PERBANDINGAN KINERJA APLIKASI FTP *SERVER* PADA JARINGAN NAT *FULL CONE* DENGAN *TUNNELING* IPv6 TEREDO TERHADAP JARINGAN NAT *FULL CONE* IPv4 MURNI DAN JARINGAN IPv6 MURNI

SKRIPSI

Oleh

FICKY FATTURRAHMAN

04 03 03 042X

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GANJIL 2007/2008

PERBANDINGAN KINERJA APLIKASI FTP *SERVER* PADA JARINGAN NAT *FULL CONE* DENGAN *TUNNELING* IPv6 TEREDO TERHADAP JARINGAN NAT *FULL CONE* IPv4 MURNI DAN JARINGAN IPv6 MURNI

SKRIPSI

Oleh

FICKY FATTURRAHMAN

04 03 03 042X

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GANJIL 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

PERBANDINGAN KINERJA APLIKASI FTP *SERVER* PADA JARINGAN NAT *FULL CONE* DENGAN *TUNNELING* IPv6 TEREDO TERHADAP JARINGAN NAT *FULL CONE* IPv4 MURNI DAN JARINGAN IPv6 MURNI

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau di instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 8 Januari 2007

Ficky Fatturrahman

NPM 04 03 03 042X

PENGESAHAN

Skripsi dengan judul :

PERBANDINGAN KINERJA APLIKASI FTP *SERVER* PADA JARINGAN NAT *FULL CONE* DENGAN *TUNNELING* IPv6 TEREDO TERHADAP JARINGAN NAT *FULL CONE* IPv4 MURNI DAN JARINGAN IPv6 MURNI

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi. Skripsi ini telah diujikan pada sidang ujian skripsi pada tanggal 3 Januari 2008 dan dinyatakan memenuhi syarat/sah sebagai skripsi pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, 3 Januari 2008

Dosen Pembimbing,

Ir. A. Endang Sriningsih, MT

NIP 130 781 318

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

Ir. A. Endang Sriningsih, MT

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Ficky Fatturrahman
NPM 04 03 03 042 X
Departemen Teknik Elektro

Dosen Pembimbing
Ir. A. Endang Sriningsih, MT

PERBANDINGAN KINERJA APLIKASI FTP SERVER PADA JARINGAN NAT FULL CONE DENGAN TUNNELING IPv6 TEREDO TERHADAP JARINGAN NAT FULL CONE IPv4 MURNI DAN JARINGAN IPv6 MURNI

ABSTRAK

Persediaan alamat IPv4 semakin menipis. Salah satu solusi untuk mengatasinya adalah *Network Address Translation* (NAT). Dengan NAT, lebih dari satu *host* pada jaringan *private* dapat dihubungkan ke jaringan publik seperti Internet hanya dengan menggunakan satu alamat IP publik.

Sementara itu, migrasi menuju teknologi yang lebih maju yakni IPv6 sudah tidak terelakkan. Untuk itu diperlukan suatu mekanisme transisi yang memungkinkan *coexistence* antara jaringan IPv6 yang akan dibangun dengan jaringan IPv4 yang telah ada, salah satunya adalah dengan *tunneling*.

Pada umumnya, sebagian besar metode *tunneling* yang ada tidak mendukung jaringan NAT IPv4. Hanya metode *tunneling* Teredo yang dapat menembus jaringan NAT. Untuk itu, perlu dilakukan penelitian khusus mengenai kinerja jaringan yang menggunakan tunneling IPv6 Teredo pada aplikasi-aplikasi tertentu, terutama aplikasi yang populer digunakan seperti FTP untuk transfer file antar jaringan. Penelitian yang dilakukan adalah membandingkan kinerja aplikasi FTP *server* pada jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo terhadap jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni pada aplikasi yang sama. Parameter yang dibandingkan adalah *latency* (s) dan *throughput* (KBytes/s). Hasil penelitian menunjukkan bahwa *tunneling* Teredo memiliki kinerja lebih buruk dari jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni pada aplikasi FTP *server*, namun demikian, tidak terlampau jauh kinerjanya dari jaringan IPv6 murni pada simulasi jaringan WAN sebenarnya, hanya sedikit lebih buruk dengan *range latency* lebih besar 7,4 - 28,08 % dan *range throughput* lebih kecil 2,89 - 16,55 % dari jaringan IPv6 murni, sehingga Teredo cocok digunakan untuk memberikan koneksi IPv6 kepada *node* jaringan di belakang NAT IPv4 pada periode transisi nanti ketika sebagian besar *node* telah bermigrasi ke IPv6.

Kata kunci : IPv6, NAT,Teredo *tunneling*, FTP.

Ficky Fatturrahman
NPM 04 03 03 042 X
Electrical Engineering Department

Counsellor
Ir. A. Endang Sriningsih, MT

COMPARISON OF FTP SERVER APLLICATION PERFORMANCE ON FULL CONE NAT CONFIGURED NETWORK WITH IPv6 TEREDO TUNNELING TOWARD PURE IPv4 FULL CONE NAT NETWORK AND PURE IPv6 NETWORK

ABSTRACT

Availability of IPv4 address has gone thinner. One of the solutions to overcome this problem is Network Address Translation (NAT). NAT can connect one or more hosts in private network to public network like Internet with just one public IP address.

Meanwhile, migration process into more advanced technology, which is IPv6, is inevitable. Therefore, we need transition mechanism that can provide coexistence between newly born IPv6 networks with old IPv4 networks, such as is tunneling.

Generally, most of available tunneling methods do not provide IPv4 NAT networks. Only Teredo tunneling method can penetrate NAT. Therefore, we need a research to examine Teredo IPv6 tunneling network performance on certain application, mostly on popular application like FTP which can transfer file between networks. The research is comparing FTP server application performance on full cone NAT configured network with IPv6 Teredo tunneling toward pure full cone NAT IPv4 network and pure IPv6 network with the same application. Parameters to be compared are latency (s) and throughput (KBytes). The research done shows that Teredo tunneling performance on FTP application is lower than pure IPv4 full cone NAT network and pure IPv6 network, however, on real WAN simulated network, Teredo performance is only a little bit lower than pure IPv6 network, Teredo latency is higher between 7,4 - 28,08 % than pure IPv6 network and Teredo throughput is lower 2,89 - 16,55 % than pure IPv6 network, so it's suitable to provide IPv6 connectivity for nodes that is located behind IPv4 NAT in this transition period when most of the node have migrate to IPv6.

Keywords : IPv6, NAT,Teredo *tunneling*, FTP.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

DAFTAR SINGKATAN

FTP	File Transfer Protocol
GUI	Graphical User Interface
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISATAP	Intrasite Automatic Tunnel Access Protocol
LAN	Local Area Network
NAT	Network Address Translation
NAT-PT	Network Address Translation-Protocol Translation
NDP	Neighbor Discovery Protocol
OS	Operating System
PC	Personal Computer
QoS	Quality of Service
RA	Router Advertisement
RS	Router Solicitation
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UDPv4	User Datagram Protocol version 4
WAN	Wide Area Network

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Perkembangan Internet telah sampai pada titik yang tidak pernah dibayangkan sebelumnya. Perkembangan Internet sedemikian besar hingga di dunia ini terdapat kurang lebih 100 juta *host*, dan lebih dari 350 juta *user* yang aktif di Internet (melebihi jumlah penduduk di Indonesia), dan masih akan bertambah tiap tahunnya. Akibatnya, alamat IPv4 yang tersedia akan semakin berkurang. Secara teoritis, tersedia 4,294,967,296 atau 4 milyar alamat IPv4 yang unik (2^{32} alamat). Namun, pada kenyataannya alamat IP yang tersedia hanya sekitar 3,2 – 3.3 milyar alamat. Hal tersebut dikarenakan oleh penggunaan alamat yang dibagi-bagi ke dalam kelas-kelas tertentu, pengalokasian alamat yang digunakan untuk *multicasting*, dan lain-lain.

Untuk mengantisipasi habisnya alamat IPv4, dikembangkanlah IPv6. Panjang alamat pada IPv6 adalah 128 bit, sehingga jumlah alamat yang tersedia adalah 2^{128} ($3,4 \times 10^{38}$), jauh lebih banyak dari IPv4. Namun implementasi IPv6 secara menyeluruh akan terealisasi dalam waktu yang lama (secara bertahap, tidak secara langsung), karena penggunaan IPv6 memerlukan modifikasi keseluruhan infrastruktur Internet. Di sinilah teknologi transisi berperan. Teknologi transisi IPv4 ke IPv6 secara garis besar terbagi menjadi tiga jenis, yaitu *dual-stack*, *translation*, dan *tunneling*.

Metode transisi yang paling populer adalah *tunneling*, karena *tunneling* memungkinkan *coexistence* antara dua *cloud* IP (IPv4 dan IPv6). Namun, tidak semua *node* mendukung beberapa metode *tunneling*, contohnya adalah jaringan IPv4 dengan NAT. Hanya metode *tunneling* IPv6 Teredo yang dapat menembus NAT. Saat ini, belum ada penelitian khusus mengenai kinerja jaringan yang menggunakan *tunneling* IPv6 Teredo pada aplikasi-aplikasi tertentu, terutama aplikasi yang populer digunakan seperti FTP untuk transfer file antar jaringan.

1.2. TUJUAN

Tujuan dari skripsi ini adalah menyajikan unjuk kerja aplikasi FTP *server* pada jaringan NAT *full cone* dengan metode *tunneling* IPv6 Teredo, sebagai salah satu solusi metode transisi *coexistence* antara jaringan IPv4 dengan jaringan IPv6, serta bagaimana kinerjanya jika dibandingkan dengan aplikasi yang sama pada jaringan NAT *full cone* IPv4 murni dan pada jaringan IPv6 murni.

1.3. PEMBATASAN MASALAH

Pada skripsi ini akan dilakukan pengujian dan perbandingan kinerja aplikasi FTP *server* pada jaringan NAT *full cone* dengan metode *tunneling* IPv6 Teredo terhadap jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni. Pengujian dilakukan dengan menggunakan jaringan lokal berupa *test-bed* yang terdiri atas tujuh buah PC. Parameter yang diperbandingkan adalah *latency* dan *throughput*.

1.4. SISTEMATIKA PENULISAN

Untuk memberikan gambaran mengenai apa saja yang ditulis dalam skripsi ini, maka secara garis besar sistematika/cara penulisan pada skripsi ini terdiri dari lima bab sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini akan dijelaskan secara singkat mengenai latar belakang, tujuan, pembatasan masalah dan sistematika penulisan, dengan maksud memberikan gambaran tentang isi skripsi.

BAB II LANDASAN TEORI

Pada bab ini akan dijelaskan tentang teori-teori yang mendukung uji coba yang dilakukan. Bab ini akan menjelaskan tentang IPv6, mekanisme transisi dari IPv4 ke IPv6, *tunneling* IPv6 Teredo, NAT, dan FTP *server*.

BAB III KONFIGURASI & METODE PENGAMBILAN DATA

Pada bab ini akan dijelaskan topologi dan konfigurasi jaringan yang disusun untuk penelitian serta metode yang digunakan dalam implementasi *tunneling* IPv6 Teredo, dalam hal ini perangkat lunak yang digunakan.

BAB IV ANALISA

Pada bab ini akan dijelaskan mengenai pengolahan data yang dilakukan dan analisa hasil pengolahan data tersebut.

BAB V KESIMPULAN

Pada bab ini berisi kesimpulan-kesimpulan yang dapat ditarik dari uji coba yang telah dilakukan.

BAB II
LANDASAN TEORI

2.1. IPv6

IPv6 adalah standar internet masa depan yang telah ditetapkan oleh IETF (*Internet Engineering Task Force*). IPv6 dikembangkan untuk mengatasi/menyempurnakan kekurangan-kekurangan pada teknologi pendahulunya yaitu IPv4[1].

2.1.1. Penyempurnaan yang Dilakukan dari IPv4 ke IPv6

- Penambahan panjang bit alamat, pada IPv4 hanya ada 32 bit, pada IPv6 dikembangkan menjadi 128 bit. Jika dibandingkan pada IPv4, dari 32 bit hanya diperoleh 4.294.967.296 alamat, sedangkan pada IPv6, dari 128 bit bisa didapat 340,282,366,920,938,463,463,374,607,431,768,211,456 alamat atau sekitar 3.4×10^{38} alamat, maka kapasitas alamat yang tersedia pada IPv6 jauh lebih banyak daripada IPv4.
- Penyederhanaan format *header*, walaupun panjang alamat IPv6 lebih panjang dari IPv4, namun format *header* pada IPv6 lebih sederhana daripada *header* IPv4. Beberapa *field* pada IPv4 dihilangkan atau dibuat bersifat *optional*, tujuannya agar mengurangi beban *payload* pada saat pemrosesan paket dan mengurangi beban *bandwidth* dari *header* IPv6 itu sendiri.
- Penambahan dukungan yang lebih baik untuk *extensions* dan *options*, penyederhanaan *header* IP memungkinkan penambahan beberapa *option header* dan *option extension* baru, proses pengiriman yang lebih efektif, serta fleksibilitas untuk penambahan *option* di masa yang akan datang.
- Penambahan kemampuan *flow labeling*, dengan *flow labeling*, IPv6 dapat melakukan *labeling* terhadap paket-paket dari *traffic flow* tertentu yang membutuhkan penanganan khusus, seperti QoS yang berbeda atau layanan-layanan yang bersifat *realtime*.

- Penambahan fitur autentikasi dan *privacy*, pada IPv6 telah ditambahkan fitur autentikasi, dukungan integritas data, dan *data confidentially* (*optional*).

2.1.2. Pengalamatan IPv6

Pada IPv6, ada tiga tipe pengalamatan yang berbeda. Pengalamatan tersebut antara lain :

- *Unicast*, merupakan pengenal dari sebuah *interface* tunggal. Sebuah paket yang dikirim ke alamat *unicast* akan sampai pada *interface* yang diidentifikasi oleh alamat tersebut.
- *Anycast*, merupakan pengenal dari sekumpulan *interface*, biasanya dimiliki oleh *node* yang berbeda. Sebuah paket yang dikirim ke alamat *anycast* akan sampai pada *interface* yang diidentifikasi oleh alamat tersebut. *Interface* yang dituju adalah yang terdekat berdasarkan pengukuran jarak oleh *routing protocol*).
- *Multicast*, merupakan pengenal dari sekumpulan *interface*, biasanya dimiliki oleh *node* yang berbeda. Sebuah paket yang dikirim ke alamat *multicast* akan sampai pada semua *interface* yang diidentifikasi dari alamat tersebut.

Pengalamatan *broadcast* tidak lagi digunakan pada IPv6 karena fungsinya telah digantikan oleh *multicast*.

Semua tipe alamat IPv6 diberikan sebagai pengenal *interface*, bukan *node*. Sebuah alamat IPv6 *unicast* mengacu pada sebuah *interface*, karena tiap *interface* adalah milik sebuah *node*, maka alamat *unicast interface* tiap *node* tersebut dapat digunakan sebagai pengenal untuk *node*. Semua *interface* harus memiliki setidaknya satu *link-local unicast address*. Sebuah *interface* dapat pula memiliki beberapa alamat IPv6 dari barbagai tipe (*unicast*, *multicast*, dan *anycast*) atau *scope*.

IPv6 memiliki format penulisan alamat yang berbeda dengan pendahulunya (IPv4). Jika pada IPv4 ditulis dalam desimal yang terbagi menjadi empat bagian, maka pada IPv6 alamat ditulis dalam heksadesimal yang terbagi menjadi delapan bagian. Format penulisan alamat IPv6 adalah x:x:x:x:x:x:x:x, dimana x adalah empat *digit* bilangan heksadesimal. Contoh penulisan alamat IPv6 dapat dilihat sebagai berikut :

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

Untuk memudahkan penulisan alamat dengan angka nol dalam jumlah banyak, maka angka nol tersebut dapat diganti dengan "::". Contohnya adalah sebagai berikut :

- 1080:0:0:0:8:800:200C:417A (*unicast address*)
- FF01:0:0:0:0:0:0:101 (*multicast address*)
- 0:0:0:0:0:0:0:1 (*loopback address*)
- 0:0:0:0:0:0:0:0 (*unspecified addresses*)

Dapat disederhanakan menjadi :

- 1080::8:800:200C:417A (*unicast address*)
- FF01::101 (*multicast address*)
- ::1 (*loopback address*)
- :: (*unspecified addresses*)

Penulisan *prefix* pada alamat IPv6 sama seperti penulisan *prefix* pada alamat IPv4 yang ditulis dalam notasi CIDR. Penulisan *prefix* alamat IPv6 adalah alamat-IPv6/panjang-*prefix*, dimana alamat-IPv6 adalah notasi penulisan alamat pada IPv6 yang telah dijelaskan sebelumnya, dan panjang-*prefix* adalah bilangan desimal yang menyatakan berapa banyak jumlah bit yang diambil dari sebelah kiri untuk digunakan sebagai *prefix*. Contohnya adalah :

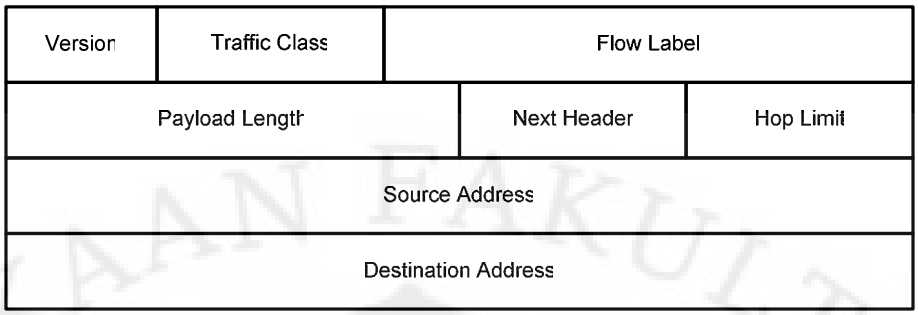
- 12AB:0000:0000:CD30:0000:0000:0000:0000/60
- 12AB::CD30:0:0:0:0/60
- 12AB:0:0:CD30::/60

Berikut ini adalah cara untuk menuliskan alamat *node* dan *prefix*-nya :

Alamat *node* : 12AB:0:0:CD30:123:4567:89AB:CDEF
Nomor *subnet* : 12AB:0:0:CD30::/60
Maka penulisannya menjadi : 12AB:0:0:CD30:123:4567:89AB:CDEF/60

2.1.3. Format *Header* IPv6

Seperti yang telah dijelaskan sebelumnya, IPv6 memiliki format *header* yang lebih sederhana jika dibandingkan dengan format *header* pada IPv4. Susunan format *header* pada IPv6 dapat dilihat pada Gambar 2.1.



