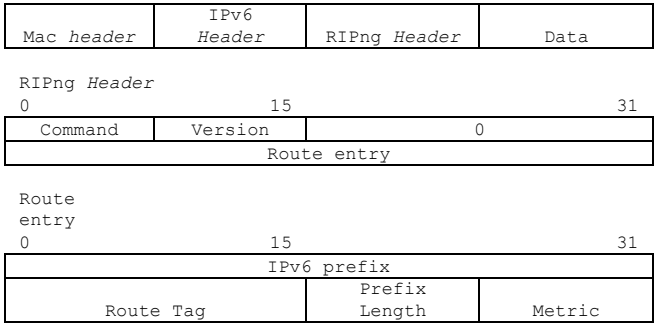
RIPng ini digunakan untuk internal *routing* protokol dan menggunakan protokol UDP sebagai transport. RIPng ini menggunakan port 521 sebagai komunikasi antar RIPng.

Metode yang dipakai RIPng adalah distance vector (vektor jarak), yaitu:

---

[6] P. Marques, F. Dupont, 1999, **Use of BGP-4 Multiprotocol *Extension*s for IPv6 Inter-Domain *Routing*,** Request for Comments 2545.

1. Jarak local network dihitung 0
2. Kemudian mencari neighbour sekitar dan dihitung jaraknya dan cost.
3. Dibandingkan jarak dan cost antar neighbour.
4. Dilakukan perhitungan secara kontinue.
5. Menggunakan algoritma Ballman-Ford.



Gambar II.24Format RIPng Header

Command pada RIPng *Header* berisi:

1 Request, meminta daftar tabel *routing* pada RIPng yang lain

2 Response, membalas request dari RIPng yang lain dan memberikan daftar *routing*.

Protokol RIPng ini memiliki beberapa kelemahan

1. Hanya bisa sampai 15 HOP
2. Lambat dalam memproses *routing*, dikarena melakukan pengecekan terus menerus
3. Bersifat Classful

Perbedaan yang terjadi antara RIP pada IPv4 (RIPv2) dan IPv6 (RIPng)[7] adalah port UDP dimana pada IPv4 menggunakan port 520

[7] G. Malkin, R. Minnear, 1997, **RIPng for IPv6**, Request for Comments 2080.

sedangkan IPv6 menggunakan port 521 sebagai media transpor. RIPng hanya memiliki 2 perintah yaitu *response* dan *request*, berbeda dengan RIPv2 yang memiliki banyak perintah dan banyak yang tidak terpakai dan ada yang dibuang pada RIPng seperti authentifikasi. Perubahan yang terjadi dari RIPv2 ke RIPng antara lain, ukuran *routing* yang tidak lagi dibatasi, *subnet* IPv4 digantikan dengan *prefix* IPv6, next-hop dihilangkan tetapi kegunaannya tidak dihilangkan, authentifikasi dihilangkan, namun kemampuan yang hanya sampai 15 hop masih sama.

2.6.3. OSPFv3

Open Shortest *Path* First adalah *routing* protokol yang digunakan pada IPv6. OSPF ini berdasarkan atas *Link*-state dan bukan berdasarkan atas jarak. Setiap *node* dari OSPF mengumpulkan data state dan mengumpulkan pada *Link* State Packet.

LSP dibroadcast pada setiap *node* untuk mencapai keseluruhan network. Setelah seluruh network memiliki "map" hasil dari informasi LSP dan dijadikan dasar *link*-state dari OSPF. Kemudian setiap OSPF akan melakukan pencarian dengan metode SPF (Shortest *Path* First) untuk menemukan jarak yang lebih efisien.

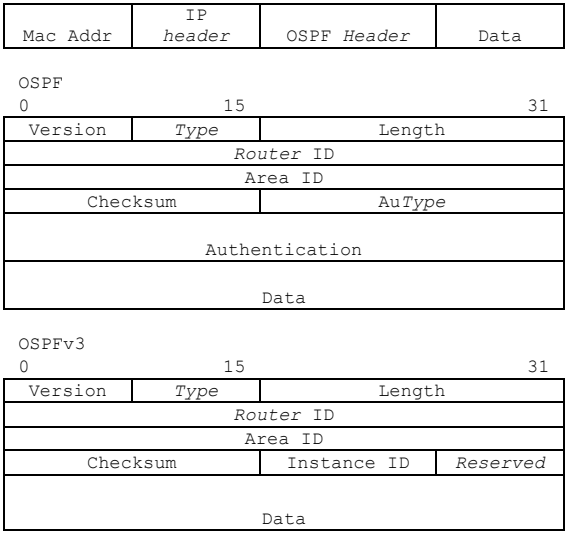
Routing table yang dihasilkan berdasarkan atas informasi LSP yang didapat sehingga OSPF memberikan informasi LSP secara flood, karena OSPF sudah memiliki kemampuan untuk memilih informasi LSP yang sama maka flood ini tidak mengakibat exhousted.

OSPF ini menggunakan protokol TCP bukan UDP, mendukung VLSM (Variable Length Subnet Mask).

OSPF menggunakan algoritma Shortest *Path* First (SPF) oleh Dijkstra, yaitu:

1. Diasumsikan sudah ada data table sebelumnya. Data yang diperlukan antara lain *PATH* (ID, *path* cost, arah forwarding) TENTATIVE (ID, *path* cost, arah forwarding), Forwarding database.
2. Taruh local sebagai root dari tree dengna ID,0,0 pada *PATH*.
3. Temukan *link* N dan taruh di *PATH*. Hitung jarak Root-N dan N-M, apabila M belum terdapat di *PATH* atau TENTATIVE, apabila nilainya lebih baik taruh di TENTATIVE.

4. Apabila TENTATIVE bernilai kosong , batalkan. Lainnya, masukkan nilai TENTATIVE ke *PATH*.



Gambar II.25Format OSPF Header

Keterangan OSPF:

Version, 8 bit, diisi dengan dengan versi dari OSPF

Type, 8 bit, diisi dengan *Type* code dari OSPF yaitu:

1. Hello, untuk mengetahui adanya pasangan OSPF
2. Database Description, mengirimkan deskripsi dari OSPF
3. *Link* State Request, meminta data dari pasangan OSPF
4. *Link* State Update, mengupdate data table pada OSPF
5. *Link* State Aknowledgment, mengirimkan pesan error

Length, 16 bit, panjang *header* dan data dari OSPF

Router ID, 32 bit, *Router* ID dari *source* paket

Area ID, 32 bit, Area dari paket ini.

Checksum, 16 bit

Au*Type*, 16 bit, model autentifikasi dari OSPF

Authentication, 64 bit, misal tanpa autenticasi, simple password, cryptographic password

Data

Keterangan untuk OSPFv3:

Version, 8 bit, diset 3

Checksum, 16 bit, CRC

Instance ID, 8 bit

Reserves, 8 bit diset 0

Perbandingan antara *link*-state daripada distance vector:

a. Konversi lebih cepat dari pada distance vector
b. Mudah dalam bentuk Topologi jaringan
c. Mudah dalam hal *Routing*
d. Bisa memiliki *routing* table yang komplek

Perbedaan yang terjadi antara OSPF IPv4 dengan OSPF IPv6[8] adalah:

a. Komunikasi menggunakan *link-state* tidak menggunakan subnet.
b. Menghilangkan alamat semantic.
c. Menggunakan scope IPv6 yaitu: *link-local scope, area-scope, AS scope.*

[8] R. Coltun, D. Ferguson, J. Moy, 1999, **OSPF for IPv6,** Request for Comments 2740.

d. Mendukung multi OSPF pada *link* yang sama.
e. Menggunakan alamat *link-local.*
f. Menghilangkan authentifikasi.
g. Perubahan format paket.

Ini Kosong!

BAB III. IMPLEMENTASI IPV6 PADA OS LINUX

3.1 Syarat-syarat yang diperlukan untuk implementasi IPv6

3.1.1 Syarat personal

1. Berpengalaman dengan tool Unix, seperti *grep, awk, find*, dan paham akan kegunaan dari *command line* dari Unix.
2. Berpengalaman dalam bidang networking, dimana paham tentang macam-macam *layer*, protokol, *Address*, kabel, konektor, dan lain sebagainya.
3. Paham tentang konfigurasi IPv4
4. Paham tentang Domain Name System (DNS)
5. Paham tentang konsep strategi jaringan

3.1.2 Hardware

Pemilihan hardware sangat penting atas ke kompatibelan dengan networking.

3.2 Dasar

3.2.1 Kernel yang mendukung IPv6

Beberapa distribusi *Linux* sudah mendukung akan IPv6, seperti Debian, RedHat, PLD, Suse, Mandrake dan sebagainya. Kernel yang mendukung IPv6 bisa dalam bentuk modul atau sudah dimasukkan dalam image kernel. Apabila masih dalam bentuk modul.

Untuk mengecek apakah *linux* yang digunakan sudah mendukung IPv6 dapat dilakukan dengan melihat apakah pada direktori /proc sudah terdapat file: `/proc/net/if_inet6`
atau dengan cara:

```
# test -f /proc/net/if_inet6 && echo "IPv6 sudah siap"
```

Kode III-1. Test Kernel IPv6

Apabila test diatas gagal kemungkinan IPv6 berbentuk modul dan perlu dipanggil terlebih dahulu.

Untuk memanggil modul IPv6 dapat dengan cara: "`modprobe ipv6`". Apabila perintah diatas sukses, modul IPv6 akan tampak di

daftar modul kernel, untuk mengecek apakah modul ipv6 sudah terpanggil dapat dengan cara:

```
# modprobe ipv6
# lsmod |grep -w 'ipv6' && echo "IPv6 modul sukses"
```

Kode III-2. Memanggil Kernel IPv6

Apabila tampil pernyataan "sukses" berarti ipv6 sudah siap.

Catatan: modul ipv6 tidak dapat dilepas.