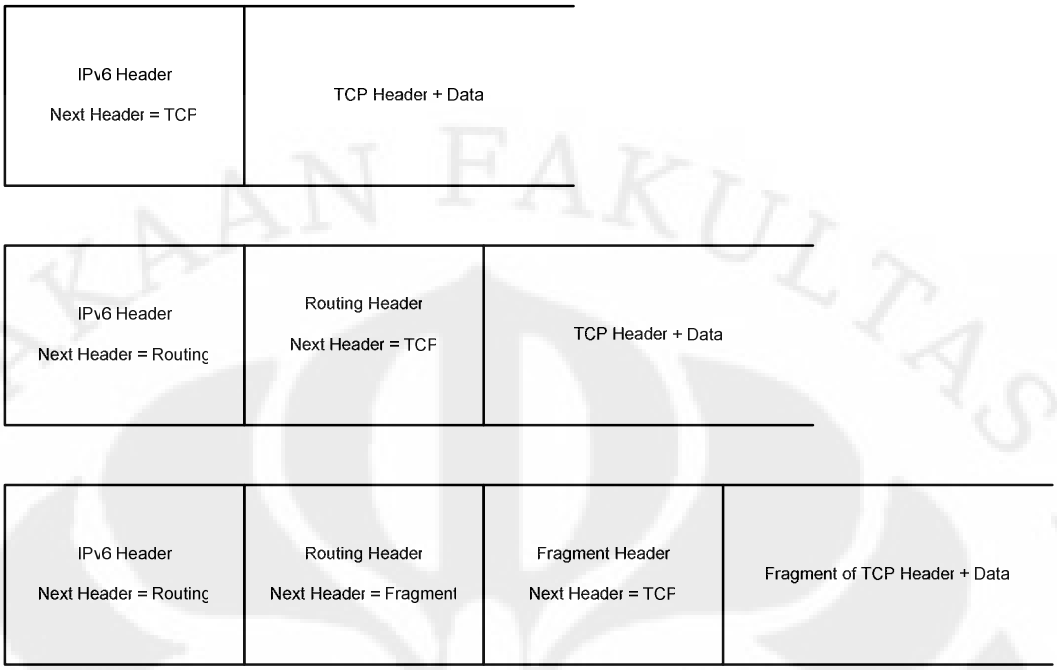
Gambar 2.1. Format *header IPv6*

Gambar 2.1 di atas menunjukkan komponen *header* IPv6 sebagai berikut :

- *Version*, merupakan empat bit bilangan yang menunjukkan nomor versi dari *Internet Protocol* adalah enam (IPv6).
- *Traffic Class*, delapan bit bilangan yang menunjukkan kelas *traffic*.
- *Flow Label*, dua puluh bit bilangan yang menunjukkan *flow label*.
- *Payload Length*, enam belas bit bilangan *integer* yang menunjukkan panjang paket IPv6 yang *header* dengan satuan oktet. Jika terdapat *extension header*, maka dianggap bagian dari *payload* termasuk panjangnya.
- *Next header*, delapan bit bilangan yang menunjukkan tipe *header* yang tepat mengikuti *header* IPv6 tersebut.
- *Hop limit*, delapan bit bilangan *integer* yang menunjukkan jumlah *node* yang dapat dilewati oleh paket. Nilainya berkurang satu setiap paket melewati satu *node*. Ketika nilainya mencapai nol, maka paket dibuang.
- *Source address*, alamat sepanjang 128 bit yang merupakan alamat sumber/asal dari paket.
- *Destination address*, alamat sepanjang 128 bit yang merupakan alamat tujuan dari paket.

2.1.4. IPv6 *Extension Header*

Pada IPv6, *extension header* adalah *header* tambahan yang menyimpan informasi *optional* pada IPv6. *Extension header* terletak di antara *header* IPv6 dan *header* pada lapisan di atasnya. Keberadaan *extension header* ini ditunjukkan oleh nilai *next header* pada *header* sebelumnya. Setiap paket IPv6 dapat memiliki satu atau lebih *extension header*, atau tidak sama sekali. Format *extension header* pada IPv6 dapat dilihat pada Gambar 2.2.



Gambar 2.2. IPv6 *extension header*

Extension header tidak diproses atau diperiksa pada tiap *node* di sepanjang jalur yang dilalui paket sebelum sampai pada *node* yang disebutkan dalam *field destination address*. *Extension header* perlu diproses secara urut sesuai dengan urutan kemunculannya pada sebuah paket. Sebuah *node* tidak boleh mencari *extension header* tertentu pada sebuah paket dan mengeksekusinya terlebih dahulu. *Hop-by-hop option header* menyimpan informasi yang harus diperiksa oleh tiap *node* yang dilewati paket tersebut, termasuk *node* sumber dan tujuan. Jika ada *hop-by-hop option header*, maka harus langsung mengikuti *header* IPv6.

Jika pada paket digunakan lebih dari satu *extension header*, maka disarankan *header* tersebut diurutkan sebagai berikut :

- *IPv6 header*
- *Hop-by-Hop Options header*
- *Destination Options header*
- *Routing header*
- *Fragment header*
- *Authentication header*
- *Encapsulating Security Payload header*

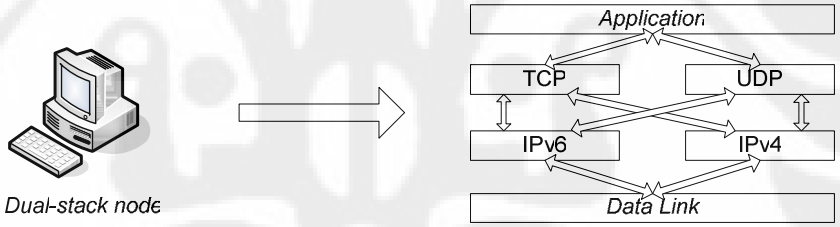
- *Destination Options header*
- *Upper-layer header*

2.2. MEKANISME TRANSISI IPV4 KE IPV6

Bermacam-macam teknologi telah tersedia untuk memudahkan proses transisi dari IPv4 ke IPv6. Secara garis besar, teknologi-teknologi tersebut dibagi menjadi tiga metode, yaitu *dual-stack*, *translation*, dan *tunneling*[2].

2.2.1. *Dual-Stack*

Metode transisi *dual-stack* adalah penggunaan dua *protocol stack* (IPv4 dan IPv6) sekaligus pada semua divais yang membutuhkan akses ke dua teknologi *layer* jaringan tersebut. Divais dengan *dual-stack* perlu dikonfigurasikan dengan alamat IPv4 dan IPv6. Divais tersebut harus dapat berkomunikasi dengan menggunakan IPv4 dan IPv6. Divais tersebut juga harus mampu menjalankan aplikasi untuk IPv4 dan IPv6 dengan urutan protokol yang berbeda. Arsitektur metode transisi *dual-stack* dapat dilihat pada Gambar 2.3.

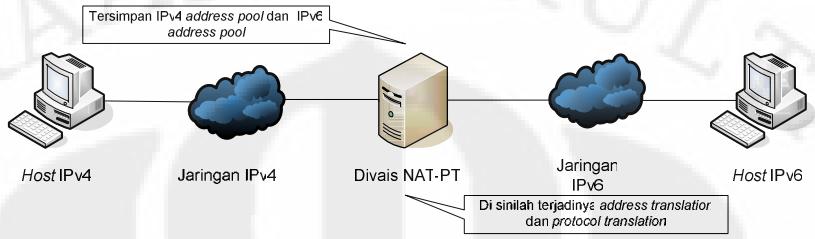


Gambar 2.3. Arsitektur *dual-stack*

2.2.2. *Translation*

Metode *translation* adalah metode transisi yang digunakan ketika diperlukan komunikasi antara *host* IPv4 dengan *host* IPv6. Sesuai dengan namanya, metode *translation* bekerja dengan menerjemahkan paket dari IPv6 ke IPv4, atau sebaliknya. Salah satu metode *translation* yang populer adalah NAT-PT. NAT-PT merupakan kombinasi dari *translation* pada *network address* dan protokol. NAT-PT bekerja dengan menerjemahkan alamat IPv4 menjadi alamat IPv6, mirip seperti NAT pada IPv4, namun NAT-PT juga menerjemahkan *protocol header*. Perbedaan NAT-PT dengan NAT pada IPv4 adalah NAT-PT tidak menerjemahkan alamat *private* dan global, namun menerjemahkan alamat IPv4 dan IPv6. Sebuah divais NAT-PT berperan sebagai *gateway* pada perbatasan

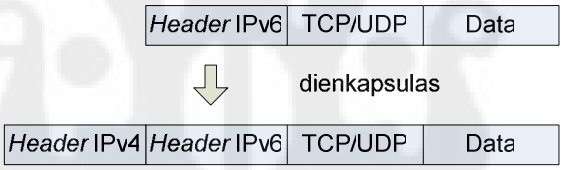
antara jaringan IPv4 dengan jaringan IPv6. Setiap divais NAT-PT menyimpan *IPv4 address pool* dan *IPv6 address pool* yang digunakan ketika komunikasi antar kedua *node* IPv4 dan IPv6 terjadi. Gambar 2.4 di bawah menunjukkan letak divais NAT-PT yang menjembatani dua jaringan IPv4 dan IPv6.



Gambar 2.4. NAT-PT sebagai *gateway*

2.2.3. *Tunneling*

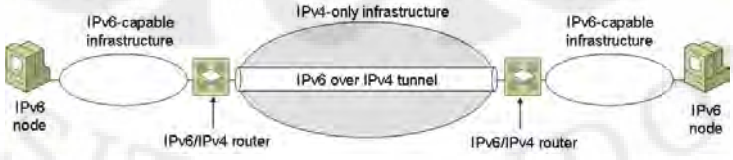
Tunneling adalah mekanisme transisi yang memungkinkan paket IPv6 dienkapsulasi ke dalam paket IPv4 agar dapat melewati infrastruktur jaringan IPv4 seolah-olah paket IPv6 tersebut melewati sebuah *tunnel* virtual. Gambar 2.5 di bawah menunjukkan bagaimana cara enkapsulasi paket pada metode *tunneling*.



Gambar 2.5. Enkapsulasi paket IPv6 ke dalam paket IPv4

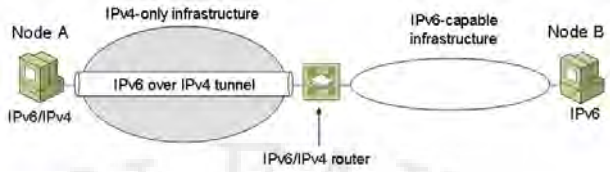
Berdasarkan tipe *endpoint*, enkapsulasi paket IPv6 ke dalam paket IPv4 dibedakan menjadi tiga tipe[3], yaitu dalam beberapa cara sebagai berikut :

- *Router-to-router*, dimana infrastruktur IPv4 menjadi *tunnel* perantara *router* IPv4/IPv6 yang terhubung, prosesnya ditunjukkan pada Gambar 2.6.



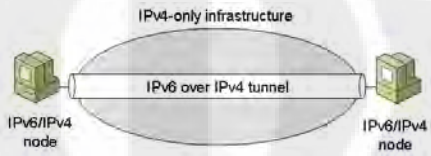
Gambar 2.6. *Router-to-Router Tunneling*

- *Host-to-router/router-to-host*, sebuah *host* IPv4/IPv6 ingin menjangkau *router* IPv4/IPv6 yang terpisah oleh *tunnel* pada infrastruktur IPv4, atau sebaliknya, prosesnya ditunjukkan Gambar 2.7.



Gambar 2.7. *Host-to-Router* dan *Router-to-Host Tunneling*

- *Host-to-host*, infrastruktur IPv4 langsung menjadi *tunnel* diantara *host* IPv4/IPv6 yang terhubung, prosesnya ditunjukkan Gambar 2.8.



Gambar 2.8. *Host-to-Host Tunneling*

Ada dua tipe *tunnel* yang dibedakan berdasarkan cara konfigurasinya, yaitu :

- *Configured tunnel*
- *Automatic tunnel*

Configured tunnel adalah *tunneling* yang membutuhkan konfigurasi manual pada kedua *tunnel endpoint*. Alamat *tunnel endpoint* yang dituju telah dikonfigurasi secara manual sebelumnya dengan *static route* yang digunakan pada *tunnel*. Ketika *tunnel endpoint* menerima paket IPv6, maka paket tersebut dienkapsulasi ke dalam paket IPv4 dan meneruskannya melewati infrastruktur IPv4 perantara menuju *tunnel endpoint* tujuan. Setelah paket enkapsulasi diterima, kemudian oleh *tunnel endpoint* tujuan paket tersebut dibuka *header*-nya dan diteruskan ke alamat IPv6 tujuan. *Configured tunnel* dipakai pada *router-to-router tunneling*.

Automatic tunnel adalah *tunneling* yang tidak membutuhkan konfigurasi manual. *Tunnel* dari *automatic tunnel* ditentukan oleh informasi yang tersimpan dalam paket IPv6, seperti alamat IP sumber atau tujuan. *Automatic tunnel* digunakan untuk *tunneling host-to-host* dan *host-to-router*. *Automatic tunnel* dipilih karena pada kedua *tunneling* tersebut paket IPv4/IPv6 langsung diteruskan ke *host* tujuan yang merupakan *tunnel endpoint*, sehingga perlu alamat IPv4 yang bisa didapat secara otomatis tanpa perlu mengkonfigurasi *tunnel* sebelumnya.

Automatic tunneling dibedakan menjadi empat tipe, yaitu tipe-tipe sebagai berikut :

2.2.3.1. Teredo

Metode *tunneling* Teredo adalah mekanisme *tunneling* yang didesain khusus untuk membuat *tunnel* bagi *node* IPv6 di atas jaringan IPv4 yang menggunakan topologi NAT (*Network Address Translation*). Informasi lebih detail mengenai Teredo dapat dilihat pada subbab 2.3.

2.2.3.2. ISATAP

Metode *tunneling Intra-Site Automatic Tunnel Addressing Protocol* (ISATAP) adalah mekanisme *tunneling* yang menghubungkan *host* atau *router* IPv6 melalui jaringan IPv4. ISATAP memandang jaringan IPv4 sebagai *link layer* dari untuk IPv6, dan melihat *node* lainnya pada jaringan sebagai *host* atau *router* IPv6 yang potensial.

2.2.3.3. 6to4

Metode *tunneling* 6to4 adalah mekanisme *tunneling* yang memungkinkan *node* IPv6 dapat berkomunikasi satu sama lainnya melalui jaringan IPv4 tanpa adanya *explicit tunnel setup*, dan *node* IPv6 tersebut dapat berkomunikasi dengan jaringan IPv6 murni melalui *relay router*. Jadi, 6to4 menggunakan *Wide Area Network* (WAN) IPv4 sebagai *unicast point-to-point link layer*. Mekanisme *tunneling* 6to4 ini hanya ditujukan sebagai alat bantu untuk memulai transisi pada permulaan masa/periode *coexistence* antara jaringan IPv4 dan IPv6, bukan sebagai solusi permanen.

2.2.3.4. 6over4

Metode *tunneling* 6over4 adalah mekanisme *tunneling* yang menggunakan IPv6 sebagai *local link address* yang kemudian dilewatkan pada jaringan *multicast* IPv4. Dengan 6over4, *host-host* IPv6 yang terisolasi pada infrastruktur yang tidak terhubung dengan *router* IPv6 dapat bekerja dengan menggunakan jaringan IPv4 yang mendukung *multicast* IPv4 sebagai "*virtual ethernet*".

2.3. TEREDO *TUNNELING*

Teredo merupakan protokol *tunneling* yang didesain untuk memberikan konektivitas IPv6 ke *node* yang berada di belakang divais NAT. Teredo bekerja dengan cara mengenkapsulasi paket IPv6 ke dalam *datagram* UDP IPv4 sehingga dapat menembus divais NAT yang berada pada internet IPv4.

Teredo hanya merupakan solusi transisi sementara dan opsi terakhir bila divais tidak mendukung metode *tunneling* lainnya seperti ISATAP, 6to4 maupun 6over4. Dalam jangka panjang, seluruh *host* IPv6 akan menggunakan koneksi jaringan IPv6 yang sebenarnya, perlahan-lahan Teredo akan ditinggalkan.

2.3.1. Fungsi-fungsi yang Dijalankan Teredo

- Mendiagnosa konektivitas UDP yang melalui IPv4 (UDPv4) serta mendeteksi tipe NAT yang digunakan.
- Memberikan alamat IPv6 unik yang *globally-routable* untuk digunakan oleh tiap *host*.
- Mengenkapsulasi paket IPv6 ke dalam datagram UDPv4 untuk dikirimkan melalui jaringan IPv4.
- Menjalurkan *traffic* antara *host* Teredo dengan *host native* IPv6.

2.3.2. Komponen-komponen *Node* Teredo

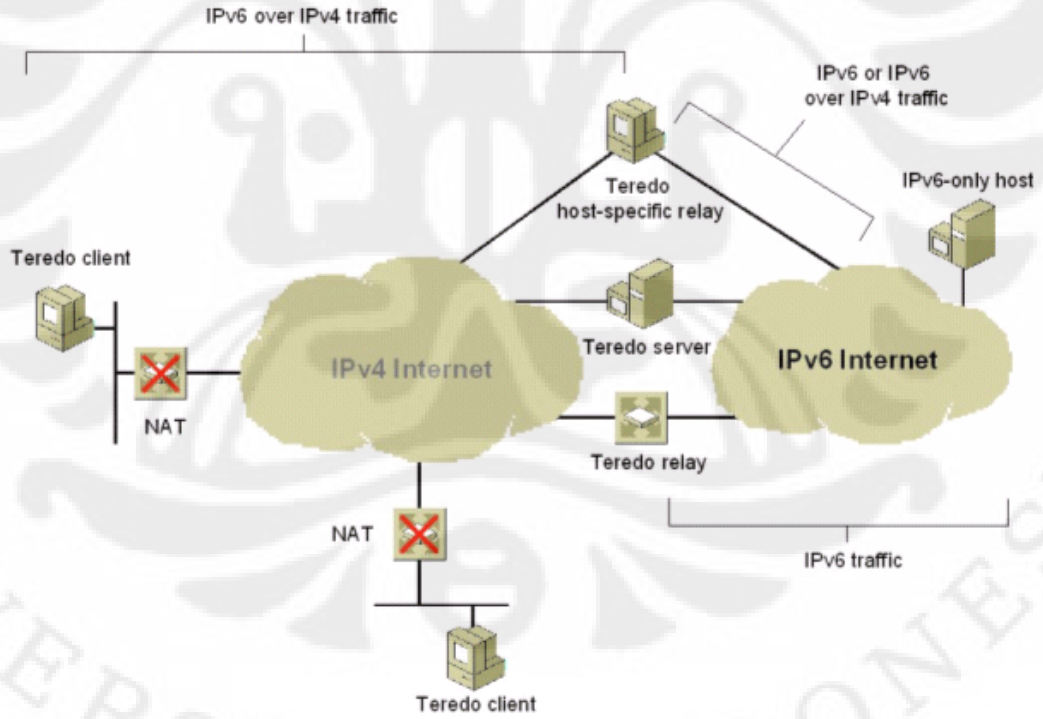
Teredo *client* adalah *host* yang memiliki koneksi IPv4 ke internet dari balik NAT dan menggunakan protokol *tunneling* IPv6 Teredo untuk mengakses Internet IPv6. Teredo *client* diberikan alamat IPv6 yang diawali dengan *prefix* Teredo (2001:0000::/32), sehingga secara tidak langsung Teredo *client* merupakan *node* yang menggunakan protokol *dual-stack*.

Teredo *server* adalah *host* yang digunakan untuk menginisialisasi konfigurasi awal untuk *tunnel* Teredo. Teredo *server* tidak pernah meneruskan paket-paket yang dikirimkan untuk *client* (kecuali dari ping IPv6 dan pada kasus NAT tipe *restricted*), dan sangat hemat akan kebutuhan *bandwidth* (maksimal hanya perlu beberapa ratus *Bytes* per menit per *client*), sehingga sebuah Teredo *server* dapat menampung *client* dalam jumlah besar.

Teredo *relay* bekerja melayani sebagai *remote end* pada *tunnel* Teredo. Teredo *relay* berfungsi meneruskan semua data/paket untuk kepentingan Teredo *client* yang dilayaninya, dengan pengecualian pada pertukaran langsung antar sesama Teredo *client*. Oleh sebab itu, *relay* membutuhkan *bandwidth* yang besar dan hanya dapat melayani *client* dalam jumlah terbatas pada waktu yang bersamaan. Setiap Teredo *relay* melayani sejumlah *host* IPv6 (contohnya seperti pada sebuah kampus/kantor), meneruskan *traffic* antara Teredo *client* dengan siapa saja *host* IPv6 yang berada dalam jangkauannya.

Teredo *host-spesific relay* adalah Teredo *relay* yang jangkauan layanannya terbatas hanya pada satu-satunya *host* Teredo *client* dimana ia terhubung dan dijalankan, sehingga tidak dibutuhkan *bandwidth* atau syarat-syarat *routing* khusus.

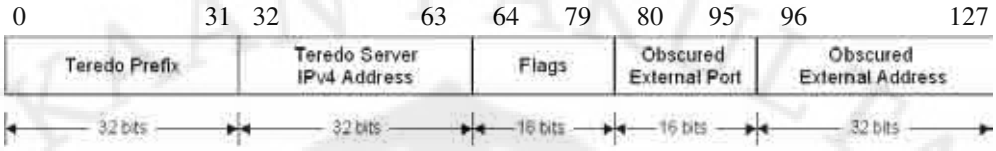
Letak komponen-komponen Teredo dapat dilihat pada Gambar 2.9 di bawah ini.



Gambar 2.9. Jaringan dengan *tunneling* Teredo

2.3.3. Pengalamatan Teredo IPv6

Format susunan alamat Teredo IPv6 pada Teredo *client* dapat dilihat pada Gambar 2.10 di bawah.



Gambar 2.10. Format susunan alamat Teredo IPv6 pada Teredo *client* (Teredo *address*)

Setiap Teredo *client* diberikan alamat IPv6 yang disusun sebagai berikut (bit orde yang lebih tertinggi dimulai dari angka 0) :

- Dari bit 0 sampai 31 diset sebagai Teredo *prefix* (biasanya 2001:0000::/32).
- Dari bit 32 sampai 63 diselipkan alamat IPv4 utama dari Teredo *server* yang digunakan dengan kondisi bit alamat tersebut diinversi.
- Dari bit 64 sampai 79 dapat digunakan untuk menetapkan sejumlah *flag*. Sementara ini hanya bit orde tertinggi yang digunakan. Bit tertinggi diset angka 1 jika Teredo *client* berada di balik NAT tipe *full cone*, angka 0 untuk sebaliknya (*restricted*).
- Dari bit 80 sampai 95 memuat nomor *port* UDP yang dipetakan oleh NAT pada Teredo *client* ketika bit-bit tersebut diinversi.
- Dari bit 96 sampai 127 memuat alamat IPv4 publik dari divais NAT dengan kondisi bit alamat tersebut diinversi.

Sebagai contoh, 2001:0000:4136:e378:8000:63bf:3fff:fdd2 merujuk pada Teredo *client* dengan konfigurasi berikut :

- Menggunakan Teredo *server* pada alamat 65.54.227.120 (didapat dari 4136:e378 pada bit 32-63 dijadikan biner lalu diinversi atau dari 4136:e378 dixor dengan ffffffff, lalu dijadikan alamat IPv4 dalam desimal).
- Ditempatkan di balik NAT *full cone*, didapat dari 8000 dalam heksadesimal diubah ke biner menjadi 1000000000000000, bit ke 64 diset angka 1.
- Menggunakan *port* UDP 40000 pada divais NAT, didapat dari 63bf dixor dengan ffff didapat 9c40 atau dalam desimal adalah 40000.

- NAT memiliki alamat IPv4 publik 192.0.2.45 (didapat dari 3fff:fdd2 pada 32 bit terakhir dijadikan biner lalu diinversi atau dari 3fff:fdd2 dixor dengan ffffffff, lalu dijadikan alamat IPv4 dalam desimal).

2.3.4. Teredo *Server*

Teredo *server* digunakan oleh Teredo *client* untuk mendapatkan Teredo *address* dan mendeteksi tipe NAT di balik jaringan di mana Teredo *client* berada (jika ada) secara otomatis. Teredo *client* juga mempertahankan ikatannya pada NAT melalui Teredo *server* yang terhubungnya, dengan mengirim paket UDP pada interval waktu yang tetap. Tujuannya untuk memastikan *server* tersebut dapat menghubungi tiap-tiap *client*, dimana dibutuhkan untuk *hole punching* agar Teredo dapat bekerja dengan benar pada kasus NAT tipe *restricted*.

Untuk kasus NAT tipe *restricted*, jika Teredo *relay* perlu mengirimkan paket IPv6 ke Teredo *client*, maka pertama-tama *relay* akan mengirimkan paket Teredo *bubble* ke Teredo *server* dari *client* tersebut, dimana alamat IP bisa diperoleh dari alamat IPv6 Teredo dari Teredo *client*. Lalu *server* dapat meneruskan *bubble* tersebut ke *client*, sehingga *software* Teredo *client* dapat mengetahui *hole punching* harus diselesaikan melewati Teredo *relay*.

Teredo *server* juga dapat mengirimkan paket ICMPv6 dari Teredo *client* melewati Internet IPv6. Pada prakteknya, pengiriman paket ICMPv6 digunakan ketika Teredo *client* ingin menghubungi *node* IPv6 *native*, maka *client* perlu mencari dimana Teredo *relay* yang bertanggung jawab atas *client* tersebut berada, maka *reply* dari paket ICMPv6 tersebut dapat menunjukkan Teredo *relay* mana yang digunakan.

Menggunakan sebuah Teredo *server* hanya membutuhkan jumlah *bandwidth* yang kecil karena Teredo *server* tidak terlibat dalam transmisi dan penerimaan paket IPv6 yang sebenarnya. Selain itu, *server* juga tidak terlibat pada akses ke *routing protocol* Internet mana pun. Kebutuhan mendasar untuk Teredo *server* antara lain :

- Kemampuan untuk memengirimkan paket ICMPv6 dengan alamat sumber berasal dari *prefix* Teredo.

- Dua alamat IPv4 publik terpisah (meskipun tidak tertulis pada spesifikasi resmi), alamat IPv4 kedua diperlukan pada proses pendeteksian tipe NAT yang digunakan, apakah *full cone*, *restricted*, atau *symmetric*.

2.3.5. Teredo *Relay*

Teredo *relay* berpotensi untuk membutuhkan *bandwidth* jaringan dalam jumlah besar. Selain itu, Teredo *relay* juga harus mampu mengekspor rute dari *prefix* IPv6 Teredo (2001:0::/32) menuju *host* IPv6 lainnya. Dengan demikian, Teredo *relay* akan menerima *traffic* dari pengalamatan *host* IPv6 ke Teredo *client* mana saja, dan meneruskannya melalui UDP/IPv4. Di antara *relay* dan *client* inilah proses enkapsulasi dan dekapsulasi terjadi.

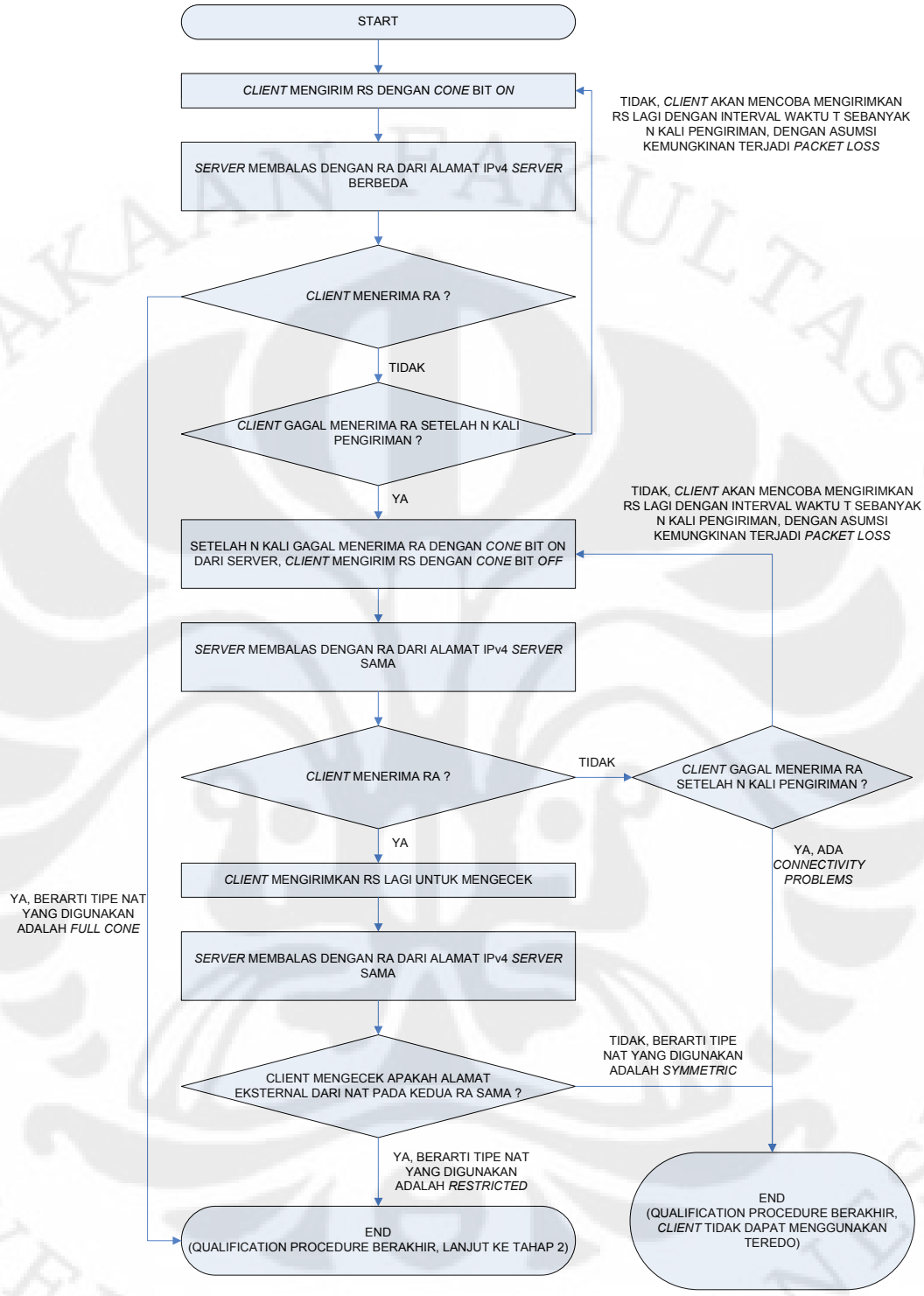
2.3.6. Cara Kerja Teredo

Secara garis besar proses kerja Teredo terbagi menjadi tiga tahap, yaitu *qualification procedure*, penentuan Teredo relay mana yang digunakan, pengiriman paket via Teredo relay.

2.3.6.1. Qualification procedure

Qualification procedure merupakan proses yang paling penting dalam Teredo, karena proses ini akan menentukan apakah *client* dapat menggunakan Teredo, apa tipe NAT yang digunakan, serta pemberian alamat IPv6 Teredo dari Teredo *server* ke Teredo *client*. Proses yang dilakukan adalah dengan menggunakan Proses yang dilakukan adalah dengan memanfaatkan *Neighbor Discovery Protocol* (NDP). Pertama-tama *client* mengirimkan *Router Solicitation* (RS) kepada alamat IPv4 publik dari *server*. Kemudian *server* membalas dengan mengirimkan *Router Advertisement* (RA) kepada *client*. Di dalam RA tersebut termuat data informasi alamat Teredo yang akan digunakan oleh Teredo *client* (Teredo *address*). Pengiriman RS yang pertama dilakukan oleh *client* adalah dengan *cone* bit diset *on* atau aktif, untuk mengetahui apakah *client* berada di balik NAT tipe *full cone* atau bukan. Dengan mengirimkan RS diset bit *cone* aktif, maka *server* akan memberikan balasan dengan mengirimkan RA dari alamat IPv4 publik yang berbeda dari alamat IPv4 publik yang dikirimkan RS oleh *client*. Jika *client* menerima RA, maka tahap pertama selesai dengan diketahuinya tipe NAT yang digunakan adalah *full cone* (karena menerima dapat paket dari luar dengan

bebas), berlanjut ke tahap kedua. Jika *client* tidak menerima RA, maka akan *client* akan menunggu pada interval waktu tertentu (secara *default* menunggu selama 4 detik) untuk mencoba mengirimkan RS lagi (secara *default client* mencoba lagi sebanyak 3 kali), karena ada kemungkinan paket RA hilang akibat *packet loss*. Jika *client* masih belum menerima RA, maka *client* dianggap pasti bukan berada di belakang NAT *full cone*, namun ada kemungkinan *client* berada di belakang NAT *restricted* atau *symmetric*. Kemudian *client* mencoba mengirimkan RS lagi, kali ini dengan bit *cone* diset *off*. Dengan bit *cone* diset *off* maka *server* akan mengirimkan RA dengan alamat IPv4 publik sumber pengirim Teredo *server* yang sama dengan alamat yang dikirimkan RS oleh *client*. Jika *client* masih tidak menerima RA, maka akan *client* akan menunggu pada interval waktu tertentu (secara *default* menunggu selama 4 detik) untuk mencoba mengirimkan RS lagi (secara *default client* mencoba lagi sebanyak 3 kali), karena ada kemungkinan paket RA hilang akibat *packet loss*. Jika *client* masih tidak menerima RA, maka dianggap ada masalah koneksi (*connectivity problems*), *client* menyerah dan tidak mengirimkan RS lagi. Jika *client* menerima RA, maka *client* akan melakukan pengecekan dengan mengirimkan RS dengan bit *cone* diset *off* ke alamat IPv4 publik *server* yang berbeda dari pengiriman RS pertama. Lalu setelah menerima RA kedua, *client* membandingkan RA pertama dengan yang kedua. Jika pada kedua RA terdapat alamat eksternal dari NAT yang sama, maka tahap pertama selesai dengan diketahuinya tipe NAT yang digunakan adalah *restricted* (karena hanya dapat menerima paket dari *host* eksternal di luar NAT yang sebelumnya pernah dikirimi paket), berlanjut ke tahap kedua. Jika ternyata alamat eksternal dari NAT yang didapat dari kedua RA berbeda, maka *client* dianggap tidak dapat menggunakan *service* (layanan) Teredo karena berada di belakang NAT tipe *symmetric*, jika *client* ingin menggunakan *service* (layanan) Teredo, maka perlu dilakukan konfigurasi manual untuk tiap *host* eksternal yang akan berkomunikasi dengan Teredo *client*. Alamat eksternal NAT diperoleh dari RA, karena RA memuat informasi Teredo *address*, dan di dalam Teredo *address* tersebut tersimpan alamat eksternal dari NAT yang digunakan. Alur kerja dari *qualification procedure* pada Teredo dapat dilihat pada Gambar 2.11 berikut.



Gambar 2.11. *Qualification procedure* pada Teredo

2.3.6.2. Penentuan Teredo relay mana yang digunakan

Penentuan Teredo *relay* dilakukan pada saat pertama kali *client* ingin berkomunikasi dengan *host* pada jaringan IPv6. Caranya adalah *client* mengirimkan paket ICMPv6 (ping6) ke *host* IPv6 yang ingin dihubungi, paket

ICMPv6 (dienkapsulasi ke dalam UDPv4) dikirimkan pertama-tama melalui Teredo *server*. Setelah sampai di *server*, paket tersebut didekapsulasi dari *header* UDPv4 dan diteruskan ke *cloud* IPv6 agar sampai pada *host* IPv6 tujuan. Setelah sampai, penerima paket kemudian membalas dengan memberikan *reply* yang dikirimkan melalui Teredo *relay* terdekat. *Relay* tersebut kemudian meneruskan *reply* dengan mengenkapsulasinya ke dalam paket UDPv4 agar dapat menembus NAT. Setelah *client* menerima *reply* tersebut, maka *client* dapat mengetahui *relay* terdekat mana yang dapat digunakan dari informasi yang terdapat dalam *reply* tersebut. Pada kasus NAT dengan tipe *full cone*, pengiriman paket *reply* berlangsung normal dari *relay* menembus NAT hingga sampai ke *client*. Pada kasus NAT dengan tipe *restricted*, pengiriman paket *reply* tidak dapat dilakukan dari *relay* secara langsung, karena *relay* sebelumnya belum pernah dikirimi paket oleh *client*, maka NAT tidak memperbolehkan *relay* mengirim paket ke *client*, sehingga *relay* mengirimkan paket melalui *server*. Proses inilah yang disebut dengan pengiriman *bubble packet*. Pertama-tama, *relay* mengirimkan paket *reply* (*bubble packet*) tersebut ke *server*. *Relay* dapat mengetahui alamat *server* dari alamat tujuan dari *bubble packet* tersebut (alamat tujuannya adalah alamat Teredo *client*), karena dalam alamat Teredo *client* tersimpan informasi alamat Teredo *server* yang digunakan oleh *client* tersebut. Kemudian oleh *server* paket tersebut diteruskan ke *client*. Setelah mendapatkannya, kini *client* mengetahui alamat dari *relay* dan dapat mengirimkan paket ke *relay*. Dengan demikian NAT melihat *relay* sebagai *host* atau *router* yang sudah pernah dikirimi paket oleh *client* dan memperbolehkan paket/*traffic* yang datang dari *relay*.

2.3.6.3. Pengiriman paket via Teredo relay

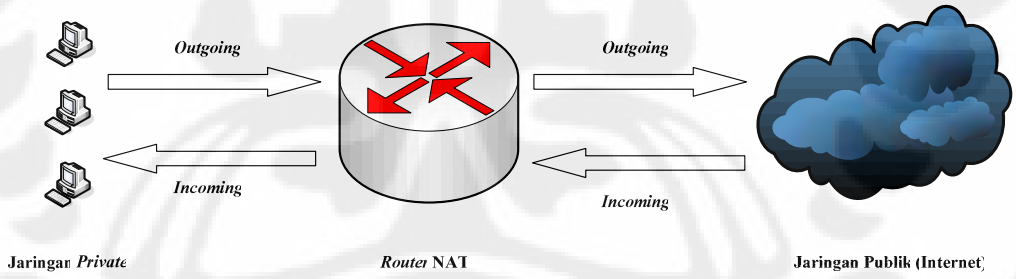
Pada tahap ini adalah tahap terakhir di mana proses pengiriman paket antara Teredo *client* dan *host* IPv6 di luar NAT berlangsung dengan melalui Teredo *relay*. Pengiriman berlangsung dengan adanya proses enkapsulasi dan dekapsulasi pada sisi *relay* dan *client*. Paket IPv6 dienkapsulasi ke dalam paket UDP IPv4 (UDPv4) agar dapat menembus NAT.

2.3.7. Keterbatasan Teredo

Tidak semua divais NAT didukung oleh Teredo. Hanya divais NAT tipe *full cone*, *restricted*, dan *port restricted* NAT saja yang didukung oleh Teredo, sementara divais tipe *symmetric* NAT tidak didukung secara *automatic* oleh Teredo, perlu konfigurasi manual untuk setiap kali Teredo *client* ingin terhubung ke luar jaringan NAT. *Bandwidth* yang tersedia untuk semua Teredo *client* menuju ke Internet IPv6 dibatasi oleh jumlah Teredo *relay* yang tersedia.

2.4. NAT (*NETWORK ADDRESS TRANSLATION*)

NAT memungkinkan alamat IP lokal/*private* pada jaringan *private* terhubung ke jaringan publik seperti Internet[4]. *Router* NAT ditempatkan di antara jaringan lokal dan jaringan publik berfungsi seperti jembatan yang memisahkan keduanya, NAT mentranslasikan alamat IP lokal/internal menjadi alamat IP global yang unik sebelum mengirimkan paket ke jaringan luar seperti Internet. Untuk menjalankan NAT, dibutuhkan sebuah divais (*firewall*, *computer*, atau *router*). Jaringan internal umumnya berupa LAN (*Local Area Network*), biasanya dirujuk sebagai *stub domain*. Mekanisme translasi pada NAT dapat dilihat pada Gambar 2.12.



Gambar 2.12. Mekanisme translasi pada *Router* NAT

Berdasarkan Cara Kerja Pengiriman Paket NAT digolongkan ke dalam tiga tipe, yaitu tipe-tipe berikut :

2.4.1. Full Cone NAT

Full Cone NAT adalah NAT dimana *entry* pada *translation table*-nya menyimpan pemetaan antara alamat internal dan eksternal beserta nomor *port*. Semua paket yang berasal dari *host* eksternal diterima dan diteruskan ke alamat tujuan paket pada jaringan internal.

2.4.2. Restricted Cone NAT

Restricted Cone NAT memiliki pemetaan seperti pada *Full Cone* NAT, namun tidak semua paket dari alamat eksternal diteruskan ke alamat internal, paket diteruskan jika berasal dari *host* pada jaringan eksternal yang sebelumnya pernah menerima paket dari *host* internal.