Untuk dapat meload kernel modul ipv6 secara otomatis dapat dilakukan dengan menambahkan konfigurasi di file /etc/modules.conf, yaitu:

```
/etc/modules.conf

alias net-pf-10 ipv6  # automatically load IPv6 module on demand
```

Kode III-3. Memanggil Kernel IPv6 Secara Otomatis

Apabila cara-cara diatas masih gagal, berarti kernel masih belum mendukung IPv6, untuk itu diperlukan kompilasi ulang kernel. Cara yang perlu dilakukan adalah:

1. Download *source* kernel dari http://www.kernel.org, versi stable pada saat dokumen ini ditulis adalah 2.2.20 atau 2.4.18
2. Extrak file kernel yang didownload di direktori /usr/src dengan perintah:
   ```
   # tar -jxvf linux-2.4.18.tar.bz2 -C /usr/src
   ```
3. Jalankan "make menuconfig" untuk memulai konfigurasi atas kernel.
4. Konfigurasi yang perlu diperhatikan untuk dapat mendukung IPv6 antara lain:
   ```
   CONFIG_EXPERIMENTAL=y
   CONFIG_NET=y
   CONFIG_PACKET=y
   CONFIG_PACKET_MMAP=y
   CONFIG_NETLINK_DEV=y
   CONFIG_NETFILTER=y
   CONFIG_NETFILTER_DEBUG=y
   CONFIG_FILTER=y
   CONFIG_UNIX=y
   CONFIG_INET=y
   CONFIG_IP_MULTICAST=y
   CONFIG_IP_ADVANCED_ROUTER=y
   CONFIG_IP_MULTIPLE_TABLES=y
   CONFIG_IP_ROUTE_FWMARK=y
   ```

```
CONFIG_IP_ROUTE_NAT=y
CONFIG_IP_ROUTE_MULTIPATH=y
CONFIG_IP_ROUTE_TOS=y
CONFIG_IP_ROUTE_VERBOSE=y
CONFIG_IP_ROUTE_LARGE_TABLES=y
CONFIG_IP_PNP=y
CONFIG_NET_IPIP=m
CONFIG_NET_IPGRE=m
CONFIG_NET_IPGRE_BROADCAST=y
CONFIG_IP_MROUTE=y
CONFIG_IP_PIMSM_V1=y
CONFIG_IP_PIMSM_V2=y
CONFIG_ARPD=y
CONFIG_INET_ECN=y
CONFIG_SYN_COOKIES=y
CONFIG_IPV6=y
CONFIG_IP6_NF_IPTABLES=m
CONFIG_IP6_NF_MATCH_LIMIT=m
CONFIG_IP6_NF_MATCH_MAC=m
CONFIG_IP6_NF_MATCH_MULTIPORT=m
CONFIG_IP6_NF_MATCH_OWNER=m
CONFIG_IP6_NF_MATCH_MARK=m
CONFIG_IP6_NF_FILTER=m
CONFIG_IP6_NF_TARGET_LOG=m
CONFIG_IP6_NF_MANGLE=m
CONFIG_IP6_NF_TARGET_MARK=m
```

5. Konfigurasi kernel yang lainnya disesuaikan dengan kondisi hardware, dan kemampuan dari *linux* box tersebut.
6. Untuk memulai kompilasi jalankan perintah:
   ```
   # make dep
   # make bzImage atau # make install
   # make modules modules_install
   ```
7. Restart mesin *Linux* tersebut.
8. Untuk mengecek apakah kernel sudah mendukung IPv6 dapat dilakukan dengan:
   ```
   # dmesg | grep IPv6
   ```
   Apabila terdapat output dibawah ini, berarti kernel sudah mendukung IPv6.
   ```
   IPv6 v0.8 for NET4.0
   IPv6 over IPv4 tunneling driver
   ```

Kode III-4. Kompilasi Kernel IPv6

3.2.2 IPv6 Network Configuration tools

IPv6 Network Configuration tools adalah program/aplikasi yang dapat digunakan untuk mengkonfigurasi jaringan yang mendukung IPv6. Karena tanpa aplikasi ini tidak akan dapat dibangun jaringan yang mendukung IPv6, walaupun kernel sudah mendukung IPv6.

3.2.2.1 Paket net-tools

Pada paket net-tools terdapat aplikasi ifconfig dan route, dimana aplikasi ini berguna untuk menyetting IPv6. Jalankan ifconfig -? Atau route -?, apabila menemukan output seperti IPv6 atau inet6 berarti tools ini sudah mendukung IPv6. Atau jalankan:

```
# /sbin/ifconfig -? 2>& 1|grep -qw 'inet6' && echo "'ifconfig'
mendukung IPv6"


# /sbin/route -? 2>& 1|grep -qw 'inet6' && echo "'route' mendukung
IPv6"
```

Kode III-5. Net-Tools mendukung IPv6

3.2.2.2 Paket iproute

Alexey N. Kuznetsov (maintener untuk kode *Linux* Networking) menbuat satu paket dimana paket tersebut dapat mengkonfigurasi janringan melalui peralatan net*link*. Program yang dipakai dari paket iproute ini yaitu: "ip".

```
# /sbin/ip 2>&1 |grep -qw 'inet6' && echo "'ip' mendukung IPv6"
```

Kode III-6. Iproute mendukung IPv6

Apabila perintah ip ini tidak dapat ditemukan, dianjurkan untuk menginstall paket iproute terlebih dahulu. Paket iproute ini dapat didownload dari ftp://ftp.inr.ac.ru/ip-*routing*/, atau dari distribusi *linux* yang digunakan. Apabila menggunakan distribusi Debian jalankan perintah "apt-get install iproute".

3.2.3 Program tes IPv6

Setelah memastikan system yang digunakan sudah mendukung IPv6, kemudian perlu dilakukan tes apakah sudah dapat dilakukan komunikasi IPv6.

3.2.3.1 IPv6 ping

Progam ini biasanya tergabung dalam paket iputils. Program ini didesign untuk melakukan tes sederhana dengan cara mengirimkan ICMPv6 paket echo-request dan menunggu paket ICMPv6 echo-reply. Cara menggunakannya sebagai berikut:

```
# ping6 <host_dengan_alamat_ipv6>
# ping6 <alamat_ipv6>
# ping6 [-I <device>] <link-local-alamat_ipv6>

Contoh:
```

```
# ping6 -c 1 ::1
PING ::1(::1) from ::1 : 56 data bytes
64 bytes from ::1: icmp_seq=0 hops=64 time=292 usec
--- ::1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.292/0.292/0.292/0.000 ms
```

Kode III-7. Ping6

3.2.3.2 IPv6 traceroute6

Program ini biasanya tergabung dalam paket iputils. Program ini mirip dengan traceroute dari IPv4. Lain dengan versi IPv4, traceroute IPv6 tidak menggunakan paket ICMP echo-request melainkan menggunakan UDP. Berikut contoh traceroute6:

```
# traceroute6 www.6bone.net
traceroute     to     6bone.net     (3ffe:b00:c18:1::10)     from
3ffe:b80:2:4d50::2, 30 hops max, 16 byte packets
 1                              3ffe:b00:c18:1:2a0:c9ff:fefc:1feb
(3ffe:b00:c18:1:2a0:c9ff:fefc:1feb)     1430.33  ms     2002.22  ms
2469.15 ms
 2   www.6bone.net  (3ffe:b00:c18:1::10)   1989.9  ms   2773.09  ms
1793.13 ms
```

Kode III-8. Traceroute6

3.2.3.3 IPv6 trace*path*6

Program ini biasanya tergabung dalam paket iputils. Program ini mirip dengan traceroute6 dan melacak jalur yang dituju, dan MTU pada jalur tersebut. Berikut ini contoh trace*path*6:

```
# trace path6 www.6bone.net
 1?: [LOCALHOST]                      pmtu 1480
 1:  3ffe:b00:c18:1:2a0:c9ff:fefc:1feb        830.040ms
 2:  www.6bone.net                              asymm  3 922.719ms
reached
     Resume: pmtu 1480 hops 2 back 3
```

Kode III-9. Tracepath6

3.2.3.4 IPv6 tcpdump

Linux memiliki utility untuk menangkap paket yang dilewatlan. Berikut ini contoh penggunakan tcpdump. Aplikasi tcpdump yang sudah mendukung IPv6 mulai versi 3.6.

Tcpdump menggunakan beberapa opsi untuk memfilter paket sehingga meminimalkan noise:

a. icmp6: memfilter trafik ICMPv6
b. ip6: memfilter trafik IPv6
c. proto ipv6: memfilter trafik *tunnel* IPv6-in-IPv4
d. not port ssh: untuk melihat paket yang berjalan di sesi SSH

Begitu juga opsi lainnya yang berguna untuk memberikan informasi tentang paket:

a. "-s 512": meningkatkan panjang paket sepanjang 512 byte
b. "-vv": lebih verbose
c. "-n": menampilkan dalam bentuk numeric IP

Contoh penggunaan tcpdump

```
# tcpdump -t -n -i eth0 -s 512 -vv ip6 or proto ipv6
tcpdump: listening on eth0
202.154.63.9 > 206.123.31.114: fe80::909:909 > ff02::5: OSPFv3-
hello 36: rtrid 202.154.63.9 backbone V6/E/R ifid 0.0.0.60 pri 1
int 10 dead 40 nbrs [hlim 1] (len 36) (DF) (ttl 64, id 0, len 96)
167.205.23.16 > 202.154.63.9: fe80::202:44ff:fe06:c94 > ff02::5:
OSPFv3-hello 36: rtrid 167.205.23.16 backbone V6/E/R ifid 0.0.0.16
pri 255 int 10 dead 40 nbrs [hlim 1] (len 36) (ttl 16, id 24957,
len 96)

2 packets received by filter
0 packets dropped by kernel
```

Kode III-10. TCPDump

3.3 Konfigurasi

Pada *node*, terdapat beberapa peralatan jaringan, dan dapat dibedakan menjadi 2 kelas

1. Fisik, seperti eth0, tr0
2. Virtual, seperti ppp0, tun0, tap0, sit0, isdn0, ippp0

Interface yang termasuk kelas fisik yaitu Ehternet atau Token-Ring dan tidak diperlukan perlakuan khusus.

Interface yang termasuk kelas virtual diperlukan perlakuan khusus, misal

a. *Interface tunnel* IPv6-in-IPv4, biasanya dilambangkan dengan sitX. Nama sit merupakan singkatan dari Simple Internet Transition. Perangkat ini memiliki kemampuan untuk mengencapsulasi paket IPv6 kedalam paket IPv4 dan di*tunnel*kan ke titik terakhir. Sit0 merupakan device khusus dan tidak dapat digunakan.

b. *Interface* PPP, mendukung IPv6 dari PPP daemon

c. *Interface* ISDN HDLC, sudah mendukung IPv6 dari kernel.

d. *Interface* ISDN PPP (ippp), masih belum mendukung IPv6.

e. SLIP + PLIP, *interface* ini masih belum mendukung IPv6.

f. Ether-tap, sudah mendukung IPv6 dan merupakan stateless configuration.

g. ATM, sudah mendukung IPv6 dengan menggunakan kernel dari USAGI.

3.3.1 *Interface*

Untuk mengkonfigurasi *interface* dapat dilakukan dengan metode:

3.3.1.1 ip

Untuk up/down *interface*:

```
# ip link set dev eth0 up
# ip link set dev eth0 down
```

Kode III-11. Perintah ip untuk up/down interface

Untuk melihat alamat IPv6 *interface*

```
# /sbin/ip -6 addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen
100
    inet6 2001:200:830:7000::1/56 scope global
    inet6 fe80::290:27ff:fe0c:ed49/10 scope link
    inet6 2001:470:1f00:371::1/64 scope global
    inet6 3ffe:b80:bb4:1::1/64 scope global
```

Kode III-12. Perintah ip untuk melihat interface

Untuk menambah alamat IPv6 pada *interface*

```
# /sbin/ip -6 addr add 3ffe:ffff:0:f101::1/64 dev eth0
```

Kode III-13. Perintah ip untuk menambah alamat IPv6

Untuk menghilangkan alamat IPv6 dari *interface*

```
# /sbin/ip -6 addr del 3ffe:ffff:0:f101::1/64 dev eth0
```

Kode III-14. Perintah ip untuk menghilangkan alamat IPv6

3.3.1.2 ifconfig

Untuk up/down *interface*:

```
# /sbin/ifconfig eth0 up
# /sbin/ifconfig eth0 down
```

Kode III-15. Perintah ifconfig untuk up/down interface

Untuk melihat alamat IPv6 pada *interface*:

```
# /sbin/ifconfig eth0 |grep "inet6 addr:"
       inet6 addr: 2001:200:830:7000::1/56 Scope:Global
       inet6 addr: fe80::290:27ff:fe0c:ed49/10 Scope:Link
       inet6 addr: 2001:470:1f00:371::1/64 Scope:Global
       inet6 addr: 3ffe:b80:bb4:1::1/64 Scope:Global
```

Kode III-16. Perintah ifconfig untuk melihat alamat IPv6

Untuk menambah alamat IPv6 pada *interface*:

```
# /sbin/ifconfig eth0 inet6 add 3ffe:ffff:0:f101::1/64
```

Kode III-17. Perintah ifconfig untuk menambah alamat IPv6

Untuk menghilangkan alamat IPv6 dari *interface*:

```
# /sbin/ifconfig eth0 inet6 del 3ffe:ffff:0:f101::1/64
```

Kode III-18. Perintah ifconfig untuk menghilangkan alamat IPv6

3.3.1.3 Debian

Apabila menggunakan distribusi debian, tambahkan konfigurasi pada file /etc/network/*interface*:

```
iface eth0 inet6 static
      Address 3ffe:1234:0005:0006::78
      netmask 64
```

Kode III-19. Konfigurasi pada Debian untuk setting interface

3.3.2 *Routing*

Untuk mengkonfigurasi *routing* dapat menggunakan metode:

3.3.2.1 ip

Melihat table *routing*:

```
# /sbin/ip -6 route show dev eth0
3ffe:ffff:0:f101::/64 proto kernel metric 256 mtu 1500 advmss 1440
fe80::/10             proto kernel metric 256 mtu 1500 advmss 1440
ff00::/8              proto kernel metric 256 mtu 1500 advmss 1440
default               proto kernel metric 256 mtu 1500 advmss 1440
```

Kode III-20 Perintah ip untuk melihat tabel routing

Menambah *routing* melalui gateway

```
# /sbin/ip -6 route add 2000::/3 via 3ffe:ffff:0:f101::1
```

Kode III-21. Perintah ip untuk menambah routing melalui gateway

Menghilangkan *routing* yang melalui gateway

```
# /sbin/ip -6 route del 2000::/3 via 3ffe:ffff:0:f101::1
```

Kode III-22. Perintah ip untuk menghilangkan routing melalui gateway

Menambah *routing* melalui *interface*

```
# /sbin/ip -6 route add 2000::/3 dev eth0 metric 1
```

Kode III-23. Perintah ip untuk menambah routing melalui interface

Menghilangkan *routing* melalui *interface*:

```
# /sbin/ip -6 route del 2000::/3 dev eth0
```

Kode III-24. Perintah ip untuk menghilangkan routing melalui interface

3.3.2.2 route

Melihat table *routing*:

```
# /sbin/route -A inet6 |grep -w "eth0"
3ffe:ffff:0:f101 ::/64 :: UA  256 0 0 eth0 <- Interface route for
global Address
fe80::/10        ::       UA  256 0 0 eth0 <- Interface route for
link-local Address
ff00::/8         ::       UA  256 0 0 eth0 <- Interface route for
all multicast Addresses
::/0              ::        UDA 256 0 0 eth0 <- Automatic default
route
```

Kode III-25. Perintah route untuk melihat tabel routing

Menambahkan *routing* IPv6 melalui gateway:

```
# /sbin/route -A inet6 add 2000::/3 gw 3ffe:ffff:0:f101::1
```

Kode III-26. Perintah route untuk menambah routing melalui gateway

Menghilangkan *routing* IPv6 melalui gateway:

```
# /sbin/route -A inet6 del 2000::/3 gw 3ffe:ffff:0:f101::1
```

Kode III-27. Perintah route untuk menghilangkan routing melalui gateway

Menambahkan *routing* IPv6 melalui *interface*:

```
# /sbin/route -A inet6 add 2000::/3 dev eth0
```

Kode III-28. Perintah route untuk menambahkan routing melalui interface

Menghilangkan *routing* IPv6 melalui *interface*:

```
# /sbin/route -A inet6 del 2000::/3 dev eth0
```

Kode III-29. Perintah route untuk menghilangkan routing melalui interface

3.3.3 *Tunnel* IPv6-in-IPv4

Untuk membuat koneksi IPv6 ke dunia dapat dilakukan dengan menggunakan *tunnel* IPv6-in-IPv4 dan terhubung ke IPv6 provider. Untuk menggunakan *tunnel* IPv6-in-IPv4 ini diperlukan beberapa alamat IP. Diperlukan informasi akan IPv4 yang akan dibuat *tunnel* dari kedua sisi yang akan dihubungkan. Selain itu *Tunnel* IPv6-in-IPv4 ini dapat juga dilakukan apabila dalam network anda terdapat *router* yang masih belum support IPv6, sehingga dapat mendistribusikan IPv6 walaupun ke jaringan internal.

3.3.3.1 ip

Membuat *tunnel* IPv6-in-IPv4:

```
ip tunnel add mytunl mode sit remote 192.168.100.42
ip link set mytunl up
ip addr add 3ffe:1500:5:6::101/64 dev mytunl
ip route add 2000::0/3 via 3ffe:1500:5:6::100
```

Kode III-30. Perintah ip untuk membuat tunnel ipv6-in-ipv4

3.3.3.2 ifconfig dan route

Membuat *tunnel* IPv6-in-IPv4:

```
ifconfig sit0 up
ifconfig sit0 tunnel ::192.168.100.42
ifconfig sit1 up
ifconfig sit1 add 3ffe:1500:0005:0006::101/64
route -A inet6 add 2000::0/3 gw 3ffe:1500:0005:0006::100
```

Kode III-31. Perintah ifconfig untuk membuat tunnel ipv6-in-ipv4

3.3.3.3 Debian

Membuat *tunnel* IPv6-in-IPv4:

```
iface mytunl ine6 v4tunnel
      Address 3ffe:1500:5:6::101
      netmask 64
      endpoint 192.168.100.42
      up ip route add 2000::0/3 via 3ffe:1500:5:6::100
      up ip tunnel change mytunl ttl 64
```

Kode III-32. Konfigurasi pada Debian untuk tunnel ipv6-in-ipv4

BAB IV. IMPLEMENTASI JARINGAN IPV6 DI ITS-NET

4.1. Pendahuluan

Jaringan yang ada di ITS-Net masih menggunakan protokol IPv4. Perangkat jaringan yang dimiliki oleh ITS-Net antara lain:

1. *Router* 3Com Corebuilder 3500 dengan modul Fiber Optik dan Ethernet yang tersebar di beberapa jurusan di ITS.
2. Switch 3Com SuperStack versi 1100 dan 3300, yang tersebar di beberapa jurusan di ITS.
3. *Linux* PC server yang memberikan pelayanan berupa dns server, mail server, web server, proxy server, dan ftp server.

Backbone ITS-Net berada di Perpustakaan ITS lt6, dan menggunakan layanan ISP dari RadNet dengan menggunakan media WaveLAN.

Local Area Network ITS dibedakan menjadi beberapa VLan sehingga memudahkan untuk mengelola jaringan yang ada.

Peta jaringan ITS-Net (lampiran)

4.2. Koneksi ke IPv6 Cloud

Cara yang dipakai di ITS-Net adalah 6in4, yaitu menggunakan *tunnel* IPv6-in-IPv4.

ITS-Net memiliki 3 jalur *tunnel* ke backbone IPv6, yaitu melaui AI3-ITB, Freenet6, dan *Tunnel*broker.net. Setting yang perlu dilakukan adalah membuat konfigurasi pada file /etc/network/*interface*:

```
#Tunnel AI3-ITB
auto mytunl
iface mytunl inet6 v4tunnel
        Address 2001:200:830:131::2
        netmask 128
        endpoint 167.205.23.16
        up route -A inet6 add 2001:200:830::/48 dev mytunl


#Tunnel tunnelbroker.net
auto tun64
iface tun64 inet6 v4tunnel
```

```
        Address 2001:470:1F00:FFFF::21F
        netmask 127
        endpoint 64.71.128.82
        up route -A inet6 add 2001:470:1F00:FFFF::/64 dev tun64
```

Kode IV-1. Konfigurasi tunnel ipv6-in-ipv4

Sedangkan untuk koneksi Freenet6 menggunakan paket aplikasi freenet6 dari Debian. Tambahkan konfigurasi pada file /etc/freenet6/tspc.conf, yaitu:

```
tsp_version=1.0.0
auth_method=any
client_v4=auto
userid=dhidhel-ipv6
passwd=ZC4PUKybhQ
template=setup
server=tsps1.freenet6.net
if_tunnel=sit1
host_Type=router
prefixlen=48
if_prefix=eth0
dns_server=ns.ipv6.its.ac.id
```

Kode IV-2. Konfigurasi freenet6

Kemudian restart aplikasi freenet6 dengan cara /etc/init.d/freenet6 restart.