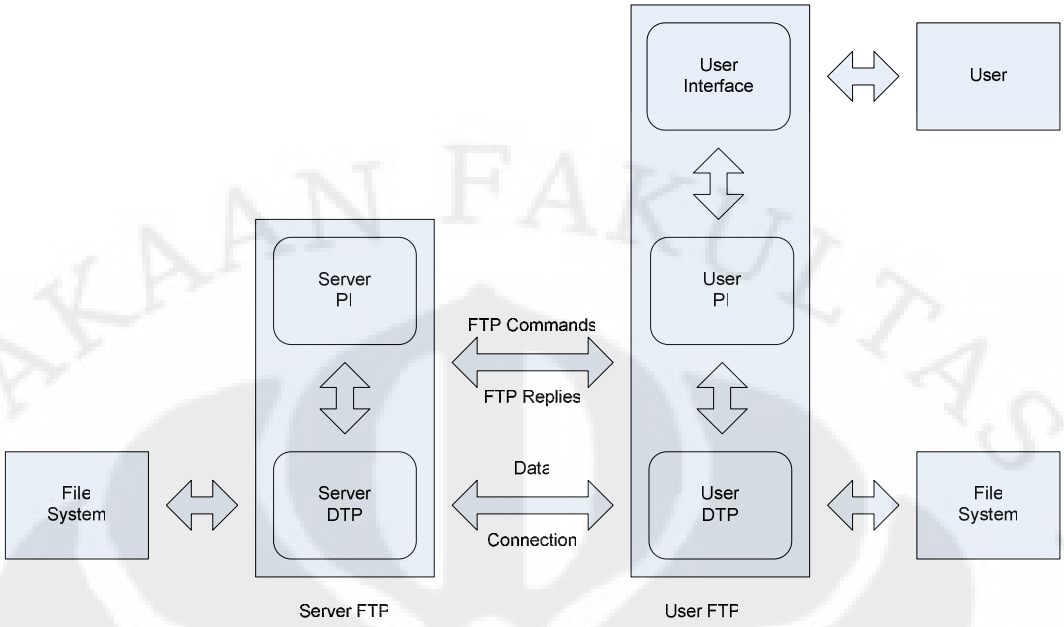
2.4.3. Symmetric NAT

Symmetric NAT adalah NAT yang memetakan alamat internal dan *port* yang sama ke alamat eksternal dan port yang berbeda-beda, tergantung dari alamat eksternal tujuan. Jika sebuah *host* mengirimkan beberapa paket ke alamat tujuan yang berbeda, digunakan pemetaan yang berbeda untuk tiap paket yang dikirimkan tersebut. Hanya *host* pada jaringan eksternal yang menerima paket tersebut yang boleh membalas mengirimkan paket kembali.

2.5. FTP (*FILE TRANSFER PROTOCOL*)

FTP (*File Transfer Protocol*) adalah protokol yang digunakan untuk mempertukarkan *file* antar komputer di dalam jaringan yang mendukung protokol TCP/IP, seperti Internet. Untuk memastikan bahwa file terkirim dan diterima tanpa terjadi *loss* pada *file* yang dipertukarkan, FTP menggunakan protokol TCP pada *transport layer*.

Seperti protokol yang bekerja pada model TCP/IP pada umumnya, FTP juga bekerja berdasarkan konsep *client/server*[5]. FTP *server* adalah *server* yang menyediakan layanan untuk pertukaran *file* ketika mendapatkan *request* dari FTP *client*. FTP *client* adalah *client* yang meminta koneksi ke FTP *server* untuk melakukan pertukaran *file*. Cara kerjanya, sebuah FTP *client* membuka koneksi ke FTP *server* untuk mengirimkan atau mengambil *file* dari FTP *server* tersebut. Setelah koneksi terbuka (FTP *client* terhubung dengan FTP *server*), maka *client* dapat melakukan manipulasi *file* seperti *upload file* ke *server*, *download file* dari *server*, *rename file* yang terdapat di *server*, maupun *delete file* yang terdapat di *server*, tergantung dari jenis *permission* yang diberikan *server* kepada *client*. Proses pertukaran *file* pada FTP dapat dilihat pada Gambar 2.13.



Gambar 2.13. Proses pertukaran file pada FTP

FTP bekerja seperti pada gambar di atas. *User* me-*request* koneksi FTP melalui *User Interface* yang dapat berupa *software* FTP *client*. Lalu *User Interface* melakukan hubungan ke *User* PI (*Protocol Interpreter*) yang kemudian melakukan hubungan ke *server* PI melalui *default port* untuk FTP, yaitu *port* 21. PI berperan memegang kendali dan meneruskan perintah/*command* FTP. PI juga mengendalikan DTP (*Data Transfer Process*). DTP menerima perintah/*command* transfer dari PI untuk mengirim atau menyimpan *file* ke medium penyimpanan.

Untuk menentukan siapa *client* yang berhak mengakses *server* serta apa saja hak akses *client*, digunakan sebuah sistem autentikasi untuk memastikan apakah *client* tersebut berhak mengakses *server* atau tidak. Autentikasi tersebut berupa permintaan *username* dan *password* dari FTP *client*.

Berdasarkan hak akses *client*-nya, FTP *server* dibedakan menjadi dua jenis, *anonymous* dan *restricted*. FTP *server* yang membatasi siapa *client* yang boleh mengaksesnya serta membatasi hak akses dari *client* disebut sebagai *restricted* FTP. FTP *server* yang membebaskan siapa saja untuk mengaksesnya, disebut sebagai *anonymous* FTP. *Anonymous* FTP memungkinkan semua orang untuk mengakses dan mengambil file secara bebas tanpa perlu memiliki akun pada FTP *server*. Pada sesi autentikasi *anonymous* FTP, *client* hanya perlu mengisi *anonymous* sebagai *username* dan alamat *email* sebagai *password*.

BAB III
KONFIGURASI & METODE PENGAMBILAN DATA

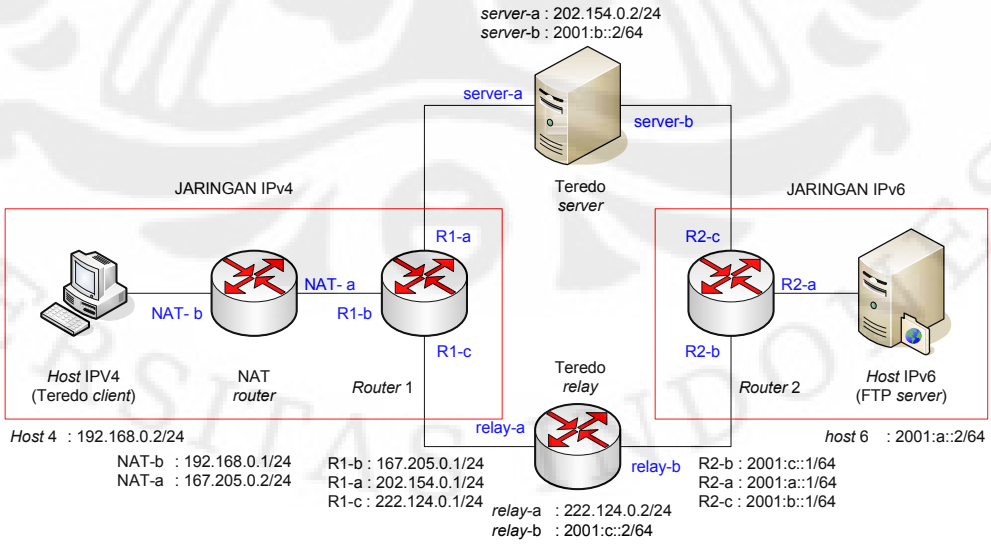
3.1. JARINGAN NAT *FULL CONE* DENGAN *TUNNELING* IPv6 TEREDO

3.1.1. Topologi Jaringan

Jaringan *test-bed* yang digunakan untuk melakukan implementasi *tunneling* Teredo merupakan sebuah jaringan lokal yang terdiri atas tujuh buah PC. Topologi ke tujuh PC tersebut antara lain :

- dua buah PC sebagai PC *router*
- satu buah PC sebagai *router* NAT *full cone*
- satu buah PC sebagai Teredo *server*
- satu buah PC sebagai Teredo *relay*
- satu buah PC sebagai *host* IPv6 (FTP *server*)
- dan satu buah PC sebagai *host* IPv4 (Teredo *client* sekaligus FTP *client*) di belakang NAT.

Topologi *test-bed* jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo secara umum dapat dilihat pada Gambar 3.1.



Gambar 3.1. Topologi jaringan *tunneling* IPv6 Teredo

3.1.2. Konfigurasi PC pada Jaringan

Pada Gambar 3.1, dapat dilihat bahwa dua jaringan IPv4 dan IPv6 dipisahkan oleh Teredo *relay* dan Teredo *server*. Perintah yang digunakan untuk konfigurasi Teredo dilampirkan pada Lampiran 1. Tiap-tiap PC pada topologi jaringan diposisikan dengan fungsi-fungsi berikut :

3.1.2.1. Host IPv6 (FTP server)

Host IPv6 di sini adalah sebuah *node* pada jaringan IPv6 yang diposisikan sebagai FTP *server* dalam jaringan IPv6 dengan fungsi melayani permintaan layanan FTP dari FTP *client*. FTP *server* dikonfigurasi sebagai *anonymous* FTP agar mudah diakses tanpa perlu autentikasi.

3.1.2.2. Router 2

Router 2 diposisikan sebagai *router* IPv6 yang memisahkan FTP *server* IPv6 ke Teredo *relay* dan Teredo *server*. *Router* 2 menjalurkan *traffic* antara FTP *server* (*host* 6) dengan Teredo *relay*.

3.1.2.3. Teredo Server

Teredo *server* bekerja pada saat inisialisasi awal jaringan dengan memberikan alamat IPv6 ke Teredo *client* (memberikan *prefix* Teredo). Selebihnya Teredo *server* hanya bekerja mempertahankan agar *tunnel* Teredo tetap eksis dengan menerima dan mengirimkan paket kepada Teredo *client* pada interval waktu tertentu (*maintaining* Teredo *tunnel*).

3.1.2.4. Teredo Relay

Teredo *relay* merupakan *router* yang menghubungkan jaringan IPv4 dengan IPv6, berfungsi menyalurkan *traffic* sehingga butuh *bandwidth* besar. Teredo *relay* menjalurkan *traffic* antara *router* 1 dan *router* 2.

3.1.2.5. Router 1

Router 1 diposisikan sebagai *router* IPv4 yang memisahkan jaringan NAT *host* IPv4 (Teredo *client* / FTP *client*) dengan Teredo *relay* dan Teredo *server*. *Router* 1 menjalurkan *traffic* antara *router* NAT dengan Teredo *server* dan Teredo *relay*. Pada *router* 1 ini dapat digunakan perangkat lunak *network emulator*,

fungsinya agar dapat mensimulasikan *cloud* jaringan yang sebenarnya. Tujuannya agar seolah-olah ada sebuah *cloud* jaringan yang berada di luar jaringan NAT.

3.1.2.6. Router NAT

Router NAT adalah PC *router* IPv4 yang difungsikan sebagai NAT *full cone* pada jaringan lokal tempat *host* IPv4 (Teredo *client* / FTP *client*) berada agar *host* IPv4 yang berada pada jaringan lokal tersebut dapat mengakses jaringan eksternal. Tipe NAT yang digunakan adalah *full cone* NAT.

3.1.2.7. Host IPv4 (Teredo client / FTP client)

Host IPv4 merupakan FTP *client* sekaligus Teredo *client*, karena *host* IPv4 mengakses FTP *server* melalui *tunneling* IPv6 Teredo.

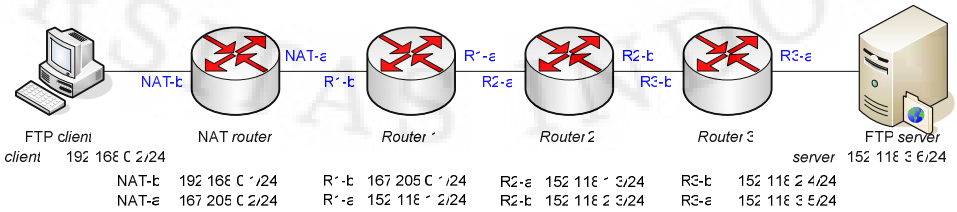
3.2. JARINGAN NAT *FULL CONE* IPv4 MURNI

3.2.1. Topologi Jaringan

Sebagai bahan perbandingan, perlu dirancang sebuah jaringan NAT IPv4 murni. Topologi dirancang dengan kondisi tertentu agar jumlah *node* yang dilalui paket sama seperti jumlah *node* yang dilalui paket pada jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo. Tujuannya agar hasil perbandingan yang dilakukan relevan. Jaringan *test-bed* yang digunakan untuk melakukan pengujian jaringan NAT *full cone* IPv4 murni merupakan sebuah jaringan lokal yang terdiri atas enam buah PC. Topologi keenam PC tersebut antara lain :

- tiga buah PC sebagai PC *router*
- satu buah PC sebagai *router* NAT *full cone*
- satu buah PC sebagai FTP *server*
- dan satu buah PC sebagai *host* IPv4 (Teredo / FTP *client*) di belakang NAT.

Topologi *test-bed* jaringan NAT IPv4 murni secara umum dapat dilihat pada Gambar 3.2.



Gambar 3.2. Topologi jaringan NAT IPv4 murni

3.2.2. Konfigurasi PC pada Jaringan

Pada Gambar 3.2, dapat dilihat bahwa NAT *full cone* IPv4 memisahkan FTP *client* dan FTP *server*. Perintah yang digunakan untuk konfigurasi NAT IPv4 dilampirkan pada Lampiran 2. Antara NAT dan FTP *server* dipisahkan oleh tiga buah *router*. Tiap-tiap PC pada topologi jaringan diposisikan dengan fungsi-fungsi berikut :

3.2.2.1. FTP server

FTP *server* di sini berfungsi melayani permintaan layanan FTP dari FTP *client*. FTP *server* dikonfigurasi sebagai *anonymous* FTP agar mudah diakses tanpa perlu autentikasi.

3.2.2.2. Router 1

Router 1 diposisikan sebagai salah satu dari tiga *router* IPv4 yang memisahkan jaringan NAT *full cone* IPv4 dengan FTP *server*. *Router* 1 juga berfungsi menjalurkan *traffic* antara *node-node* yang dipisahkannya. Pada *router* 1 ini dapat digunakan perangkat lunak *network emulator*, fungsinya agar dapat mensimulasikan *cloud* jaringan yang sebenarnya. Tujuannya agar seolah-olah ada sebuah *cloud* jaringan yang berada di luar jaringan NAT.

3.2.2.3. Router 2 dan router 3

Router 2 dan *router* 3 diposisikan sebagai *router* IPv4 yang memisahkan FTP *server* dengan *router* 1 serta berfungsi untuk menjalurkan *traffic* antara keduanya.

3.2.2.4. Router NAT

Router NAT adalah PC *router* yang difungsikan sebagai NAT *full cone* pada jaringan lokal tempat FTP *client* berada agar *host* pada jaringan lokal tersebut dapat mengakses jaringan eksternal. Tipe NAT yang digunakan adalah *full cone* NAT.

3.2.2.5. FTP client

FTP *client* adalah sebuah *host* pada jaringan NAT IPv4 yang berfungsi mengakses FTP *server* melalui jaringan NAT *full cone* IPv4 murni.

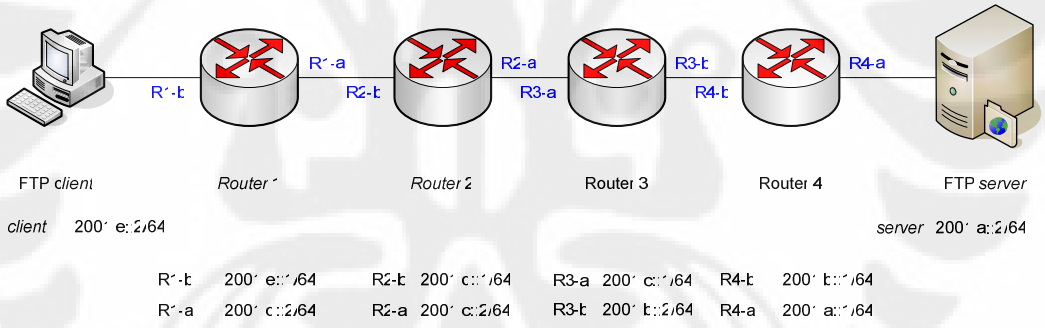
3.3. JARINGAN IPv6 MURNI

3.3.1. Topologi Jaringan

Sebagai bahan perbandingan tambahan, perlu dirancang sebuah jaringan IPv6 murni. Topologi dirancang dengan kondisi tertentu agar jumlah *node* yang dilalui paket sama seperti jumlah *node* yang dilalui paket pada jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo. Tujuannya agar hasil perbandingan yang dilakukan relevan. Jaringan *test-bed* yang digunakan untuk melakukan pengujian jaringan IPv6 murni merupakan sebuah jaringan lokal yang terdiri atas enam buah PC. Topologi keenam PC tersebut antara lain :

- empat buah PC sebagai PC *router*
- satu buah PC sebagai FTP *server*
- dan satu buah PC sebagai FTP *client*.

Topologi *test-bed* jaringan IPv6 murni secara umum dapat dilihat pada Gambar 3.3.



Gambar 3.3. Topologi jaringan IPv6 murni

3.3.2. Konfigurasi PC pada Jaringan

Pada Gambar 3.3, dapat dilihat bahwa antara FTP *client* dan FTP *server* dipisahkan oleh empat buah *router*. Perintah yang digunakan untuk konfigurasi IPv6 dilampirkan pada Lampiran 3. Tiap-tiap PC pada topologi jaringan diposisikan dengan fungsi-fungsi berikut :

3.3.2.1. FTP server

FTP *server* di sini berfungsi melayani permintaan layanan FTP dari FTP *client*. FTP *server* dikonfigurasi sebagai *anonymous* FTP agar mudah diakses tanpa perlu autentikasi.

3.3.2.2. Router 2

Router 2 diposisikan sebagai salah satu dari empat *router* IPv6 yang memisahkan FTP *client* dengan FTP *server*. *Router* 2 juga berfungsi menjalurkan *traffic* antara *node-node* yang dipisahkannya. Pada *router* 2 ini dapat digunakan perangkat lunak *network emulator*, fungsinya agar dapat mensimulasikan *cloud* jaringan yang sebenarnya. Tujuannya agar seolah-olah ada sebuah *cloud* jaringan yang berada di antara FTP *server* dan FTP *client*.

3.3.2.3. Router 1, router 3 dan router 4

Router 1, *router* 3 dan *router* 4 diposisikan sebagai *router-router* IPv6 yang memisahkan FTP *server* dengan FTP *client* serta berfungsi untuk menjalurkan *traffic* antara keduanya.

3.3.2.4. FTP client

FTP *client* adalah sebuah *host* pada jaringan IPv6 yang berfungsi mengakses FTP *server* melalui jaringan IPv6 murni.

3.4. PERANGKAT LUNAK YANG DIGUNAKAN

Perangkat lunak yang digunakan pada jaringan *test-bed* ini terbagi atas tujuh jenis perangkat lunak yang dibedakan berdasarkan fungsinya. Ketujuh jenis perangkat lunak tersebut antara lain adalah *operating system* (OS) yang digunakan, perangkat lunak untuk mengkonfigurasi NAT, perangkat lunak untuk mengkonfigurasi *tunneling* IPv6 Teredo, perangkat lunak untuk menjalankan FTP *server*, perangkat lunak untuk menjalankan FTP *client*, perangkat lunak untuk menganalisa paket, dan perangkat lunak *network emulator*.