4.3. Migrasi ke IPv6

4.3.1. *Router* Advertisement

Untuk menyebarkan IPv6 yang didapat, ITS-Net menggunakan model *router* advertisement. *Router* Advertisement yang digunakan adalah RAdvD dan Zebra. Konfigurasi yang perlu dilakukan adalah, menambahkan konfigurasi pada file /etc/radvd.conf untuk RadvD dan /etc/zebra/zebra.conf untuk menggunakan Zebra.

/etc/radvd.conf

```
interface eth0
{
 AdvSendAdvert on;
 prefix 3ffe:0b80:0bb4:0001::/64
 {
 AdvOnLink on;
 AdvAutonomous on;
 };
};
```

Kode IV-3. Konfigurasi RAdvD

/etc/zebra/zebra.conf

```
interface eth0
 no ipv6 nd suppress-ra
 ipv6 nd prefix-advertisement 3ffe:0b80:0bb4:0001::/64 2592000
604800 onlink autoconfig
```

Kode IV-4. Konfigurasi Zebra

4.3.2. Domain Name Server (DNS)

ITS-Net memiliki 2 server DNS yaitu server dragon (202.154.63.2) sebagai master domain its.ac.id dan phoenix (202.154.63.3) sebagai slave domain its.ac.id.

Untuk dapat membedakan apakah penggunaan IPv6 dibuat domain baru yaitu domain ipv6.its.ac.id. Server yang memegang authorisasi domain ipv6.its.ac.id adalah ns.ipv6.its.ac.id. Perlu dilakukan adalah menambahkan zone baru ipv6 pada konfigurasi dns its.ac.id

ns.ipv6	IN	A	202.154.63.3
ipv6	IN	NS	ns.ipv6

Sedangkan pada server 202.154.63.3 menggunakan Bind9, dimana bind9 sudah mendukung protokol IPv6.

Konfigurasi yang diperlukan pada bind9 adalah pada file /etc/bind/named.conf, yaitu dengan menambahkan:

```
Options {
     directory "/var/cache/bind";
     query-source Address * port 53;
     auth-nxdomain no;
     listen-on-v6 { any; };
};
zone "." {
        Type hint;
        file "/etc/bind/db.root";
};
zone "localhost" {
        Type master;
        file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
        Type master;
        file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
        Type master;
        file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
        Type master;
```

```
        file "/etc/bind/db.255";
};
zone "ipv6.its.ac.id" {
   Type master;
   file "/etc/bind/zone/ipv6.its.ac.id";
};
zone "0.0.f.1.0.7.4.0.1.0.0.2.ip6.int" {
    Type master;
    file "/etc/bind/zone/reverse-2001-470-1f00.ip6.int";
    allow-transfer {
      none;
    };
};
```

Kode IV-5. Konfigurasi named.conf pada bind9

Kemudian pada file zone berisi:
/etc/bind/zone/ipv6.its.ac.id

```
@                          IN          SOA         ipv6.its.ac.id.
dhidhel.kebo.vlsm.org. (
                        2002052501      ; serial
                        10800           ; refresh
                        3600            ; retry
                        604800          ; expire
                        86400  )        ; minimum

                IN      NS      ns.ipv6.its.ac.id.

$ORIGIN ipv6.its.ac.id.
gate     IN       A        202.154.63.9
gtw      IN       CNAME    gate

gate     IN       AAAA     2001:470:1f00:371::1

mail     IN       A        202.154.63.7
mail     IN       AAAA     2001:470:1f00:371:210:83ff:fe01:181b

lexican  IN       A        202.154.63.4
lexican  IN       AAAA     2001:470:1f00:371:210:83ff:fe01:181f

ns       IN       A       202.154.63.3
ns       IN       AAAA     2001:470:1f00:371:210:83ff:fe01:1819

www      IN       A        202.154.63.3
www      IN       AAAA     2001:470:1f00:371:210:83ff:fe01:1819

irc      IN       A        202.154.63.8
irc      IN       AAAA     2001:470:1f00:371:210:5aff:fe72:85c5

Pythagoras        IN       A        202.154.63.13
Pythagoras        IN       AAAA
         2001:470:1f00:371:2d0:9ff:feeb:becd

Proxy    IN       A        202.154.63.10
Proxy    IN       AAAA     2001:470:1f00:371:290:27ff:fe61:57fc
```

Kode IV-6. Konfigurasi zone pada bind9

/etc/bind/zone/reverse-2001-470-1f00.ipv6.int

```
$TTL 3D
@       IN      SOA     ipv6.its.ac.id. dhidhel.kebo.vlsm.org. (
                           2002052401  ; serial
                           3H          ; refresh
                           15M         ; retry
                           1W          ; expiry
                           1D )        ; mininum
        IN      NS      ns.ipv6.its.ac.id.
;
;---------------------------------------,
; network part /64   | Host part 64 bits |
;--------------------|-------------------'
; 2001:0470:1f00:0371:0000:0000:0000:0001

;       --------------------------------,
;  Your Network| Your prefix (48 bits) |
;       -------|-------|-------|-------|
$ORIGIN 1.7.3.0.0.0.f.1.0.7.4.0.1.0.0.2.ip6.int.

; Hosts of your network
;------------------------------
; 64 bit host Address part.
;------|-------|-------|-------
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN PTR   gate.ipv6.its.ac.id.

b.1.8.1.1.0.e.f.f.f.3.8.0.1.2.0   IN PTR   mail.ipv6.its.ac.id.

f.1.8.1.1.0.e.f.f.f.3.8.0.1.2.0   IN PTR   lexican.ipv6.its.ac.id.

9.1.8.1.1.0.e.f.f.f.3.8.0.1.2.0   IN PTR   ns.ipv6.its.ac.id.

9.1.8.1.1.0.e.f.f.f.3.8.0.1.2.0   IN PTR   www.ipv6.its.ac.id.

5.c.5.8.2.7.e.f.f.f.a.5.0.1.2.0   IN PTR   irc.ipv6.its.ac.id.

d.c.e.b.b.e.e.f.f.f.9.0.0.d.2.0   IN PTR   pytha.ipv6.its.ac.id.

c.f.7.5.1.6.e.f.f.f.7.2.0.9.2.0   IN PTR   proxy.ipv6.its.ac.id.
; End of zone
```

Kode IV-7. Konfigurasi PTR pada bind9

4.3.3. Web Server

Apache adalah server web yang handal dan paling banyak digunakan oleh pada administrator yang manggunakan system operasi unix. Walaupun banyak digunakan pada system operasi unix, Apache ini juga dapat digunakan pada system operasi Windows NT/9x, 2000, Netware 5.x dan OS/2. Selain handal, Apache adalah

server web yang fleksibel dan mengimplementasikan protokol-protokol web terbaru seperti HTTP/1.1 (RFC 2616). Salah satu sebab kenapa Apache banyak digunakan karena sifat dari software Apache sendiri yang open *source* dan tidak menggunakan lisensi dalam pemakaian software tersebut.

Web server yang digunakan adalah Apache (www.apache.org) versi apache2 2.0.35-1 dimana apache ini sudah mendukung protokol IPv6. Apabila masih ingin menggunakan Apache versi 1.3.x diperlukan patch terlebih dahulu. Patch dapat didownload di ftp://ftp.kame.net/pub/kame/misc, berikut adalah cara installasi apache:

1. Ambil *source* Apache dan patch-nya untuk IPv6 dan tempatkan pada direktory yang sama. Jika sudah, ekstrak kedua file.
```
www# tar zxvf Apache_1.3.19.tar.gz
www# gunzip Apache-1.3.19-v6-20010309a.diff.gz
```
2. Ganti nama direktory apache_1.3.19 menjadi apache13
```
www# mv apache_1.3.19 apache13
```
3. Lakukan patch dengan cara:
```
www# patch < apache-1.3.19-v6-20010309a.diff
```
4. Kompile *source* apache
```
www#   ./configure   -prefix=/usr/local/apache   -enable-
rule=INET6
www# make
www# make install
```
5. Jalankan apache dengan perintah:
```
www# /usr/local/apache/bin/apachectl start
```

Kode IV-8. Instalasi Apache

Apabila menggunakan apache2 dari Debian, tambahkan konfigurasi pada /etc/apt/*source*s.list, mirror debian http://kebo.vlsm.org, yaitu:
deb http://kebo.vlsm.org/debian-extra woody apache2
Kemudian install dengan perintah “apt-get install apache2”.
Konfigurasi yang perlu dilakukan:

1. Buat *link* dari file /usr/share/doc/apache2/examples/highperformance.conf dengan cara :
2. ln –s /usr/share/doc/apache2/examples/highperformance.conf /etc/apache2/httpd.conf
3. Jalankan perintah “add*host* www.ipv6.its.ac.id” untuk membuat virtual *host* www.ipv6.its.ac.id

4. Jalankan perintah "ena*host* www.ipv6.its.ac.id" untuk mengaktifkan virtual *host* www.ipv6.its.ac.id
5. DocumentRoot untuk www.ipv6.its.ac.id berada di /var/lib/v*host*-base/www.ipv6.its.ac.id/htdocs-
6. Start apache dengan perintah "/etc/init.d/apache2 start".

Kode IV-9. Instalasi apache2

4.3.4. Mail Server

Mail server yang digunakan adalah QMail (www.qmail.org). Qmail adalah simple message transfer agent yang aman, reliable, dan efisien. Qmail ditulis oleh D.J Brenstein yang saat ini sebagai asisten Profesor di departemen Matematik, Statistik dan Computer Science Universitas Illinois Chicago. Qmail merupakan pengganti untuk semua sistem sendmail-bindmail pada *host* Unix yang terkoneksi langsung dengan internet.

Agar qmail mampu mendukung protokol alamat IPv6, *source* qmail yang dapat diambil di www.qmail.org harus dipatch terlebih dahulu. Patch IPv6 dapat didownload di http://pyon.org/fujiwara, sedangkan patch untuk tcpserver dapat didownload di http://www.fefe.de/ucspi/ucspi-tcp-0.88-ipv6.diff10.bz2 . Apabila anda menggunakan Distribusi Debian jalankan "apt-get install qmail-src ucspi-tcp-src" untuk mendownload *source* qmail dan tcpserver.

Source Qmail dan Ucspi-tcp berada di direktori /usr/src/qmail-src dan /usr/src/ucspi-tcp-src untuk membongkat *source* Debian diperlukan tools debian-devel dan kemampuan untuk memaintain paket Debian yang dapat dipelajari di www.debian.org .

Membuat paket qmail-ipv6 di Debian:

1. Ekstrak paket qmail-src dengan perintah "dpkg-*source* –x qmail_1.03-24.dsc", kemudian akan terekstrak direktori qmail-1.03
2. Patch qmail dengan patch IPv6
   /usr/src/qmail-src/qmail-1.03/# patch < qmail-1.03-v6-20001010.diff
3. Build paket qmail dengan perintah "dbuild" dan akan terbentuk paket Qmail Debian dengan nama qmail_1.03-24_i386.deb

Membuat paket ucspi-tcp-ipv6 di Debian:

1. Ekstrak paket ucspi-tcp-src dengan perintah "dpkg-*source* –x

ucspi-tcp_0.88-5.dsc", kemudian akan terekstrak direktori ucspi-tcp-0.88.

2. Patch ucspi-tcp dengan patch IPv6
   /usr/src/ucspi-tcp-src/ucspi-tcp-0.88/# patch < ucspi-tcp-0.88-ipv6.diff10
3. Build paket qmail dengan perintah "dbuild" dan akan terbentuk paket Ucspi-TCP Debian dengan nama ucspi-tcp_0.88-5_i386.deb

Installasi:

1. Install terlebih dahulu paket tcp_0.88-5_i386.deb dengan perintah:
   # dpkg –i tcp_0.88-5_i386.deb
2. Install paket qmail debian dengan perintah:
   # dpkg –i qmail_1.03-24_i386.deb
3. Install paket VCHKPW untuk mengaktifkan POP3 daemon, jalankan perintah:
   # apt-get install vchkpw
4. Install paket Courier-IMAP untuk mengaktifkan IMAP server, jalankan perintah:
   # apt-get install courier-imap

Konfigurasi:

1. Edit file /etc/init.d/qmail, supaya mengaktifkan mode Maildir: alias_empty="./Maildir/"
2. Edit file konfigurasi di /etc/qmail/
3. Buat direktori Maildir dengan perintah "maildirmake Maildir"

Jalankan qmail dengan perintah "/etc/init.d/qmail start"
Jalankan vchkpw dengan perintah "/etc/init.d/vchkpw start"
Jalankan courier-IMAP dengan perintah "/etc/init.d/courier-map start"

Kode IV-10. Instalas dan konfigurasi pada Qmail, uscpi-tcp, courier

4.3.5. Proxy Server

Http proxy server yang digunakan adalah Squid (http://www.squid-cache.org/). Squid adalah server proxy cache dengan performa tinggi untuk web client, yang mendukung akan FTP, Gopher, dan data object HTTP.

Mengcache internet object adalah menyimpan data hasil request ke internet bisa dalam bentuk FTP, HTTP, HTTPS di system

yang lebih dekat dari yang meminta request daripada ke sumber. Kemudian web browser dapat menggunakan squid dalam local area network sebagai server proxy HTTP, dan dapat menghemat waktu dan *bandwidth*

Instalasi squid

```
# apt-get install squid squidclient squid-cgi
```

Paket squid ini bergantung atas paket libc6, netbase dan dianjurkan untuk menginstall juga paket squidclient, squid-cgi

File configurasi dari squid adalah /etc/squid.conf. Pada file ini berisikan konfigurasi untuk menjalankan squid. Konfigurasi berisikan tentang port HTTP, port ICP, incoming dan outgoing request, informasi tentang firewall dan beberapa tentang timeout. Pada squid.conf juga berisi tentang penjelasan akan konfigurasi

Untuk menjalankan squid, pertama membuat direktory cache dengan perintah :

```
squid -z
```

kemudian menjalankannya melalui */etc/init.d/squid start*

Paket squid dari Debian belum mendukung IPv6, untuk itu perlu dilakukan kompilasi ulang dan mendownload Squid-CVS, yang perlu dilakukan adalah:

1. Install paket cvs, libpam-dev, libldap2-dev, automake1.5.
2. Buat direktory cvs, dengan perintah “mkdir cvs”
3. Masuk ke server squid cvs dengan cara:
   ```
   cvs -d :pserver:anonymous@cvs.devel.squid-cache.org/cvsroot/squid login
   ```
4. Download squid-cvs dengan IPv6 enable
   ```
   cvs -d :pserver:anonymous@cvs.devel.squid-cache.org/cvsroot/squid co -r ipv6 -d squid-ipv6 squid
   ```
5. Jalankan perintah “./bootstrap.sh”
6. Patch squid dengan patch IPv6 yang dapat didownload di ftp://ftp.bieringer.de/pub/*linux*/IPv6/squid/
   ```
   # patch -p1 -b < squid-2.5-make.patch
   # patch -p1 -b < squid-2.5-config.patch
   # patch -p1 -b < squid-perl*path*.patch
   # patch -p1 < squid-location.patch
   # patch -p0 < squid-2.5-ipv6config.patch
   # patch -p0 < squid-2.5-ipv6onlyrequestbug.patch
   ```
7. Lakukan Installasi dengan perintah:
   ```
   make clean
   ./configure \
   --enable-ipv6 \
   --exec_prefix=/usr \
   --bindir=/usr/sbin \
   --libexecdir=/usr/lib/squid \
   --localstatedir=/var \
   ```

```
        --sysconfdir=/etc \
        --enable-poll \
        --enable-removal-policies="heap,lru" \
        --enable-storeio="aufs,diskd,ufs" \
        --enable-carp \
        --with-pthreads \
        --enable-basic-auth-helpers="LDAP,NCSA,PAM,SMB,MSNT" \
        --enable-cache-digest \
        --enable-underscores \
        --enable-referer-log \
        --enable-useragent-log \
        --enable-async-io=64
        make
        make install
```

8. Konfigurasi file /etc/squid.conf:

```
        Binding of Addresses
                udp_outgoing_Address ::
                udp_incoming_Address ::
                tcp_outgoing_Address ::
        ACLs
                acl localhost src ::ffff:127.0.0.1 ::1
                acl to_localhost dst ::ffff:127.0.0.1 ::1
                acl all src ::/0
                acl sitelocalsrc src fec0::/48
                acl linklocalsrc src fe80::/64
                acl globaldst dst 3ffe::/16 2000::/3
                acl ipv4src src ::ffff:0:0/96
                acl ipv4dst dst ::ffff:0:0/96
        Disable WCCP
                wccp_router ::
```

9. Jalankan Squid dengan perintah "/etc/init.d/squid start"

Kode IV-11. Instalasi dan konfigurasi Squid

4.3.6. FTP server

Ftp server yang digunakan dan sudah mendukung protokol IPv6 adalah pure-ftpd (http://pureftpd.*source*forge.net) . Untuk installasi pure-ftpd, tambahkan terlebih dahulu file /etc/apt/*source*s.list dengan:

```
deb http://kebo.vlsm.org/debian-extra woody pureftpd
```

Kemudian install paket pure-ftpd dengan perintah "apt-get install pure-ftpd".

Jalankan pure-ftpd dengan mode standalone (daemon), dengan perintah "/etc/init.d/pure-ftpd start".

4.3.7. Ssh (Secure Shell)

SSH adalah program untuk login ke mesin secara remote dan dengan ssh ini dapat menjalankan program selayaknya pada

shell. SSH memiliki jaringan komunikasi yang terenkripsi dengan aman dan dapat menghubungkan dua *host* yang memiliki jaringan yang tidak ketat keamanannya. X11 dan TCP/IP dapat di forwardkan melalui SSH ini. SSH ini sebagai pengganti rlogin, rsh, rcp dan dapat digunakan sebagai komunikasi yang aman.

Installasi SSH di Debian dapat dilakukan dengan perintah "apt-get install ssh". SSH sudah mendukung protokol IPv6. Untuk menjalankan daemon sshd, jalankan perintah "/etc/init.d/ssh start".

4.3.8. Firewall

Firewall yang mendukung protokol IPv6 hanya ada di kernel 2.4.x. Firewall yang digunakan menggunakan paket netfilter6/ip6tables (http://www.netfilter.org) .

Sebelum menggunakan firewall dengan netfilter6 kernel harus sudah mendukung "packet filtering IPv6", apabila belum diharuskan untuk kompilasi ulang kernel dengan mengikuti konfigurasi berikut:

```
Code maturity level Options
Prompt for development and/or incomplete code/drivers : yes

Networking Options
Network packet filtering: yes
The IPv6 protocol: module
IPv6: Netfilter Configuration
IP6 tables support: module

All new Options like following:
limit match support: module
MAC Address match support: module
Multiple port match support: module
Owner match support: module
netfilter MARK match support: module
Aggregated Address check: module
Packet filtering: module
REJECT target support: module
LOG target support: module
Packet mangling: module
MARK target support: module
```

Kode IV-12. Kernel yang mendukung firewall IPv6

Kemudian kompilasi kernel dengan perintah "make dep install modules modules_install", restart mesin dengan menggunakan kernel yang baru.

Load module untuk mengukung IPv6 firewall dengan perintah:

```
# modprobe ip6_tables
```

Paket firewall utility di Debian bernama iptables, untuk installasi iptables jalankan perintah “apt-get install iptables”.

Untuk menggunakan firewall IPv6, gunakan perintah ip6_tables, contoh:

Melist daftar firewall

```
# ip6_tables –nL
```

Memperbolehkan SSH dari 3ffe:400:100::1/128

```
# ip6tables -A INPUT -i sit+ -p tcp -s 3ffe:400:100::1/128 --sport
512:65535 --dport 22 -j ACCEPT
```

BAB V. PENGUJIAN JARINGAN IPV6

5.1. Koneksi ke IPv6 Cloud

Untuk memastikan apakah koneksi ke IPv6 cloud sudah aktif dapat dilakukan dengan menggunakan utility ping6 dan traceroute6 yang merupakan paket dari iputils.