Jenis perangkat lunak yang pertama adalah *operating system* yang digunakan. Sebagian besar PC pada jaringan *test-bed* diinstalasikan Ubuntu Linux 6.06 LTS sebagai *operating system*-nya, yaitu *server*, NAT dan *router*. Penggunaan OS Ubuntu Linux berdasarkan pertimbangan dukungan yang dimilikinya sangat baik terhadap IPv6. Selain itu *platform* Ubuntu Linux mendukung dan kompatibel terhadap perangkat lunak jenis lainnya yang digunakan pada jaringan *test-bed*. Semua jenis perangkat lunak lainnya yang tersedia pada Ubuntu antara lain perangkat lunak untuk mengkonfigurasi NAT,

perangkat lunak untuk mengkonfigurasi *tunneling* IPv6 Teredo, perangkat lunak FTP *server*, dan perangkat lunak *network emulator*. Pertimbangan lainnya adalah Ubuntu Linux merupakan distribusi Linux terpopuler untuk *router* maupun *server* pada kurun waktu 12 bulan terakhir menurut distrowatch.com, situs yang memantau perkembangan distribusi Linux. Sedangkan pada PC yang diposisikan sebagai FTP *client* digunakan sistem operasi Microsoft Windows XP Professional Service Pack 2. Pemilihan Windows XP sebagai FTP *client* atas dasar pertimbangan bahwa Windows XP adalah sistem operasi yang populer digunakan sebagai komputer *desktop*, 92,42 % pangsa pasar komputer *desktop* diisi oleh Windows menurut survey yang dilakukan marketshare.hitslink.com pada bulan November 2007.

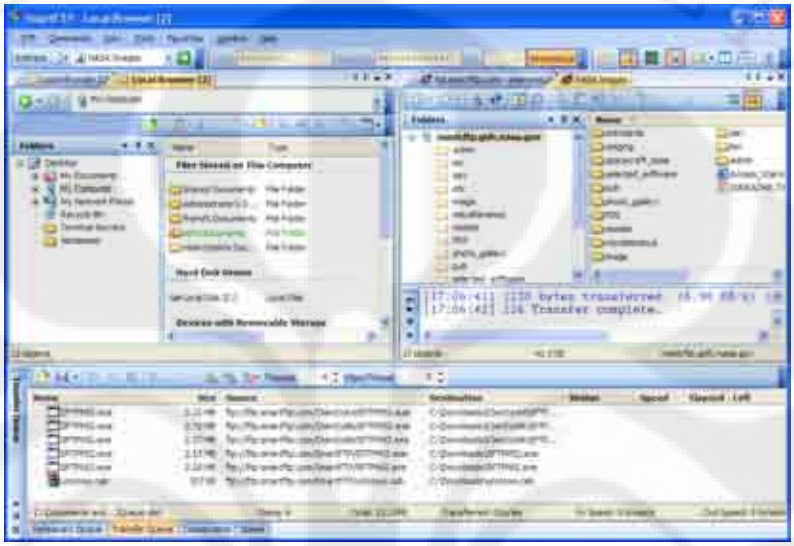
Perangkat lunak yang digunakan untuk mengkonfigurasi NAT adalah iptables. Iptables adalah perangkat lunak yang langsung tersedia paket instalasi Ubuntu Linux 6.06 LTS. Iptables dikonfigurasi berbasis perintah dalam teks.

Perangkat lunak yang digunakan untuk mengkonfigurasi *tunneling* IPv6 Teredo adalah Miredo. Miredo adalah perangkat lunak yang dikembangkan pada *platform* Linux sebagai perangkat lunak untuk mengkonfigurasi *tunneling* IPv6 Teredo. Miredo juga dikonfigurasi berbasis perintah-perintah di dalam teks.

Perangkat lunak yang digunakan untuk menjalankan FTP *server* adalah vsftpd. Vsftpd bekerja menjalankan FTP *server* pada *platform* Linux sebagai daemon (semacam *service*). Vsftpd dipilih karena dukungannya pada IPv6, sehingga vsftpd dapat dijalankan sebagai FTP server pada jaringan IPv6. Vstpd dikonfigurasi berbasis perintah dalam teks yang dilampirkan pada Lampiran 4.

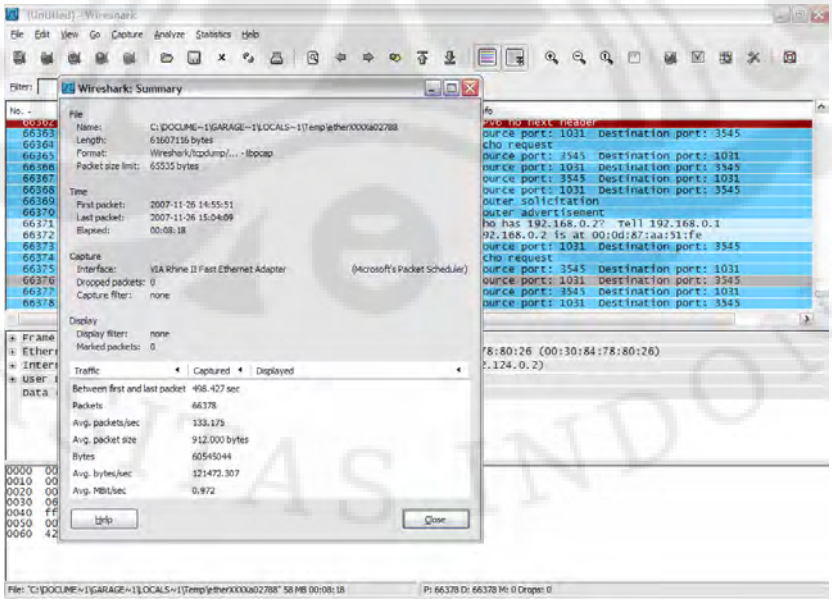
Perangkat lunak yang digunakan sebagai *network emulator* adalah netem. Netem adalah *tools emulator* jaringan yang berfungsi untuk mensimulasikan *cloud* jaringan pada satu *node* seolah-olah *node* tersebut adalah sebuah *cloud* jaringan WAN. Netem merupakan *tools* terintegrasi pada semua distribusi Linux yang menggunakan versi *kernel* terbaru yaitu *kernel* 2.6. Pada *router* yang diinstalasi *emulator* ini dikonfigurasi dengan *delay* sebesar 100ms. Pada *cloud* jaringan WAN, umumnya besar *delay* adalah 100ms[6]. Perintah yang digunakan untuk konfigurasi *netem* dilampirkan pada Lampiran 5.

Perangkat lunak yang digunakan untuk menjalankan FTP *client* adalah SmartFTP. SmartFTP adalah perangkat lunak FTP *client* yang dikembangkan untuk *platform* Windows seperti Windows XP. SmartFTP sudah berbasis GUI (*Graphical User Interface*) sehingga mudah digunakan. Antarmuka (*interface*) GUI dari SmartFTP dapat dilihat pada Gambar 3.4.



Gambar 3.4. GUI SmartFTP

Perangkat lunak yang digunakan untuk menganalisa paket adalah Wireshark. Wireshark merupakan software penganalisa paket berbasis GUI yang merupakan penerus Ethereal. GUI dari Wireshark dapat dilihat pada Gambar 3.5.



Gambar 3.5. GUI Wireshark

3.5. METODE PENGAMBILAN DATA

Pengambilan data ditujukan untuk menguji apakah Teredo dapat bekerja sebagaimana mestinya. Juga untuk melihat kinerja dari jaringan dengan menggunakan *tunneling* IPv6 Teredo. Selain itu ditujukan pula untuk membandingkan bagaimana kinerja jaringan dengan *tunneling* IPv6 Teredo terhadap jaringan IPv4 murni dan IPv6 murni tanpa *tunneling* Teredo.

Pengambilan data dilakukan dengan mengirimkan/mengambil paket-paket *file* FTP antara FTP *server* dan FTP *client*. Tipe *file* yang dikirimkan terdiri atas lima jenis tipe. Tipe-tipe file tersebut antara lain adalah:

- file1.txt sebesar 132 KB
- file2.mp3 sebesar 5,89 MB
- file3.zip sebesar 488 KB
- file4.pdf sebesar 1,88 MB
- file5.iso sebesar 51,2 MB

FTP *server* melakukan transfer *file* dengan dua jenis mode yaitu ASCII dan *binary*. Mode ASCII adalah untuk tipe *file* teks dan mode *binary* adalah untuk tipe *file* lainnya. Untuk mewakili mode transfer ASCII digunakan file1.txt sebesar 132 KB. Sedangkan untuk mewakili mode transfer binary digunakan keempat tipe *file* sisanya. Tipe file zip dipilih karena banyak FTP *server* mengompres ukuran filenya untuk menghemat kapasitas penyimpanan dan format zip adalah salah satu format kompresi yang populer digunakan[7]. Tipe file mp3 dipilih karena merupakan tipe file yang paling banyak disimpan dan didownload pada FTP *server*[8]. Dua tipe *file* lainnya yaitu pdf dan iso merupakan tipe file banyak terdapat pada FTP *server* yang berada di lingkungan Universitas Indonesia yaitu ftp://kambing.vlsm.org. Proses transfer untuk masing-masing file dilakukan sebanyak 10 kali pada masing-masing topologi yang digunakan. Untuk tiap-tiap topologi dilakukan pengambilan data pada dua kondisi, pertama tanpa simulasi jaringan yang sebenarnya, kedua dengan simulasi jaringan yang sebenarnya. Secara keseluruhan dilakukan 300 kali pengambilan data.

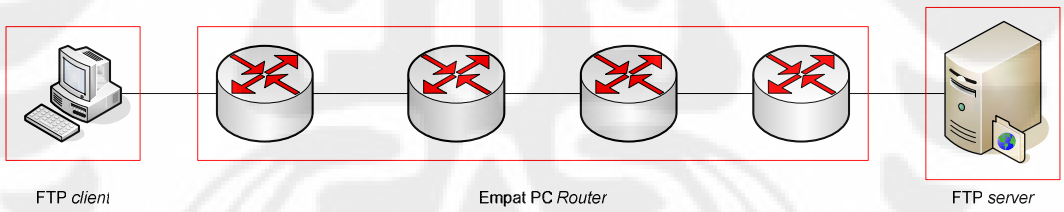
BAB IV
ANALISA

4.1. ANALISA JARINGAN *TEST-BED*

Pada pengujian dan pengambilan data untuk skripsi digunakan jaringan *test-bed* pada laboratorium Digital, pengujian yang dilakukan pada jaringan *test-bed* merupakan pengujian pada jaringan lokal, bukan pada jaringan sesungguhnya. Jaringan *test-bed* dikonfigurasi agar dapat digunakan untuk mengimplementasikan jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo, jaringan NAT *full cone* IPv4 murni, serta jaringan IPv6 murni.

Pada jaringan *test-bed*, jaringan terdiri atas tujuh buah PC yang digunakan sebagai *host* atau *client*, *server*, dan *router*. Konfigurasi tiap-tiap *node* pada jaringan lebih lengkap dapat dilihat pada bab sebelumnya. Untuk menghubungkan tiap-tiap *node* pada jaringan yang berupa PC digunakan Ethernet *card* berkecepatan 10/100Mbps. Pada jaringan *test-bed* tidak digunakan *dedicated router*, melainkan PC yang dikonfigurasikan sebagai *router* (PC *router*). Semua PC *router* yang digunakan dikonfigurasi dengan menggunakan sistem operasi Ubuntu Linux 6.06 LTS (*Long Term Support*). Pada sisi *server*, juga digunakan PC yang dikonfigurasi dengn OS Ubuntu Linux 6.06. Konfigurasi *router* dan *server* dengan menggunakan sistem operasi Ubuntu Linux 6.06 merupakan pilihan yang tepat, sebab sistem operasi tersebut terbukti lebih unggul digunakan sebagai sistem operasi pada *router* dan *server* daripada sistem operasi berbasis Windows. Karena dengan Ubuntu Linux, sistem hanya perlu dikonfigurasi sekali saja agar dapat bekerja sebagai *router* maupun *server*, meskipun PC telah *restart*. Tidak seperti sistem operasi Windows yang merepotkan, sebab *router* perlu dikonfigurasi setiap kali PC *restart*[9]. Sedangkan pada sisi *client* digunakan sistem operasi Microsoft Windows XP Professional Service Pack 2. Spesifikasi lebih lengkap masing-masing PC yang digunakan pada jaringan *test-bed* ini dilampirkan pada Lampiran 6.

Topologi yang diterapkan pada jaringan *test-bed* adalah topologi *bus*. Meskipun digunakan tujuh buah PC pada jaringan *test-bed* ini, namun sebenarnya hanya enam *node* yang dilalui oleh paket-paket FTP, dapat dilihat pada Gambar 4.1. Sebenarnya hanya jaringan NAT dengan *tunneling* IPv6 Teredo yang menggunakan ketujuh PC tersebut. PC yang tidak dilalui paket FTP adalah PC yang dikonfigurasi sebagai Teredo *server*, sebab Teredo *server* memang tidak berperan pada proses pengiriman data di *tunneling* IPv6 Teredo. Teredo *server* hanya berperan pada saat inisialisasi jaringan untuk memberikan alamat IPv6 (Teredo *prefix*) kepada Teredo *client* (dalam hal ini FTP *client*). Setelah itu Teredo *server* hanya bekerja untuk *maintaining* alamat IPv6 Teredo *client* dengan cara mengirimkan paket-paketnya pada interval waktu tertentu. Selebihnya Teredo *server* tidak berperan pada pengiriman paket-paket FTP. Pada dasarnya, *node* yang dilalui oleh paket-paket FTP pada ketiga jenis jaringan adalah enam buah PC, dua buah *node* FTP (FTP *server* dan FTP *client*) serta empat buah PC *router* yang berada di antara FTP *server* dan FTP *client*. PC *router* yang dilalui paket-paket FTP tersebut dapat berupa PC *router* biasa, NAT *router*, maupun Teredo *relay* yang bekerja sebagai *router*.



Gambar 4.1. *Node-node* yang dilalui paket FTP

Sebenarnya, jaringan *test-bed* yang digunakan merupakan jaringan lokal yang tidak terhubung ke jaringan di dunia luar, sehingga tidak dapat menggambarkan *cloud* jaringan WAN yang sesungguhnya. Oleh sebab itu, jaringan *test-bed* dikonfigurasi sedemikian rupa dengan menggunakan *network emulator* (WAN *emulator*) agar dapat menggambarkan jaringan WAN yang sebenarnya. *Network emulator* yang digunakan adalah *tools* netem (*network emulator*) yang merupakan bagian dari *tools* tc (*traffic controller*) yang ada pada sistem operasi Linux dengan kernel 2.6 ke atas. *Tools* netem dan tc dapat mengendalikan *traffic* pada jaringan dengan membatasi *bandwidth* dan *delay*,

mengatur *packet loss*, *packet duplication*, *packet re-ordering*, *packet corruption*, dan mengendalikan *rate control*. WAN *emulator* dapat mewakili *cloud* WAN sebenarnya. Kekurangan jaringan *test-bed* lainnya adalah penggunaan *static routing*, *static routing* menyebabkan pengiriman data menjadi lebih cepat jika dibandingkan dengan *dynamic routing* yang umumnya digunakan pada *cloud* WAN yang sebenarnya.

4.2. HASIL PENGOLAHAN DATA

Setelah dilakukan uji coba, maka dilakukan pengambilan data untuk kemudian diolah dan dianalisa. Data yang diambil adalah parameter perbandingan dari paket yang kemudian akan dianalisa. Paket yang dikirimkan sendiri merupakan *file-file* FTP yang ditransfer melalui jaringan *test-bed*. Penjelasan lebih lengkap mengenai tipe *file* yang dikirimkan dengan menggunakan FTP, mode pengiriman yang dilakukan, serta berapa kali pengambilan data dilakukan dapat dilihat pada bab sebelumnya. Sedangkan parameter yang dibandingkan adalah *latency* dalam satuan waktu (detik atau s) dan *throughput* dalam satuan KiloBytes per detik (KBytes/s). Parameter tersebut dipilih karena merupakan parameter yang menggambarkan kualitas seberapa cepat koneksi yang digunakan.

Pengolahan data yang dilakukan adalah mencari rata-rata dari data yang diambil pada masing-masing jenis jaringan untuk kemudian dibandingkan. Data yang diambil dilampirkan pada Lampiran 7. Hasil pengolahan data dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1. Hasil pengolahan data pada jaringan tanpa WAN *emulator*

file type	parameter	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt	latency (s)	0.7453	0.6896	0.81
132 KB	throughput (KBytes/s)	198.6331387	209.8010983	188.991447
file2.mp3	latency (s)	1.469	1.6199	6.7403
5,89 MB	throughput (KBytes/s)	4299.033289	3985.205563	1020.890208
file3.zip	latency (s)	0.7337	0.7882	1.1831
488 KB	throughput (KBytes/s)	721.0006373	665.1834899	474.144377
file4.pdf	latency (s)	1.1964	0.9916	2.9633
1,88 MB	throughput (KBytes/s)	1816.100076	2092.671361	765.3728003
file5.iso	latency (s)	14.5626	11.3241	45.7888
51,2 MB	throughput (KBytes/s)	2679.804414	4528.229506	1304.741782

Tabel 4.2. Hasil pengolahan data pada jaringan dengan WAN *emulator*

file type	parameter	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt	latency (s)	2.3273	2.564	3.2839
132 KB	throughput (KBytes/s)	62.64841768	56.26193711	46.95205967
file2.mp3	latency (s)	15.417	45.7554	49.7797
5,89 MB	throughput (KBytes/s)	382.3786155	140.9381837	136.8681565
file3.zip	latency (s)	2.7975	5.189	6.501
488 KB	throughput (KBytes/s)	173.9127059	101.3034279	87.42449453
file4.pdf	latency (s)	7.208	15.6646	16.8238
1,88 MB	throughput (KBytes/s)	270.6327496	131.3861477	129.5471651
file5.iso	latency (s)	112.8701	381.4127	423.6435
51,2 MB	throughput (KBytes/s)	450.2235621	146.7778788	140.131319

4.3. ANALISA HASIL PENGOLAHAN DATA

Analisa hasil pengolahan data dibagi menjadi lima bagian. Bagian pertama adalah analisa hasil pengolahan data pada jaringan dengan topologi jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo. Bagian kedua adalah analisa hasil pengolahan data pada jaringan dengan topologi jaringan NAT *full cone* IPv4 murni. Bagian ketiga adalah analisa hasil pengolahan data pada jaringan dengan topologi jaringan IPv6 murni. Bagian keempat adalah analisa perbandingan hasil pengolahan data pada jaringan *test-bed* yang dikonfigurasi tanpa menggunakan WAN *emulator*. Bagian kelima adalah analisa perbandingan hasil pengolahan data pada jaringan *test-bed* yang dikonfigurasi dengan menggunakan WAN *emulator*.

4.3.1. Jaringan NAT *full cone* dengan *Tunneling* IPv6 Teredo

Analisa hasil pengolahan data pertama adalah pada jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo. Analisa dilakukan dan dipisahkan berdasarkan dua parameter yaitu *latency* (s) dan *throughput* (KBytes/s). Kemudian masing-masing parameter dipisahkan lagi berdasarkan kondisi jaringan, yaitu dengan simulasi WAN atau tanpa simulasi WAN. Nilai rata-rata dari kedua parameter yaitu *latency* (s) dan *throughput* (KBytes/s) dapat dilihat pada Tabel 4.3.