5.1.1. Ping6 vs Ping

Ping6 dari gate.ipv6.its.ac.id ke www.6bone.net :

```
pts/1: dhidhel on desperate using Linux 2.4.18-kebo at Sun May 26
21:00:09
~$ ping6 www.6bone.net
PING www.6bone.net(www.6bone.net) 56 data bytes
64 bytes from www.6bone.net: icmp_seq=1 ttl=62 time=876 ms
64 bytes from www.6bone.net: icmp_seq=2 ttl=62 time=912 ms
64 bytes from www.6bone.net: icmp_seq=3 ttl=62 time=759 ms
64 bytes from www.6bone.net: icmp_seq=4 ttl=62 time=777 ms

--- www.6bone.net ping statistics ---
5 packets transmitted, 4 received, 20% loss, time 4041ms
rtt min/avg/max/mdev = 759.891/831.809/912.987/64.669 ms
```

Apabila dibandingkan dengan ping menggunakan jalur IPv4:

```
pts/1: dhidhel on desperate using Linux 2.4.18-kebo at Sun May 26
21:00:39
~$ ping www.6bone.net
PING 6bone.net (131.243.129.43) from 202.154.63.9 : 56(84) bytes
of data.
64 bytes from 6bone.net (131.243.129.43): icmp_seq=1 ttl=110
time=2258 ms
64 bytes from 6bone.net (131.243.129.43): icmp_seq=2 ttl=110
time=2401 ms
64 bytes from 6bone.net (131.243.129.43): icmp_seq=3 ttl=110
time=2402 ms
64 bytes from 6bone.net (131.243.129.43): icmp_seq=4 ttl=110
time=1790 ms

--- 6bone.net ping statistics ---
6 packets transmitted, 4 received, 33% loss, time 10395ms
rtt min/avg/max/mdev = 1790.629/2213.557/2402.956/251.124 ms, pipe
3
```

Perbedaan yang terjadi pada waktu yang ditempuh oleh paket icmp, dimana apabila menggunakan IPv6 koneksi yang terjadi bisa lebih cepat. Hal ini dikarena pada IPv6 pemrosesan terhadap paket yang

datang lebih sedikit, dan karena adanya *routing* protokol dari *backbone*.

5.1.2. Traceroute6 vs Traceroute

Traceroute6 dari gate.ipv6.its.ac.id ke www.6bone.net:

```
pts/1: dhidhel on desperate using Linux 2.4.18-kebo at Sun May 26
21:06:22
~$ traceroute6 www.6bone.net
traceroute      to      6bone.net      (3ffe:b00:c18:1::10)      from
3ffe:b80:2:4d50::2, 30 hops max, 16 byte packets
 1                                3ffe:b00:c18:1:2a0:c9ff:fefc:1feb
(3ffe:b00:c18:1:2a0:c9ff:fefc:1feb)     772.616   ms     778.633   ms
764.566 ms
 2   www.6bone.net  (3ffe:b00:c18:1::10)   755.883  ms   766.201  ms
810.11 ms
```

Apabila dibandingkan dengan traceroute dengan menggunakan protokol IPv4:

```
pts/1: dhidhel on desperate using Linux 2.4.18-kebo at Sun May 26
21:06:34
~$ traceroute www.6bone.net
traceroute  to  6bone.net  (131.243.129.43),  30  hops  max,  38  byte
packets
 1  gateway (202.154.63.1)  0.423 ms  0.375 ms  0.367 ms
 2  ror.sby.rad.net.id (202.154.56.9)  3.550 ms  6.772 ms  2.690
ms
 3   noc2-bb.pop-ether.sby.rad.net.id (202.154.56.1)   1662.347 ms
1216.687 ms  1
519.239 ms
 4     cisco-2660.sby.rad.net.id   (202.154.57.2)     1421.711   ms
1176.251 ms  1381.809
 ms
 5   202.59.196.17  (202.59.196.17)   2058.886  ms   2096.316  ms
1779.766 ms
 6   Serial0-0-0.GW5.HKG2.ALTER.NET  (202.130.131.1)   2235.021  ms
2107.699 ms  20
88.568 ms
 7   * 25.so-3-0-0.XR2.HKG2.ALTER.NET (210.80.2.245)   1995.697 ms
1771.025 ms
 8    0.so-6-0-0.TR2.HKG2.ALTER.NET  (210.80.48.33)    2048.653  ms
2210.304 ms *
 9   0.so-3-1-0.IR2.LAX12.Alter.Net  (210.80.49.162)   1605.595  ms
2251.823 ms  2018.373 ms
10     POS3-0.IR2.LAX9.ALTER.NET   (137.39.31.242)     1577.970   ms
688.423 ms POS2-0.IR2.LAX9.ALTER.NET (137.39.31.238)  684.241 ms
11  152.63.0.150 (152.63.0.150)  684.715 ms  678.110 ms  843.308
ms
12    0.so-1-0-0.TL2.SAC1.ALTER.NET  (152.63.15.250)    726.009  ms
728.515 ms  891.661 ms
13    0.so-7-0-0.XL2.SJC1.ALTER.NET  (152.63.55.133)    711.575  ms
745.961 ms  717.634 ms
```

```
14  POS1-0.XR2.SJC1.ALTER.NET (152.63.55.125)  676.765 ms  698.751
ms  684.184 ms
15    192.ATM4-0.BR2.SJC1.ALTER.NET  (152.63.51.177)    693.997  ms
709.925 ms  730.029 ms
16    204.255.174.50  (204.255.174.50)    698.331  ms    862.276  ms
694.756 ms
17    lbl-snv-oc48.es.net  (134.55.209.6)    690.115  ms    686.159  ms
698.719 ms
18    lbnl-ge-lbl2.es.net  (198.129.224.1)    750.239  ms    724.277  ms
685.847 ms
19    ir40gw.lbl.gov  (131.243.128.40)    691.104  ms    696.347  ms
691.351 ms
20  6bone.net (131.243.129.43)  726.098 ms  692.494 ms  714.263 ms
```

HOP yang dilalui dengan traceroute6 hanya diperlukan 2 hop, tetapi apabila menggunakan traceroute dari IPv4 memerlukan 20 hop.

5.2. DNS

Bind9 yang terpasang di ns.ipv6.its.ac.id sudah mendukung akan protokol IPv6. Untuk mengetahui apakah Bind9 sudah dapat menerima request dengan menggunakan protokol IPv6 dapat dilakukan dengan cara:

```
# netstat –nlptu | grep named
```

```
pts/0: root on phoenix using Linux 2.4.18 at Sun May 26 21:14:04
~# netstat -nlptu | grep named
tcp            0          0  127.0.0.1:953                0.0.0.0:*
LISTEN      558/named
tcp            0          0  :::53                             :::*
LISTEN      558/named
tcp            0          0  ::1:953                           :::*
LISTEN      558/named
udp            0          0  0.0.0.0:53                   0.0.0.0:*
558/named
udp            0        776  202.154.63.3:53              0.0.0.0:*
558/named
udp            0          0  127.0.0.1:53                 0.0.0.0:*
558/named
udp            0          0  :::53                             :::*
558/named
```

Untuk mengecek apakah domain ipv6.its.ac.id sudah diwakili oleh IPv6, dapat dilakukan dengan:

```
# host –vl –t any ipv6.its.ac.id
```

```
pts/0: root on phoenix using Linux 2.4.18 at Sun May 26 21:14:08
~# host -vl -t any ipv6.its.ac.id
Trying "ipv6.its.ac.id"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46595
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0,
ADDITIONAL: 2
```

```
;; QUESTION SECTION:
;ipv6.its.ac.id.                        IN      ANY

;; ANSWER SECTION:
ipv6.its.ac.id.          86400    IN       SOA      ipv6.its.ac.id.
dhidhel.kebo.vlsm.org. 2002052602 10800 3600 604800 86400
ipv6.its.ac.id.         86400   IN      NS      ns.ipv6.its.ac.id.

;; ADDITIONAL SECTION:
ns.ipv6.its.ac.id.      86400   IN      A       202.154.63.3
ns.ipv6.its.ac.id.               86400        IN               AAAA
3ffe:b80:bb4:1:210:83ff:fe01:1819

Received 150 bytes from ::1#53 in 3 ms
```

5.3. Web

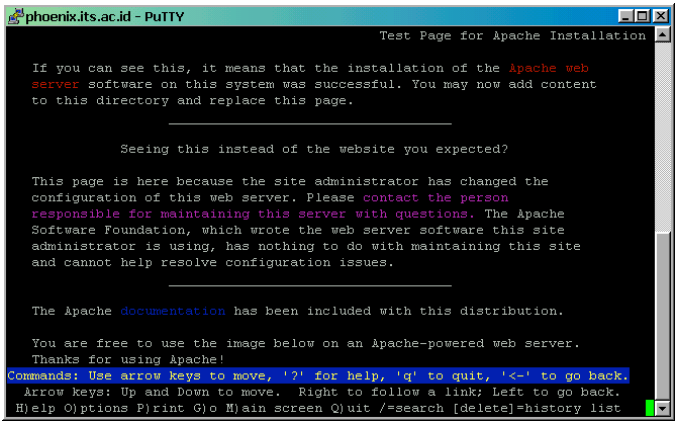
Apache2 yang terpasang di server www.ipv6.its.ac.id aka 202.154.63.3 sudah mendukung akan protokol IPv6. Untuk mengetahui apakah Apache2 sudah dapat menerima request dengan menggunakan protokol IPv6 dapat dilakukan dengan cara:

```
# netstat –nlptu | grep apache
```

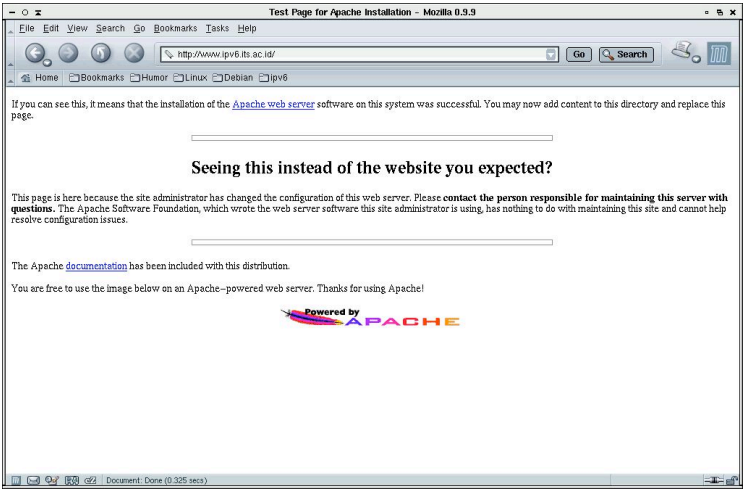
```
pts/0: root on phoenix using Linux 2.4.18 at Sun May 26 20:53:30
~# netstat -nlptu | grep apache
tcp            0          0  :::80                                :::*
LISTEN      200/apache2
```

Hasil dari netstat menunjukkan port 80 dari apache2 sudah siap menerima request.

Browser client yang sudah mensupport akan protokol IPv6 masih dibilang sedikit. Browser yang sudah mendukung IPv6 di *Linux* antara lain lynx dan Mozilla.

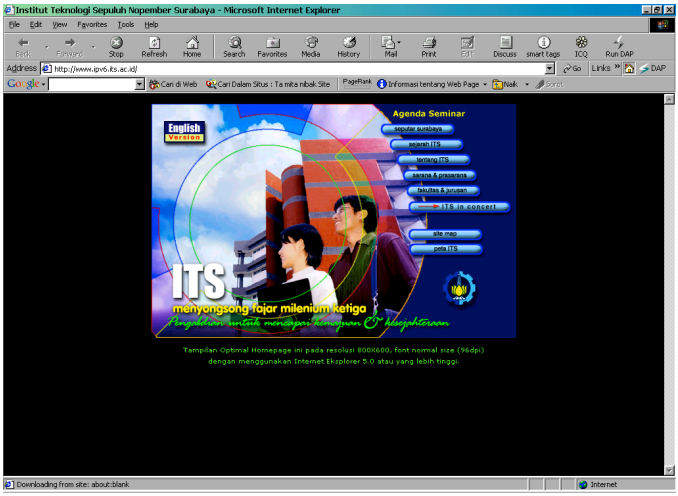


Gambar V.1 Lynx pada putty (www.ipv6.its.ac.id)



Gambar V.2 Mozilla (www.ipv6.its.ac.id)

Apabila menggunakan Internet Explorer yang notabene hanya mendukung IPv4, www.ipv6.its.ac.id akan terlihat:



Gambar V.3 Internet Explorer (www.ipv6.its.ac.id)

Pada log Apache2, terlihat yang sudah mengakses www.ipv6.its.ac.id:

```
pts/0: root on phoenix using Linux 2.4.18 at Sun May 26 21:10:09
/var/lib/vhost-base/www.ipv6.its.ac.id/logs# tail -f access.log-
3ffe:b80:bb4:1::1   -   -   [26/May/2002:20:44:52   +0700]   "GET
/apache_pb.gif HTTP/
 200 2326 "http://www.ipv6.its.ac.id/" "Mozilla/5.0 (X11; U; Linux
i686; en-
rv:0.9.9) Gecko/20020412 Debian/0.9.9-6"
3ffe:b80:bb4:1::1   -   -   [26/May/2002:20:57:17   +0700]   "GET   /
HTTP/1.0" 200 1456
 "Lynx/2.8.4rel.1 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.6c"
3ffe:b80:bb4:1::1   -   -   [26/May/2002:20:57:47   +0700]   "GET   /
HTTP/1.0" 200 1456
 "Lynx/2.8.4rel.1 libwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.6c"
::1 - - [26/May/2002:20:58:10 +0700] "GET / HTTP/1.0" 200 1456 "-"
"Lynx/2.8
1.1 libwww-FM/2.14"
```

5.4. Mail

Qmail yang terpasang di server phoenix.ipv6.its.ac.id, phytagoras.ipv6.its.ac.id dan mail.its.ac.id sudah mendukung protokol IPv6. Untuk mengetahui apakah Qmail sudah siap menerima request SMTP dapat dilakukan dengan cara:

```
# netstat -nlptu | grep 25
```

```
pts/1: root on phoenix using Linux 2.4.18 at Sun May 26 21:26:46
~# netstat -nlptu | grep 25
tcp            0          0 :::25                                 :::*
LISTEN      775/tcpserver
```

Untuk mengetahui apakah POP3 (110), IMAP (143) server sudah mendukung IPv6 dapat dilakukan dengan cara:

```
# netstat -nlptu
```

```
pts/1: root on phoenix using Linux 2.4.18 at Sun May 26 21:26:46
tcp            0          0 :::110                                :::*
LISTEN      323/tcpserver
tcp            0          0 :::143                                :::*
LISTEN      223/couriertcpd
```

Dicobakan mengirim mail dari dhidhel@phoenix.ipv6.its.ac.id ke dhidhel@phoenix.ipv6.its.ac.id dan dhidhel@mail.its.ac.id ke dhidhel@phoenix.ipv6.its.ac.id, hasil yang dapat di lihat di /var/log/mail.log

```
May 26 21:22:29 phoenix qmail: 1022422949.611818 status: local
0/10 remote 0/20
May 26 21:22:29 phoenix qmail: 1022422949.627430 end msg 49919
May 26 21:22:42 phoenix qmail: 1022422962.469480 new msg 49919
May 26 21:22:42 phoenix qmail: 1022422962.470618 info msg 49919:
bytes 493 from <dhidhel@phoenix.ipv6.its.ac.id> qp 782 uid 1000
May 26 21:22:42 phoenix qmail: 1022422962.564287 starting delivery
2: msg 49919 to local dhidhel@phoenix.ipv6.its.ac.id
May 26 21:22:42 phoenix qmail: 1022422962.565023 status: local
1/10 remote 0/20
May 26 21:22:42 phoenix qmail: 1022422962.656362 delivery 2:
success: did_1+0+0/
May 26 21:22:42 phoenix qmail: 1022422962.657559 status: local
0/10 remote 0/20
May 26 21:22:42 phoenix qmail: 1022422962.658509 end msg 49919
q
May 26 21:23:22 phoenix qmail: 1022423002.623150 new msg 49919
May 26 21:23:22 phoenix qmail: 1022423002.624132 info msg 49919:
bytes 653 from <dhidhel@its.ac.id> qp 788 uid 64011
May 26 21:23:22 phoenix qmail: 1022423002.727841 starting delivery
3: msg 49919 to local dhidhel@phoenix.ipv6.its.ac.id
May 26 21:23:22 phoenix qmail: 1022423002.728544 status: local
1/10 remote 0/20
May 26 21:23:22 phoenix qmail: 1022423002.779035 delivery 3:
success: did_1+0+0/
May 26 21:23:22 phoenix qmail: 1022423002.780243 status: local
0/10 remote 0/20
May 26 21:23:22 phoenix qmail: 1022423002.781187 end msg 49919
```

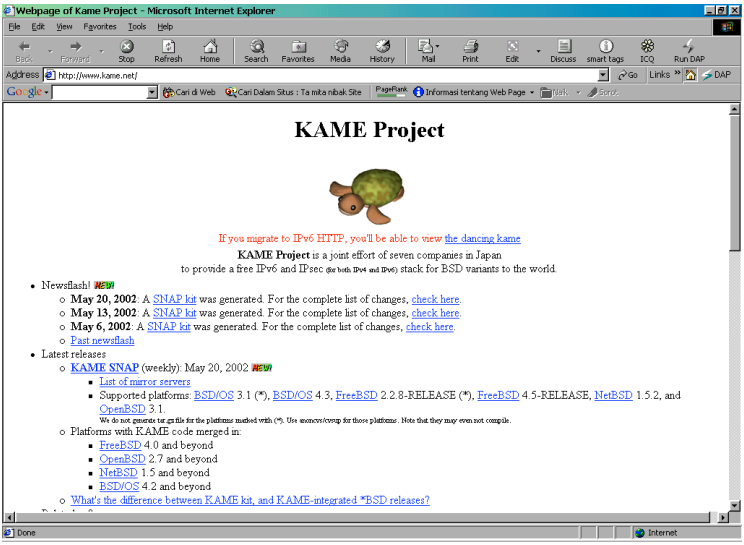
5.5. Proxy

Squid-2.5Devel yang terpasang di proxy.ipv6.its.ac/proxy.its.ac.id sudah mendukung protokol IPv6. Untuk mengetahui apakah squid sudah mendukung IPv6 dapat dilakukan dengan cara:

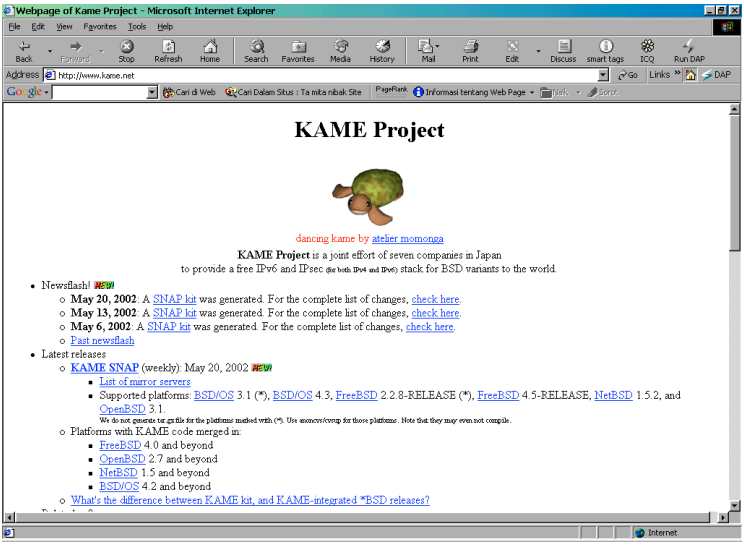
```
# netstat -nlptu | grep squid
```

```
pts/0: root on cerberus using Linux 2.4.18 at Sun May 26 21:25:47
~# netstat -nlptu | grep squid
tcp        0      0 :::8080                 :::*
LISTEN      1053/(squid)
udp        0      0 :::32773                :::*
1053/(squid)
udp        0      0 :::3130                 :::*
1053/(squid)
```

Diuji cobakan dengan menggunakan Internet Explorer yang pertama tidak menggunakan proxy dan berikutnya menggunakan proxy. *Site* yang dibuka adalah www.kame.net



Gambar V.4 IE dengan proxy IPv4 (www.kame.net)



Gambar V.5 IE dengan proxy IPv6 (www.kame.net)

Terjadi perbedaan, apabila menggunakan proxy IPv4, Internet Explorer hanya merequest ke www.kame.net (203.178.141.220) tetapi apabila menggunakan proxy IPv6, Internet Explorer merequest www.kame.net (2001:200:0:4819:210:f3ff:fe03:4d0). Selain itu perbedaan yang terjadi apabila menggunakan proxy IPv4 gambar kura-kura dari www.kame.net tidak dapat berdansa berbeda apabila menggunakan proxy yang sudah mendukung IPv6.