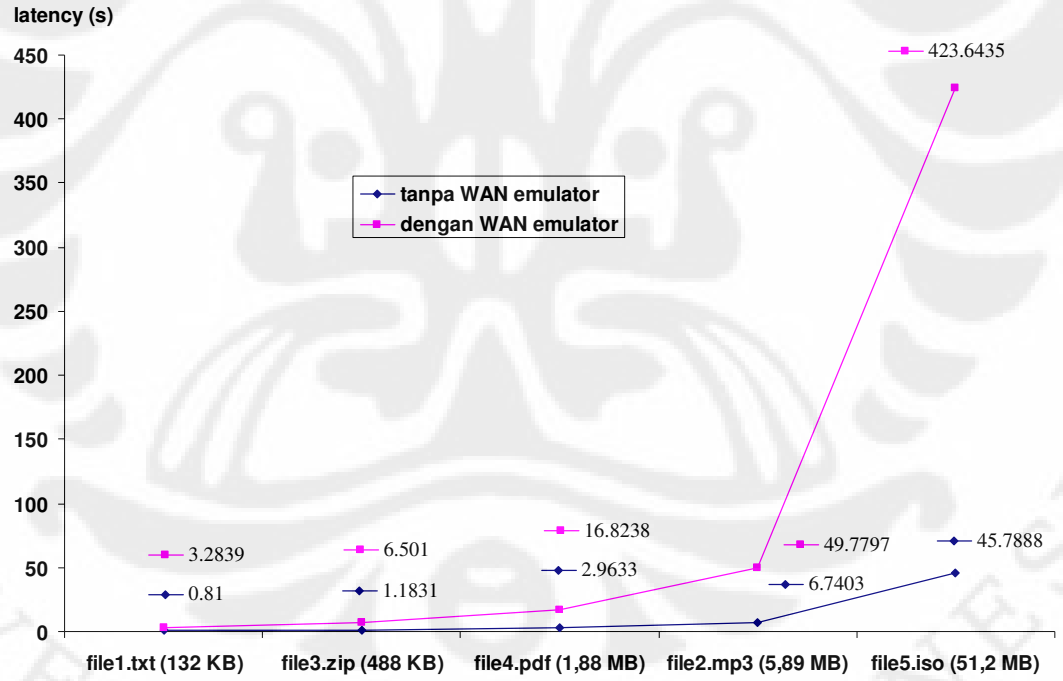
Tabel 4.3. Rata-rata *latency* dan *throughput* Teredo

filetype	latency (s)		throughput (KBytes/s)	
	tanpa WAN emulator	dengan WAN emulator	tanpa WAN emulator	dengan WAN emulator
file1.txt (132 KB)	0.81	3.2839	188.991447	46.95205967
file3.zip (488 KB)	1.1831	6.501	474.144377	87.42449453
file4.pdf (1,88 MB)	2.9633	16.8238	765.3728003	129.5471651
file2.mp3 (5,89 MB)	6.7403	49.7797	1020.890208	136.8681565
file5.iso (51,2 MB)	45.7888	423.6435	1304.741782	140.131319

4.3.1.1. Latency pada jaringan NAT full cone dengan tunneling IPv6 Teredo

Parameter pertama adalah *latency*, nilai yang disajikan didapat dari rata-rata pengolahan data *latency* jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo pada pengiriman lima tipe *file* melalui FTP. Rata-rata *latency* jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo dapat dilihat pada Tabel 4.3.

Untuk memudahkan dalam menganalisa rata-rata *latency* pada Tabel 4.3, dapat dilihat grafiknya pada Gambar 4.2.



Gambar 4.2. Grafik *latency* pada topologi jaringan Teredo

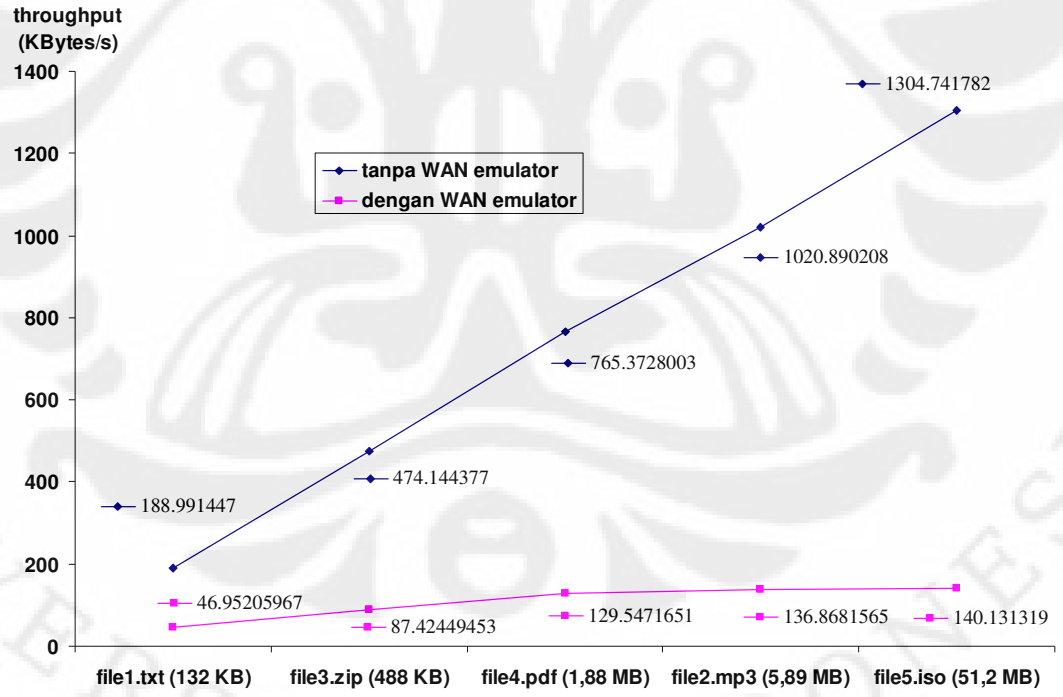
Pada Gambar 4.2 dapat dilihat grafik *latency* dari jaringan NAT dengan *tunneling* IPv6 Teredo. Grafik *latency* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan

FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Pada grafik jelas terlihat sebuah kecenderungan kelinearan, besar *latency* pada masing-masing *file* berbanding lurus dengan ukuran *file*. Maksudnya semakin besar ukuran *file,* maka semakin lama pula waktu yang dibutuhkan untuk mengirimkan *file* tersebut ke tujuannya (semakin besar *latency*). Kelinearan tersebut berlaku pada kedua kondisi, tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah).

4.3.1.2. Throughput pada jaringan NAT full cone dengan tunneling IPv6 Teredo

Parameter kedua adalah *throughput*, nilai yang disajikan didapat dari rata-rata pengolahan data *throughput* jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo pada pengiriman lima tipe *file* melalui FTP. Rata-rata *throughput* jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo dapat dilihat pada Tabel 4.3.

Untuk memudahkan dalam menganalisa rata-rata *throughput* pada Tabel 4.3, dapat dilihat grafiknya pada Gambar 4.3.



Gambar 4.3. Grafik *throughput* pada topologi jaringan Teredo

Pada Gambar 4.3 dapat dilihat grafik *throughput* dari jaringan NAT *full cone* dengan *tunneling* IPv6 Teredo. Grafik *throughput* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi WAN (garis biru) dan dengan simulasi WAN

(garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Seperti halnya yang terjadi pada grafik *latency*, pada grafik *throughput* juga terlihat kelinearan pada kedua kondisi, tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah).

4.3.2. Jaringan NAT *full cone* IPv4 Murni

Analisa hasil pengolahan data kedua adalah pada jaringan NAT *full cone* IPv4 murni. Analisa dilakukan dan dipisahkan berdasarkan dua parameter yaitu *latency* (s) dan *throughput* (KBytes/s). Kemudian masing-masing parameter dipisahkan lagi berdasarkan kondisi jaringan, yaitu dengan simulasi WAN atau tanpa simulasi WAN. Nilai rata-rata dari kedua parameter yaitu *latency* (s) dan *throughput* (KBytes/s) dapat dilihat pada Tabel 4.4.

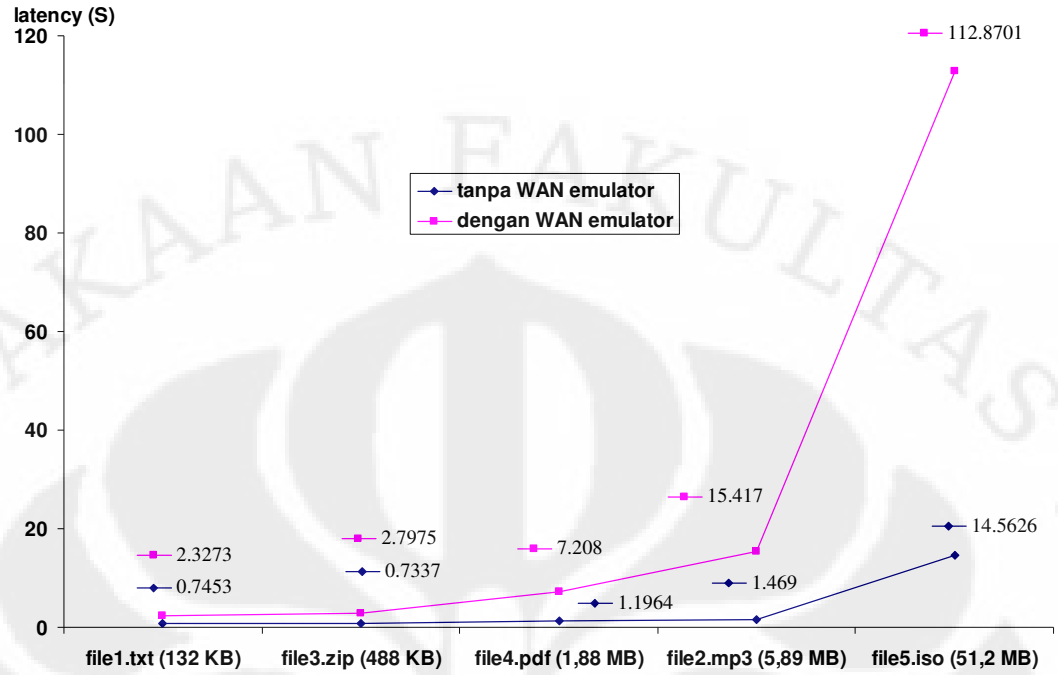
Tabel 4.4. Rata-rata *latency* dan *throughput* NAT *full cone* IPv4 murni

filetype	latency (s)		throughput (KBytes/s)	
	tanpa WAN emulator	dengan WAN emulator	tanpa WAN emulator	dengan WAN emulator
file1.txt (132 KB)	0.7453	2.3273	198.6331387	62.64841768
file3.zip (488 KB)	0.7337	2.7975	721.0006373	173.9127059
file4.pdf (1,88 MB)	1.1964	7.208	1816.100076	270.6327496
file2.mp3 (5,89 MB)	1.469	15.417	4299.033289	382.3786155
file5.iso (51,2 MB)	14.5626	112.8701	2679.804414	450.2235621

4.3.2.1. Latency pada jaringan NAT full cone IPv4 murni

Parameter pertama adalah *latency*, nilai yang disajikan didapat dari rata-rata pengolahan data *latency* jaringan NAT *full cone* IPv4 murni pada pengiriman lima tipe *file* melalui FTP. Nilai rata-rata *latency* pada jaringan NAT *full cone* IPv4 murni dapat dilihat pada Tabel 4.4.

Untuk memudahkan dalam menganalisa rata-rata *latency* pada Tabel 4.4, dapat dilihat grafiknya pada Gambar 4.4.



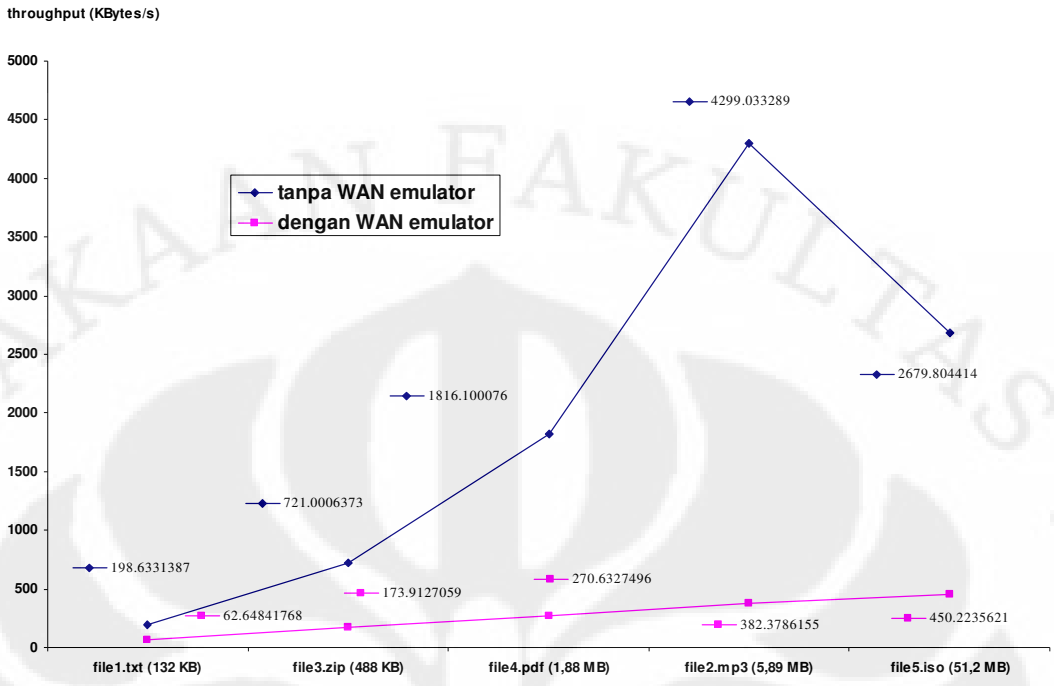
Gambar 4.4. Grafik *latency* pada topologi jaringan NAT *full cone* IPv4 murni

Pada Gambar 4.4 dapat dilihat grafik *latency* jaringan NAT *full cone* IPv4 murni. Grafik *latency* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Pada grafik, garis terlihat cenderung linear, besar *latency* pada masing-masing *file* berbanding lurus dengan ukuran *file*. Maksudnya semakin besar ukuran *file,* maka semakin lama pula waktu yang dibutuhkan untuk mengirimkan *file* tersebut ke tujuannya (semakin besar *latency*). Kelinearan tersebut berlaku pada kedua kondisi, tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah).

4.3.2.2. Throughput pada jaringan NAT full cone IPv4 murni

Parameter kedua adalah *throughput*, nilai yang disajikan didapat dari rata-rata pengolahan data *throughput* jaringan NAT *full cone* IPv4 murni pada pengiriman lima tipe *file* melalui FTP. Nilai rata-rata *throughput* pada jaringan NAT *full cone* IPv4 murni dapat dilihat pada Tabel 4.4.

Untuk memudahkan dalam menganalisa rata-rata *throughput* pada Tabel 4.4, dapat dilihat grafiknya pada Gambar 4.5.



Gambar 4.5. Grafik *throughput* pada topologi jaringan NAT *full cone* IPv4 murni

Pada Gambar 4.5 dapat dilihat grafik *throughput* dari jaringan NAT *full cone* IPv4 murni. Grafik *throughput* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Tidak seperti pada *latency*, grafik *throughput* tidak selalu linear. Jika pada *latency* grafik linear karena semakin besar ukuran *file,* maka semakin lama pula waktu yang dibutuhkan untuk mengirimkan *file* tersebut ke tujuannya (semakin besar *latency*). Sedangkan nilai *throughput* tidak selalu linear seperti *latency*, karena nilai *throughput* didapat dari pembagian ukuran *file* yang dikirimkan dengan *latency* pengiriman *file*. Seperti contohnya pada Gambar 4.5, nilai rata-rata *throughput* file5.iso lebih kecil dari file2.mp3.

4.3.3. Jaringan IPv6 Murni

Analisa hasil pengolahan data ketiga adalah pada jaringan IPv6 murni. Analisa dilakukan dan dipisahkan berdasarkan dua parameter yaitu *latency* (s) dan *throughput* (KBytes/s). Kemudian masing-masing parameter dipisahkan lagi berdasarkan kondisi jaringan, yaitu dengan simulasi WAN atau tanpa simulasi

WAN. Nilai rata-rata dari kedua parameter yaitu *latency* (s) dan *throughput* (KBytes/s) dapat dilihat pada Tabel 4.5.

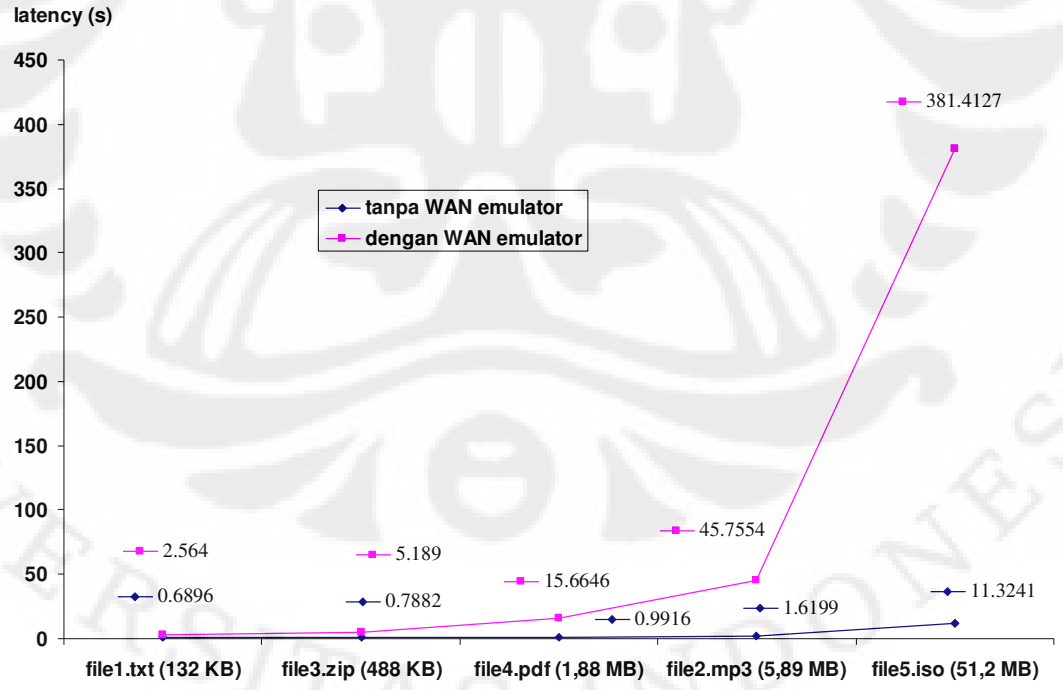
Tabel 4.5. Rata-rata *latency* dan *throughput* IPv6 murni

	latency (s)		throughput (KBytes/s)	
filetype	tanpa WAN emulator	dengan WAN emulator	tanpa WAN emulator	dengan WAN emulator
file1.txt (132 KB)	0.6896	2.564	209.8010983	56.26193711
file3.zip (488 KB)	0.7882	5.189	665.1834899	101.3034279
file4.pdf (1,88 MB)	0.9916	15.6646	2092.671361	131.3861477
file2.mp3 (5,89 MB)	1.6199	45.7554	3985.205563	140.9381837
file5.iso (51,2 MB)	11.3241	381.4127	4528.229506	146.7778788

4.3.3.1. Latency pada jaringan IPv6 murni

Parameter pertama adalah *latency*, nilai yang disajikan didapat dari rata-rata pengolahan data *latency* jaringan IPv6 murni pada pengiriman lima tipe *file* melalui FTP. Nilai rata-rata *latency* pada jaringan IPv6 murni dapat dilihat pada Tabel 4.5.

Untuk memudahkan dalam menganalisa rata-rata *latency* pada Tabel 4.5, dapat dilihat grafiknya pada Gambar 4.6.



Gambar 4.6. Grafik *latency* pada topologi jaringan IPv6 murni

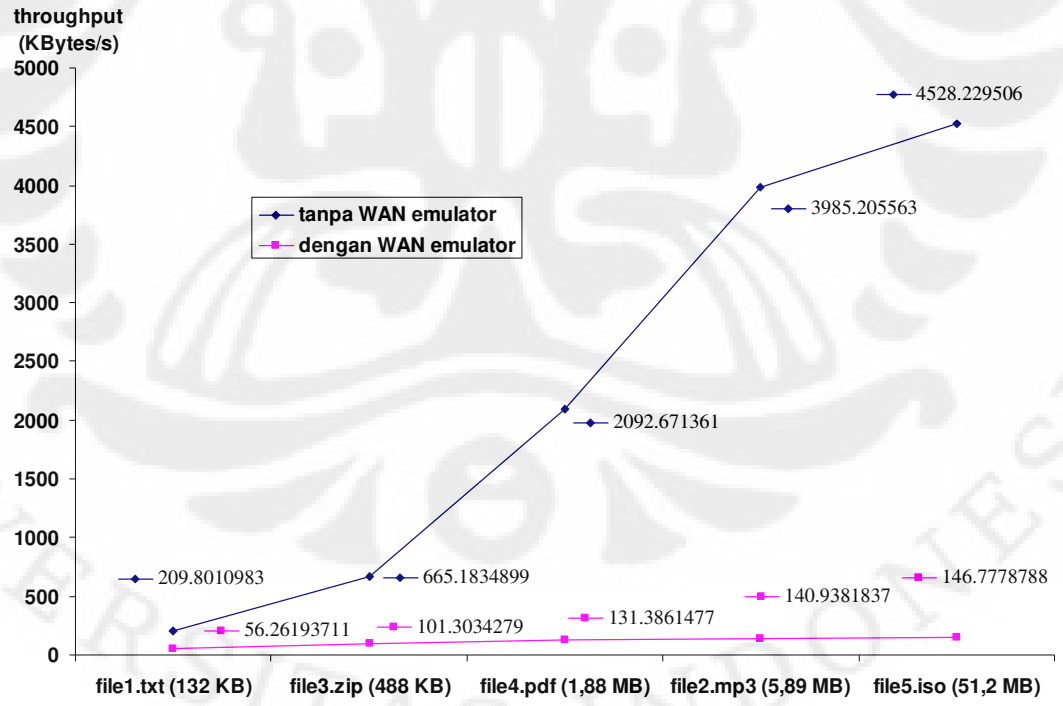
Pada Gambar 4.6 dapat dilihat grafik *latency* dari jaringan IPv6 murni. Grafik *latency* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi WAN

(garis biru) dan dengan simulasi WAN (garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Pada grafik terlihat kecenderungan kelinearan, besar *latency* pada masing-masing *file* berbanding lurus dengan ukuran *file*. Maksudnya semakin besar ukuran *file,* maka semakin lama pula waktu yang dibutuhkan untuk mengirimkan *file* tersebut ke tujuannya (semakin besar *latency*). Kelinearan tersebut berlaku pada kedua kondisi, tanpa simulasi WAN (garis biru) dan dengan simulasi WAN (garis merah).

4.3.3.2. Throughput pada jaringan IPv6 murni

Parameter kedua adalah *throughput*, nilai yang disajikan didapat dari rata-rata pengolahan data *throughput* jaringan IPv6 murni pada pengiriman lima tipe *file* melalui FTP. Nilai rata-rata *throughput* pada jaringan IPv6 murni dapat dilihat pada Tabel 4.5.

Untuk memudahkan dalam menganalisa rata-rata *throughput* pada Tabel 4.5, dapat dilihat grafiknya pada Gambar 4.7.



Gambar 4.7. Grafik *throughput* pada topologi jaringan IPv6 murni

Pada Gambar 4.7 dapat dilihat grafik *throughput* dari jaringan IPv6 murni. Grafik *throughput* dipisahkan menjadi dua, yaitu pada kondisi tanpa simulasi

WAN (garis biru) dan dengan simulasi WAN (garis merah). Grafik diurutkan berdasarkan ukuran besar (*size*) kelima *file* yang dikirimkan dengan FTP dari yang terkecil hingga yang terbesar (diurutkan pada sumbu-x). Pada gambar terlihat garis *throughput* cenderung linear membesar sebanding dengan peningkatan ukuran *file* yang dikirimkan. Sebenarnya nilai *throughput* tidak selalu linear seperti *latency*, karena nilai *throughput* didapat dari pembagian ukuran *file* yang dikirimkan dengan *latency*.

4.3.4. Jaringan *Test-bed* tanpa WAN *Emulator*

Analisa perbandingan hasil pengolahan data berikutnya adalah pada jaringan *test-bed* yang dikonfigurasi tanpa menggunakan WAN *emulator*. Perbandingan dilakukan dan dipisahkan berdasarkan dua parameter yaitu *latency* (s) dan *throughput* (KBytes/s). Kemudian, nilai rata-rata dari kedua parameter tersebut dibandingkan antar jenis/tipe jaringan.

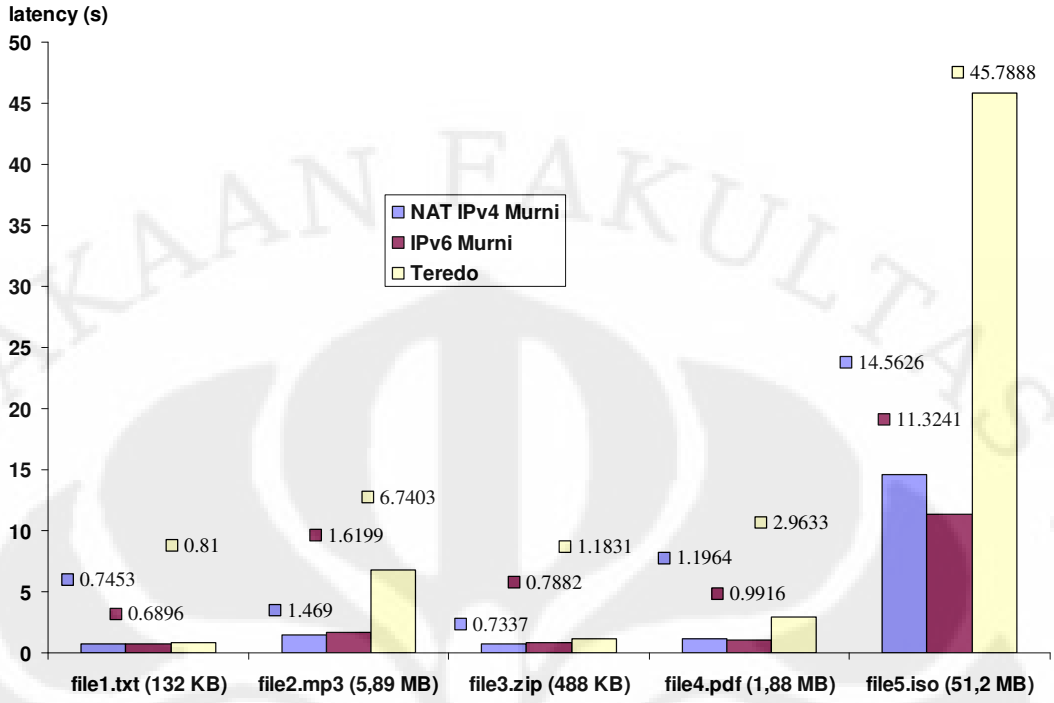
4.3.4.1. Latency pada test-bed tanpa WAN emulator

Parameter pertama adalah *latency*, nilai yang disajikan didapat dari rata-rata pengolahan data *latency* masing-masing jenis jaringan dan lima tipe *file* yang dikirimkan melalui FTP. Nilai rata-rata *latency* pada jaringan *test-bed* tanpa konfigurasi WAN *emulator* dapat dilihat pada Tabel 4.6.

Tabel 4.6. Perbandingan *latency* pada *test-bed* tanpa WAN *emulator*

	latency (s)		
filetype	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt (132 KB)	0.7453	0.6896	0.81
file2.mp3 (5,89 MB)	1.469	1.6199	6.7403
file3.zip (488 KB)	0.7337	0.7882	1.1831
file4.pdf (1,88 MB)	1.1964	0.9916	2.9633
file5.iso (51,2 MB)	14.5626	11.3241	45.7888

Untuk memudahkan dalam membandingkan rata-rata *latency* pada Tabel 4.6, dapat dilihat grafik perbandingannya pada Gambar 4.8.



Gambar 4.8. Grafik *latency* pada *test-bed* tanpa WAN *emulator*

Pada Gambar 4.8 di atas, dapat dilihat grafik perbandingan *latency* dari kelima *file* yang dikirimkan dengan FTP pada masing-masing jenis jaringan. Pada perbandingan rata-rata *latency* pengiriman kesemua tipe *file*, nilai yang dimiliki Teredo adalah yang terbesar. Ini menunjukkan jenis jaringan Teredo membutuhkan waktu paling lama untuk mengirimkan *file* FTP dibandingkan dengan jaringan NAT *full cone* IPv4 murni dan IPv6 murni. Pada pengiriman *file* FTP berukuran kecil (<150KBytes) seperti file1.txt (132KBytes), Teredo hanya lebih lambat sedikit (<18%) dari *latency* pada jaringan NAT *full cone* IPv4 murni (8,7%) dan jaringan IPv6 murni (17,46%). Perbedaan lebih terasa pada pengiriman *file* berukuran besar (>450KBytes) seperti file2.mp3 (5,89MBytes), file3.zip (488KBytes), file4.pdf (1,88 MBytes), dan file5.iso (51,2 MBytes). Pada pengiriman file3.zip Teredo lebih lambat 61,25 % dari jaringan NAT *full cone* IPv4 murni dan 50,1 % dari jaringan IPv6 murni. Pada pengiriman file4.pdf Teredo lebih lambat 147,68 % dari jaringan NAT *full cone* IPv4 murni dan 198,84 % dari jaringan IPv6 murni. Pada pengiriman file2.mp3 Teredo lebih lambat 358,84 % dari jaringan NAT *full cone* IPv4 murni dan 316,09 % dari jaringan IPv6 murni. Sementara pada pengiriman file5.iso Teredo lebih lambat 214,42 % dari jaringan NAT *full cone* IPv4 murni dan 304,35% dari jaringan IPv6 murni.

Sedangkan rata-rata *latency* antara jenis jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni tidak terlalu berbeda jauh. Rata-rata *latency* pengiriman file1.txt, file4.pdf, dan file5.iso pada jaringan IPv6 murni sedikit lebih cepat dari jaringan NAT *full cone* IPv4 murni (7,47 – 22,24 %), sehingga jaringan NAT *full cone* IPv4 murni sedikit lebih lambat dari jaringan IPv6 murni (8,08 – 28,6 %). Sementara itu, rata-rata *latency* pengiriman file2.mp3 dan file3.zip berlaku sebaliknya, yaitu jaringan NAT *full cone* IPv4 murni sedikit lebih cepat dari jaringan IPv6 murni (6,9 – 9,32 %), sehingga jaringan IPv6 murni sedikit lebih lambat dari jaringan NAT *full cone* IPv4 murni (7,43 – 10,27 %).

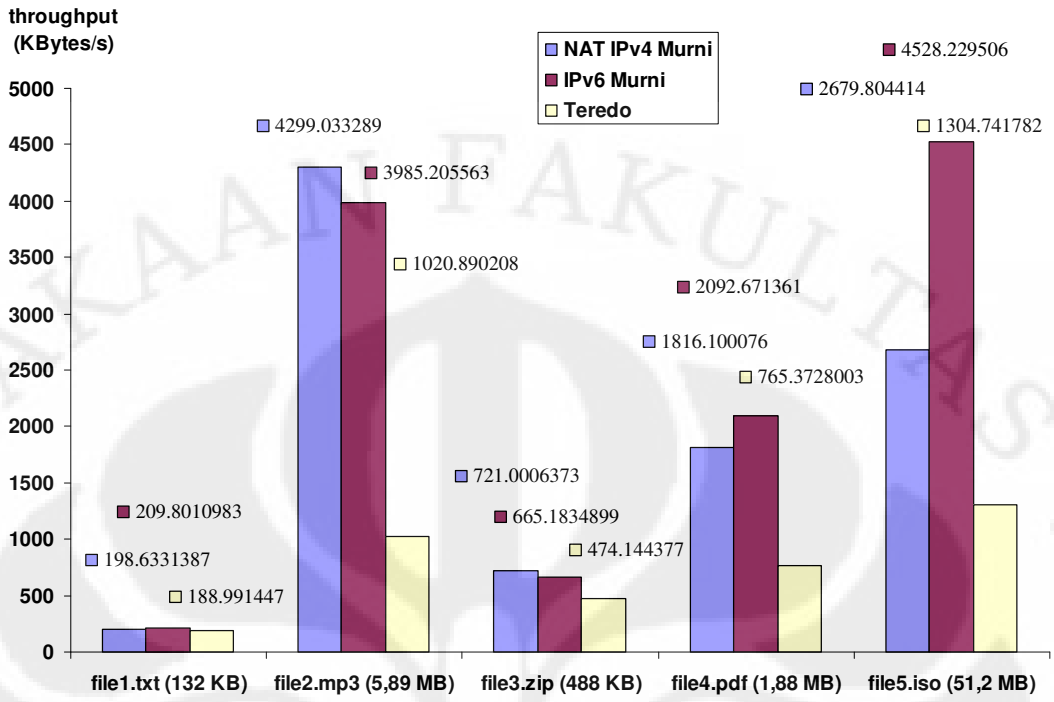
4.3.4.2. Throughput pada test-bed tanpa WAN emulator

Parameter kedua adalah *throughput*, nilai yang disajikan didapat dari rata-rata pengolahan data *throughput* masing-masing jenis jaringan dan lima tipe *file* yang dikirimkan melalui FTP. Nilai rata-rata *throughput* pada jaringan *test-bed* tanpa konfigurasi WAN *emulator* dapat dilihat pada Tabel 4.7.

Tabel 4.7. Perbandingan *throughput* pada *test-bed* tanpa WAN *emulator*

	throughput (KBytes/s)		
filetype	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt (132 KB)	198.6331387	209.8010983	188.991447
file2.mp3 (5,89 MB)	4299.033289	3985.205563	1020.890208
file3.zip (488 KB)	721.0006373	665.1834899	474.144377
file4.pdf (1,88 MB)	1816.100076	2092.671361	765.3728003
file5.iso (51,2 MB)	2679.804414	4528.229506	1304.741782

Untuk memudahkan dalam membandingkan rata-rata *throughput* pada Tabel 4.7, dapat dilihat grafik perbandingannya pada Gambar 4.9.



Gambar 4.9. Grafik *throughput* pada *test-bed* tanpa WAN *emulator*

Pada Gambar 4.9 di atas, dapat dilihat grafik perbandingan *throughput* dari kelima *file* yang dikirimkan dengan FTP pada masing-masing jenis jaringan. Pada perbandingan rata-rata *throughput* pengiriman kesemua tipe *file*, nilai yang dimiliki Teredo adalah yang terkecil. Pada pengiriman *file* FTP berukuran kecil (<150KBytes) seperti file1.txt (132KBytes), *throughput* Teredo hanya sedikit lebih kecil (<10 %) dari *throughput* pada jaringan NAT *full cone* IPv4 murni (4,85 %) dan jaringan IPv6 murni (9,92 %). Perbedaan lebih terasa pada pengiriman *file* berukuran besar (>450 KBytes) seperti file2.mp3 (5,89 MBytes), file3.zip (488KBytes), file4.pdf (1,88 MBytes), dan file5.iso (51,2 MBytes). Pada pengiriman file3.zip *throughput* Teredo lebih kecil 34,24 % dari jaringan NAT *full cone* IPv4 murni dan 28,72 % dari jaringan IPv6 murni. Pada pengiriman file4.pdf *throughput* Teredo lebih kecil 57, 86 % dari jaringan NAT *full cone* IPv4 murni dan 63,43 % dari jaringan IPv6 murni. Pada pengiriman file5.iso *throughput* Teredo lebih kecil 51,31 % dari jaringan NAT *full cone* IPv4 murni dan 71,19 % dari jaringan IPv6 murni. Sementara pada pengiriman file2.mp3 Teredo lebih lambat 76,25 % dari jaringan NAT *full cone* IPv4 murni dan 74,38 % dari jaringan IPv6 murni.

Sedangkan perbedaan rata-rata *throughput* antara jenis jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni tidak terlalu berbeda jauh, kecuali pada pengiriman file5.iso (51,2 MBytes). Rata-rata *throughput* pengiriman file1.txt dan file4.pdf, *throughput* pada jaringan IPv6 murni lebih besar dari *throughput* jaringan NAT *full cone* IPv4 murni (5,62 – 15,23 %), sehingga *throughput* jaringan NAT *full cone* IPv4 murni lebih kecil dari jaringan IPv6 murni (5,32 – 13,22 %). Sementara itu, rata-rata *throughput* pengiriman file2.mp3 dan file3.zip berlaku sebaliknya, yaitu *throughput* jaringan NAT *full cone* IPv4 murni lebih besar dari *throughput* jaringan IPv6 murni (7,87 – 8,39 %), sehingga *throughput* jaringan IPv6 murni sedikit lebih kecil dari *throughput* jaringan NAT *full cone* IPv4 murni (7,3 – 7,74 %). Perbedaan *throughput* yang besar antara jenis jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni terjadi pada pengiriman file5.iso. Nilai *throughput* jaringan NAT *full cone* IPv4 murni lebih kecil 40,82 % dari *throughput* jaringan IPv6 murni, dan *throughput* jaringan IPv6 murni lebih kecil 68,98 % dari jaringan NAT *full cone* IPv4 murni.

4.3.5. Jaringan *Test-bed* dengan WAN *Emulator*

Analisa perbandingan hasil pengolahan data selanjutnya adalah pada jaringan *test-bed* yang dikonfigurasi dengan menggunakan WAN *emulator*. Perbandingan dilakukan dan dipisahkan berdasarkan dua parameter yaitu *latency* (s) dan *throughput* (KBytes/s). Kemudian, nilai rata-rata dari kedua parameter tersebut dibandingkan antar jenis/tipe jaringan.

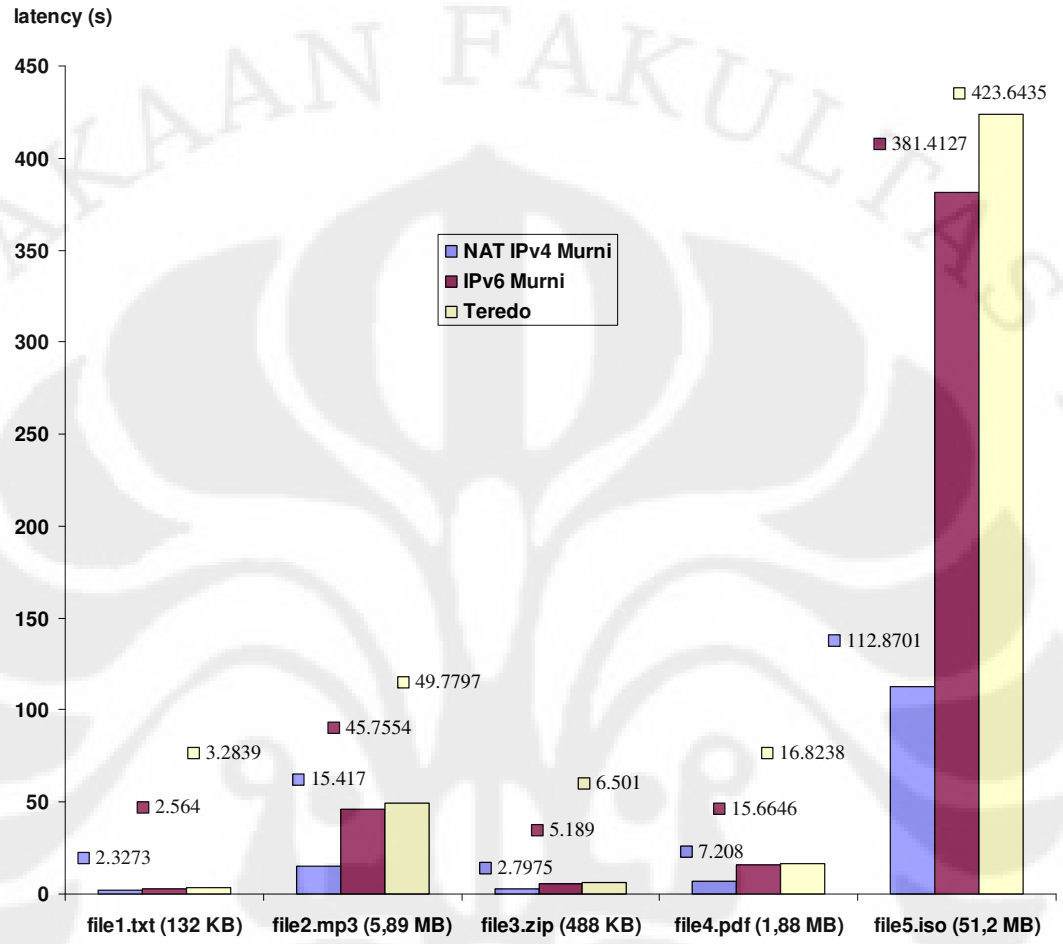
4.3.5.1. Latency pada test-bed dengan WAN emulator

Parameter pertama adalah *latency*, nilai yang disajikan didapat dari rata-rata pengolahan data *latency* masing-masing jenis jaringan dan lima tipe *file* yang dikirimkan melalui FTP. Nilai rata-rata *latency* pada jaringan *test-bed* dengan konfigurasi WAN *emulator* dapat dilihat pada Tabel 4.8

Tabel 4.8. Perbandingan *latency* pada *test-bed* dengan WAN *emulator*

	latency (s)		
filetype	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt (132 KB)	2.3273	2.564	3.2839
file2.mp3 (5,89 MB)	15.417	45.7554	49.7797
file3.zip (488 KB)	2.7975	5.189	6.501
file4.pdf (1,88 MB)	7.208	15.6646	16.8238
file5.iso (51,2 MB)	112.8701	381.4127	423.6435

Untuk memudahkan dalam membandingkan rata-rata *latency* pada Tabel 4.8, dapat dilihat grafik perbandingannya pada Gambar 4.10



Gambar 4.10. Grafik *latency* pada *test-bed* dengan WAN *emulator*

Pada Gambar 4.10 di atas, dapat dilihat grafik perbandingan *latency* dari kelima *file* yang dikirimkan dengan FTP pada masing-masing jenis jaringan. Pada perbandingan rata-rata *latency* pengiriman kesemua tipe *file*, nilai yang dimiliki Teredo adalah yang terbesar. Ini menunjukkan jenis jaringan Teredo membutuhkan waktu paling lama untuk mengirimkan *file* FTP dibandingkan dengan jaringan NAT *full cone* IPv4 murni dan IPv6 murni. Sama halnya dengan yang terjadi pada data yang diambil pada jaringan *test-bed* tanpa konfigurasi WAN *emulator*. Jika dibandingkan dengan pengiriman *file* FTP pada jaringan IPv6 murni, khususnya *file* berukuran besar (>1,8 MBytes), *latency* Teredo hanya sedikit lebih lambat. Rata-rata *latency* Teredo lebih lambat 7,4 % pada file4.pdf, lebih lambat 8,8 % pada file2.mp3, dan lebih lambat 11,07 % pada file5.iso jika

dibandingkan dengan jaringan IPv6 murni. Sementara pada *file* kecil (<500KBytes), Teredo lebih lambat 28,08 % pada file1.txt dan 25,28 % pada file3.zip jika dibandingkan dengan jaringan IPv6 murni.