Log yang muncul di file /var/log/squid/access.log terlihat berbeda dengan log yang dimiliki oleh proxy IPv4. Berikut ini log dari squid-ipv6:

```
pts/0: root on cerberus using Linux 2.4.18 at Sun May 26 21:39:44
~# tail -f /var/log/squid/access.log
::ffff:202.154.63.13   -   -   [26/May/2002:21:35:43   +0700]   "GET
http://www.k/ HTTP/1.0" 503 1121 TCP_MISS:NONE
::ffff:202.154.63.13   -   -   [26/May/2002:21:35:49   +0700]   "GET
http://www.kame.net/ HTTP/1.0" 200 9463 TCP_HIT:NONE
::ffff:202.154.63.13   -   -   [26/May/2002:21:35:50   +0700]   "GET
http://www.kame.net/img/new.png HTTP/1.0" 200 509 TCP_HIT:NONE
::ffff:202.154.63.13   -   -   [26/May/2002:21:35:50   +0700]   "GET
http://www.kame.net/img/kame-anime-small.gif  HTTP/1.0"  200  54095
```

```
TCP_HIT:NONE
::ffff:202.154.63.13  -  -  [26/May/2002:21:35:50  +0700]  "GET
http://www.kame.net/img/vpnc-test-partner.gif HTTP/1.0" 200 1112
TCP_HIT:NONE
::ffff:202.154.63.13  -  -  [26/May/2002:21:35:50  +0700]  "GET
http://www.kame.net/logo/1/images/kame3.png  HTTP/1.0"  200  6035
TCP_HIT:NONE
::ffff:202.154.63.13  -  -  [26/May/2002:21:35:50  +0700]  "GET
http://www.momonga.org/icon/momo-b2.gif   HTTP/1.0"   200   16509
TCP_HIT:NONE
::ffff:202.154.63.13  -  -  [26/May/2002:21:38:17  +0700]  "GET
http://www.google.com/search? HTTP/1.0" 500 1066 TCP_MISS:DIRECT
::ffff:10.10.100.199  -  -  [26/May/2002:21:39:22  +0700]  "GET
http://liveupdate.symanteclive update.com/autoupdt.trg    HTTP/1.1"
403 1088 TCP_DENIED:NONE
::ffff:10.10.100.199 - - [26/May/2002:21:39:22 +0700] "GET
http://liveupdate.symantec.com/autoupdt.trg HTTP/1.1" 403 1068
TCP_DENIED:NONE
```

BAB VI. PENUTUP

Setelah melalui berbagai percobaan, dan perbaikan kesalahan, jaringan IPv6 di ITS-Net ini dapat berjalan dengan cukup baik. Ada beberapa kesimpulan yang dapat diambil dan saran tentang apa yang diharapkan dapat digunakan dalam implementasi jaringan IPv6 tersebut.

6.1. Kesimpulan

Beberapa kesimpulan yang dapat diambil dari hasil laporan tugas akhir ini antara lain :

1. Jaringan IPv6 dapat di implementasikan diatas jaringan IPv4 dengan menggunakan *Tunnel* IPv6-in-IPv4.
2. Dengan menggunakan koneksi IPv6, ping timeout bisa dikurangi, dan kebutuhan akan hop menuju IPv6 cloud lebih pendek.
3. Terdapat aplikasi server yang sudah mendukung IPv6 seperti: Bind9, Apache2, Qmail, courier-IMAP, Squid, SSH dan Firewall (ip6tables).
4. Jaringan IPv6 dapat diimplementasikan menggunakan sistem operasi *Linux*. Dalam hal ini yang digunakan adalah distribus Debian 2.2 Potato dan Debian 3.0 Woody.
5. Jaringan IPv6 bisa lebih efektif dengan menggunakan protokol *routing* seperti OSPFv3, BGP4+, RIPng.

6.2. Saran

Adapun saran yang dapat digunakan untuk pengembangan dan implementasi IPv6 dengan sistem operasi *Linux* ini adalah :

1. Diperlukan kemahiran dan pengalaman tentang jaringan.
2. Menguasai networking dengan menggunakan *Linux*, dan menguasai CLI *Linux*.
3. Tidak terlalu diperlukan hardware yang canggih.
4. Media sistem yang digunakan sebaiknya menggunakan ethernet dengan kecepatan 100Mbps.

5. Masih diperlukan penelitian aplikasi apa saja yg dapat menggunakan protokol IPv6.

DAFTAR PUSTAKA

A. Conta, S. Deering, 1998, **Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,** Request for Comments 2463.

A. Durand, B. Buclin, 1999, **6Bone *Routing* Practice,** Request for Comments 2546.

A. Narten, E. Nordmark, W. Simpson, 1998, **Neighbor *Discovery* for IP Version 6 (IPv6)**, Request for Comments 2461.

B. Carpenter, C. Jung, 1999, **Transmission of IPv6 over IPv4 Domains without Explicit *Tunnel*s,** Request for Comments 2529.

Craig Small, **Debian GNU/*Linux* IPv6,** <http://people.debian.org/~csmall/>.

D. Johnson, S. Deering, 1999, ***Reserved* IPv6 Subnet *Anycast Address*es,** Request for Comments 2526.

Dhidhel Dedel Duel, 2002, **Dhidhel's Debian Mirror**, <http://kebo.vlsm.org/debian-ipv6>.

Florent Parent, Regis Desmeules, 2000, **IPv6 Tutorial (ipv6forum.pdf)**, <http://www.ipv6forum.com>.

G. Malkin, R. Minnear, 1997, **RIPng for IPv6**, Request for Comments 2080.

J. McCann, S. Deering, J. Mogul, 1996, ***Path* MTU *Discovery* for IP version 6,** Request for Comments 1981.

M. Crawford, 1998, **Transmission of IPv6 Packets over Ethernet Networks,** Request for Comments 2464.

P. Marques, F. Dupont, 1999, **Use of BGP-4 Multiprotocol *Extension*s for IPv6 Inter-Domain *Routing*,** Request for Comments 2545.

Peter Bieringer, 2002, ***Linux* IPv6,** <http://www.bieringer.de/*linux*/IPv6/>.

R. Coltun, D. Ferguson, J. Moy, 1999, **OSPF for IPv6,** Request for Comments 2740.

R. den, M. O'Dell, S. Deering, 1998, **An IPv6 Aggregatable Global *Unicast Address* Format**, Request for Comment 2374.

R. Gilligan, E. Nordmark, 1996, **Transition Mechanisms for IPv6 *Host*s and *Router*s,** Request for Comments 1933.

R. Hinden, 1998, **Proposed TLA and NLA Assigment Rule**, Request for Comments 2450.

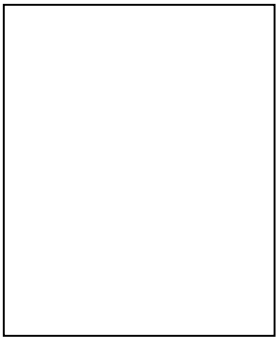
R. Hinden, B. Carpenter, L. Masinter, 1999, **Format for Literal IPv6 *Address*es in URL's**, Request for Comments 2732.
R. Hinden, R. Fink, J. Postel, 1998, **IPv6 Testing *Address* Allocation,** Request for Comments 2471.
R. Hinden, S. Deering, 1998, **IP Version 6 *Address*ing Architecture**, Request for Comments 2373.
R. Rockell, R. Fink, 2000, **6Bone Backbone *Routing* Guidelines**, Request for Comments 2772.
Riza Taufan, 2002, **Buku Pintar Internet: Teori dan Implementasi IPv6 Protokol Internet Masa Depan**, ElexMedia Komputindo.
S. Deering, R. Hinden, 1998, **Internet Protocol, Version 6 (IPv6) Specification**, Request for Comments 2460.
S. Thompson, T. Narten, 1998, **IPv6 Stateless *Address* Autoconfiguration**, Request for Comments 2462.
Thompson, S., Huitema, C., 1995, **DNS *Extension*s to support IP version 6,** Request for Comments 1886.
Viagenie Inc, 1998, **Freenet6 Free IPv6 Connectivity**, <http://www.freenet6.net>.

LAMPIRAN

USULAN TUGAS AKHIR

Proposal Tugas Akhir

DAFTAR RIWAYAT HIDUP



Nama	S. Sukaridhoto
Tempat/Tgl. Lahir	Surabaya / 6 Maret 1979
Agama	Islam
Nama Ayah	Prof. Dr. Ir. S. Sukardjono (alm)
Nama Ibu	Ida Rodiah SE
Alamat	Perum ITS B-2 Surabaya

Penulis adalah putra tunggal.
Riwayat Pendidikan :
TKK Santa Clara (1984-1985)
SDK Santa Clara , Surabaya (1985-1991)
SMPN 6, Surabaya (1991-1994).
SMU Trimurti, Surabaya (1994-1997)

Diterima di Jurusan Teknik Elektro ITS Surabaya melalui Ujian Masuk Perguruan Tinggi Negeri pada tahun 1997 dan mengambil bidang studi Teknik Sistem Komputer.

Riwayat Organisasi :
Pengurus OSIS SMPN 6 Surabaya (1992-1993)
Pengurus Himpunan T Elektro (1997-1999)
Kelompok Linux Arek Suroboyo (1999-sekarang)
VLSM http://kebo.vlsm.org (2000-sekarang)