Sedangkan jika dibandingkan dengan jaringan NAT *full cone* IPv4 murni, Teredo memiliki *latency* jauh lebih lambat. Dibandingkan jaringan NAT *full cone* IPv4 murni, Teredo memiliki *latency* lebih lambat 41,1 % pada file1.txt, 132,39 % pada file3.zip, 133,4 % pada file4.pdf, 222,89 % pada file2.mp3, dan 275,34 % pada file5.iso.

Rata-rata *latency* antara jenis jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni cukup berbeda jauh, kecuali pada pengiriman file1.txt. Rata-rata *latency* pengiriman file2.mp3, file3.zip, file4.pdf, dan file5.iso pada jaringan IPv6 murni lebih lambat 85,49 – 237,92 % dari jaringan NAT *full cone* IPv4 murni, sehingga jaringan NAT *full cone* IPv4 murni lebih cepat 46,09 – 70,41 % dari jaringan IPv6 murni. Sementara itu, rata-rata *latency* pengiriman file1.txt jaringan IPv6 murni sedikit lebih lambat dari jaringan NAT *full cone* IPv4 murni (10,17 %), sehingga jaringan NAT *full cone* IPv4 murni sedikit lebih cepat dari jaringan IPv6 murni (9,23 %).

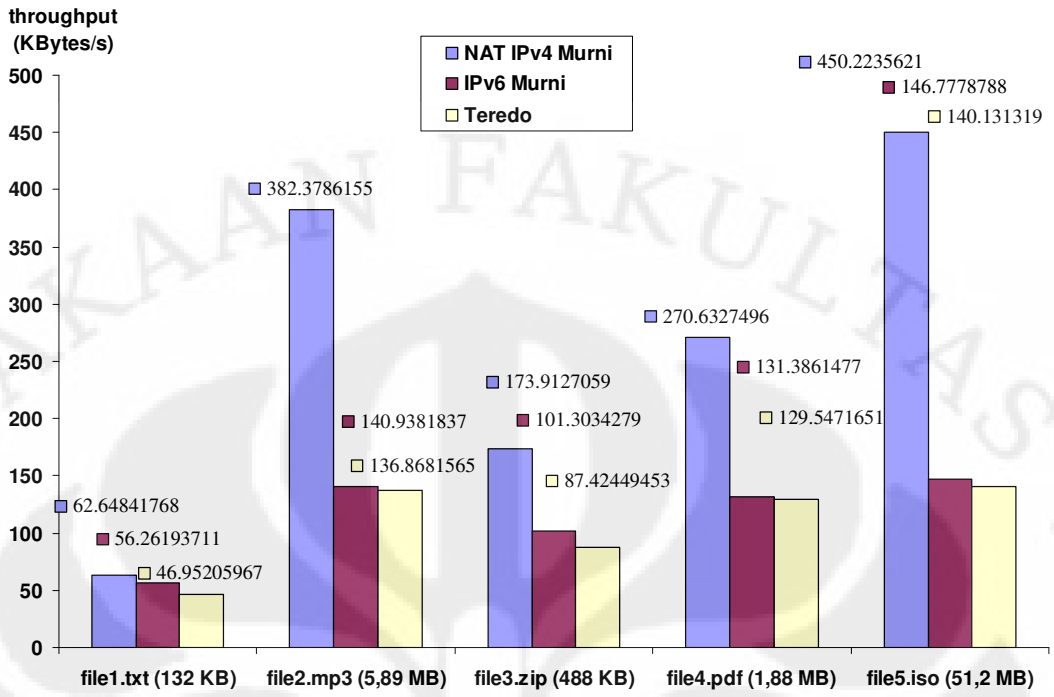
4.3.5.2. Throughput pada test-bed dengan WAN emulator

Parameter kedua adalah *throughput*, nilai yang disajikan didapat dari rata-rata pengolahan data *throughput* masing-masing jenis jaringan dan lima tipe *file* yang dikirimkan melalui FTP. Nilai rata-rata *throughput* pada jaringan *test-bed* dengan konfigurasi WAN *emulator* dapat dilihat pada Tabel 4.9.

Tabel 4.9. Perbandingan *throughput* pada *test-bed* dengan WAN *emulator*

	throughput (KBytes/s)		
filetype	NAT IPv4 Murni	IPv6 Murni	Teredo
file1.txt (132 KB)	62.64841768	56.26193711	46.95205967
file2.mp3 (5,89 MB)	382.3786155	140.9381837	136.8681565
file3.zip (488 KB)	173.9127059	101.3034279	87.42449453
file4.pdf (1,88 MB)	270.6327496	131.3861477	129.5471651
file5.iso (51,2 MB)	450.2235621	146.7778788	140.131319

Untuk memudahkan dalam membandingkan rata-rata *throughput* pada Tabel 4.9, dapat dilihat grafik perbandingannya pada Gambar 4.11.



Gambar 4.11. Grafik *throughput* pada *test-bed* dengan WAN *emulator*

Pada Gambar 4.11 di atas, dapat dilihat grafik perbandingan *throughput* dari kelima *file* yang dikirimkan dengan FTP pada masing-masing jenis jaringan. Pada perbandingan rata-rata *throughput* pengiriman kesemua tipe *file*, nilai yang dimiliki Teredo adalah yang terkecil. Sama halnya dengan yang terjadi pada data yang diambil pada jaringan *test-bed* tanpa konfigurasi WAN *emulator*. Jika dibandingkan dengan pengiriman *file* FTP pada jaringan IPv6 murni, khususnya *file*, *throughput* Teredo hanya sedikit lebih lambat. Perbedaan *throughput* yang cukup besar antara Teredo dan IPv6 terjadi pada pengiriman file1.txt (16,55 %) dan file3.txt (13,7 %). Rata-rata *throughput* Teredo lainnya hanya sedikit lebih kecil dari *throughput* IPv6 murni, yaitu 4,53 % pada file5.iso, 2,89 % pada file2.mp3, dan 1,4 % pada file4.pdf.

Sedangkan jika dibandingkan dengan jaringan NAT *full cone* IPv4 murni, Teredo memiliki *throughput* jauh lebih kecil. Dibandingkan jaringan NAT *full cone* IPv4 murni, Teredo memiliki *throughput* jauh lebih kecil, yaitu 25,05 % pada file1.txt, 49,73% pada file3.zip, 52,13 % pada file4.pdf, 64,21 % pada file2.mp3, dan 68,88 % pada file5.iso.

Rata-rata *throughput* antara jenis jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni cukup berbeda jauh, kecuali pada pengiriman file1.txt. Rata-rata *throughput* pengiriman file2.mp3, file3.zip, file4.pdf, dan file5.iso pada jaringan IPv6 murni lebih kecil 41,75 – 67,4 % dari jaringan NAT *full cone* IPv4 murni, sehingga *throughput* jaringan NAT *full cone* IPv4 murni lebih besar 71,68 – 206,74 % dari jaringan IPv6 murni. Sementara itu, rata-rata *throughput* pengiriman file1.txt jaringan IPv6 murni sedikit lebih kecil dari jaringan NAT *full cone* IPv4 murni (10,19 %), sehingga *throughput* jaringan NAT *full cone* IPv4 murni sedikit lebih besar dari jaringan IPv6 murni (11,35 %).

4.4. ANALISA PERBANDINGAN KESELURUHAN

Pada analisa rata-rata hasil pengolahan data untuk pengiriman setiap satu *file* FTP tertentu terlihat bahwa semakin besar nilai *latency* menyebabkan semakin kecil nilai *throughput*, atau semakin kecil nilai *latency* menyebabkan semakin besar nilai *throughput*. Ini menunjukkan bahwa adanya hubungan antara *throughput* dan *latency* pada pengiriman satu *file* yang sama, karena nilai *throughput* didapat dari pembagian ukuran *file* yang dikirimkan dengan *latency* pengiriman *file*, sehingga semakin besar *latency* menyebabkan semakin kecil nilai *throughput* atau sebaliknya. Untuk satu *file* yang sama, besar-kecilnya *latency* pengiriman *file* tersebut mempengaruhi nilai *throughput*, atau dengan kata lain hubungan nilai keduanya adalah berbanding terbalik.

Analisa juga menunjukkan jika dilihat dari sisi tipe/jenis *file* yang dikirimkan tidak mempengaruhi parameter yang dibandingkan (*latency* dan *throughput*). Ukuran *file* lebih berpengaruh daripada tipe *file*, karena pada mode *binary* aplikasi FTP dari sisi pengirim mengirimkan *file* dengan bit per bit, sehingga sisi penerima menerimanya sebagai *bitstream* (aliran bit-bit), tidak terpengaruh apa pun tipe/jenis *file* yang dikirimkan, lebih berpengaruh ukuran *file* yang dikirimkan.

Analisa sebelumnya menunjukkan metode *tunneling* IPv6 Teredo memiliki kinerja lebih buruk daripada jaringan NAT *full cone* IPv4 murni dan jaringan IPv6 murni, namun tidak terlampau jauh kinerjanya (sedikit lebih buruk) dari jaringan IPv6 murni pada jaringan WAN sebenarnya. Kinerja Teredo lebih

buruk dari jaringan lainnya pertama dikarenakan oleh Teredo membutuhkan waktu tambahan untuk proses enkapsulasi dan dekapsulasi paket FTP yang dikirimkan. Penyebab lainnya adalah pada Teredo, FTP *server* mengaktifkan mode kerja *extended passive mode*, yaitu *extension* yang memungkinkan FTP *server* dapat melakukan transfer data antar *node* jaringan dengan multiprotokol dengan tambahan fitur keamanan. *Extended passive mode* memerintahkan pengecekan untuk setiap paket meskipun sudah dalam status transfer data sehingga memperlambat waktu yang diperlukan untuk transfer data (memperbesar *latency*). Jadi kinerja Teredo yang merupakan jaringan dengan multiprotokol lebih buruk daripada jaringan dengan protokol tunggal seperti NAT *full cone* IPv4 murni dan IPv6 murni, namun demikian, tidak terlampau jauh kinerjanya (hanya sedikit lebih buruk) dari jaringan IPv6 murni pada jaringan WAN sebenarnya, sehingga cocok digunakan untuk memberikan koneksi IPv6 kepada *node* jaringan di belakang NAT *full cone* IPv4 yang belum mendukung IPv6 ketika nanti IPv6 banyak digunakan.

BAB V
KESIMPULAN

1. Pada transfer *file* ukuran besar (51,2 MBytes) tanpa simulasi WAN, *latency* Teredo lebih besar 214,43% dari jaringan NAT IPv4 murni dan lebih besar 304,35% dari jaringan IPv6 murni, *throughput* Teredo lebih kecil 51,31% dari jaringan NAT IPv4 murni dan lebih kecil 71,19 % dari jaringan IPv6 murni.
2. Pada transfer *file* ukuran kecil (132 KBytes) tanpa simulasi WAN, *latency* Teredo lebih besar 8,68% dari jaringan NAT IPv4 murni dan lebih besar 17,46% dari jaringan IPv6 murni, *throughput* Teredo lebih kecil 4,85% dari jaringan NAT IPv4 murni dan lebih kecil 9,92% dari jaringan IPv6 murni.
3. Pada transfer *file* ukuran besar (51,2 MBytes) dengan simulasi WAN, *latency* Teredo lebih besar 275,34% dari jaringan NAT IPv4 murni dan lebih besar 11,07% dari jaringan IPv6 murni, *throughput* Teredo lebih kecil 68,88% dari jaringan NAT IPv4 murni dan lebih kecil 4,53% dari jaringan IPv6 murni.
4. Pada transfer *file* berukuran kecil (132KBytes) dengan simulasi WAN, *latency* Teredo lebih besar 41,1% dari jaringan NAT IPv4 murni dan lebih besar 28,08 dari jaringan IPv6 murni, *throughput* Teredo lebih kecil 25,05% dari jaringan NAT IPv4 murni dan lebih kecil 16,55% dari jaringan IPv6 murni.
5. Perbedaan tipe/jenis *file* yang dikirimkan tidak mempengaruhi kinerja (nilai *latency* maupun *throughput*), namun ukuran *file* lebih berpengaruh daripada tipe *file*, karena pada mode *binary* aplikasi FTP dari sisi pengirim mengirimkan *file* dengan bit per bit, sehingga sisi penerima menerimanya sebagai *bitstream* (aliran bit-bit).
6. Metode *tunneling* IPv6 Teredo memiliki kinerja sedikit lebih buruk daripada jaringan IPv6 murni pada simulasi jaringan WAN sebenarnya, sehingga cocok digunakan untuk memberikan koneksi IPv6 kepada *node* jaringan di belakang NAT IPv4 yang belum mendukung IPv6 ketika IPv6 telah banyak digunakan.

DAFTAR ACUAN

[1] S. Deering, R. Hinden (Desember 1998),"Internet Protocol, Version 6 (IPv6) Specification," *RFC : 2460*, hal. 2. Diakses 20 November 2007, dari ietf. http://www.ietf.org/rfc/rfc2460.txt

[2] I-Ping Hsieh, Shang-Juh Kao, " Managing the Co-existing Network of IPv6 and IPv4 under Various Transition Mechanisms," *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, 2005 : hal. 1.

[3] (Januari 2007), " IPv6 Transition Technologies," *Microsoft Corporation Whitepaper*, hal. 8. Diakses 10 September 2007, dari Microsoft. http://www.microsoft.com/

[4] Andrew S. Tanenbaum, *Computer Natworks Fourth Edition*, (New Jersey : Pearson Education Inc., – Prentice Hall PTR, 2003), hal. 445.

[5] Charles M. Kozierok (20 September 2005), "Overview of FTP Operation," *FTP Overview, History and Standards*, hal. 2. Diakses 2 November 2007, dari tcpipguide. http://www.tcpipguide.com/free/t_FTPOverviewHistoryandStandards-2.htm

[6] (2007), "Emulating wide area network delays," *Net : Netem*, Diakses 25 November 2007, dari linux-foundation. http://www.linux-foundation.org/

[7] Candace Leiden, Marshall Wilensky, *TCP/IP FOR DUMMIES 2ND EDITION*, (Foster City : IDG Books Worldwide, Inc., 1997), hal. 125.

[8] (2006), "FTP 101 - A Beginner's Guide," *FTP New User Guide*, Diakses 2 November 2007, dari FTPplanet. http://www.FTPplanet.com/

[9] Audrey Octavia. "*Interoperability* Sistem dengan Metode *Tunneling* 6to4 dan ISATAP pada Jaringan Lokal IPv6." Skripsi, Program Sarjana Fakultas Teknik UI, Depok, 2006, hal. 30.

DAFTAR PUSTAKA

Deering S., R. Hinden (Desember 1998), " Internet Protocol, Version 6 (IPv6) Specification," *RFC : 2460*. Diakses 20 November 2007, dari ietf. http://www.ietf.org/rfc/rfc2460.txt

Deering S., R. Hinden (Juli 1998), " IPv6 Addressing Architecture," *RFC : 2373*. Diakses 20 November 2007, dari ietf. http://www.ietf.org/rfc/rfc2373.txt

"FTP 101 - A Beginner's Guide," *FTP New User Guide*, 2006. Diakses 2 November 2007, dari FTPplanet. http://www.FTPplanet.com/

"File Transfer Protocol," *Wikipedia, the free ensyclopedia*, 1 November 2007. Diakses 2 November 2007, dari wikipedia. http://en.wikipedia.org/File_Transfer_Protocol

Hoagland, Dr. James, "The Teredo Protocol : Tunneling Past Network Security and Other Security Implications," *Symantec Advanced Threat Research*, 2007. Diakses 7 September 2007, dari Symantec. http://www.symantec.com/

Hsieh, I-Ping, Shang-Juh Kao, " Managing the Co-existing Network of IPv6 and IPv4 under Various Transition Mechanisms," *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, 2005.

Huitema, C. (Februari 2006), "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," *RFC : 4380*. Diakses 8 September 2007, dari ietf. http://www.ietf.org/rfc/rfc4380.txt

"IPv6 Transition Technologies," *Microsoft Corporation Whitepaper*, Januari 2007. Diakses 10 September 2007, dari Microsoft.

http://www.microsoft.com/

Kozierok, Charles M. (20 September 2005), " FTP Overview, History and Standards, ". Diakses 2 November 2007, dari tcpipguide. http://www.tcpipguide.com/free/t_FTPOverviewHistoryandStandards.htm

Leiden, Candace, Marshall Wilensky, *TCP/IP FOR DUMMIES 2ND EDITION*, (Foster City : IDG Books Worldwide, Inc., 1997).

"Network address translation," *Wikipedia, the free ensyclopedia*, 23 Oktober 2007. Diakses 24 Oktober 2007, dari wikipedia. http://en.wikipedia.org/Network_address_translation

Postel, J., J. Reynolds, (Oktober 1985), " FILE TRANSFER PROTOCOL (FTP)," *RFC : 0959*. Diakses 2 November 2007, dari ietf. http://www.ietf.org/rfc/rfc0959.txt

Rooney, Tim (Februari 2007), " IPv4-to-IPv6 Transition Strategies," *BT Diamon IP Whitepaper*. Diakses 8 September 2007, dari BT Diamon IP. http://www.bt.diamonip.com/

Sanders, Chris, *PRACTICAL PACKET ANALYSIS : Using Wireshark to Solve Real-World Network Problems*, (San Francisco : No Starch Press, Inc., 2007).

Tanenbaum, Andrew S., *Computer Natworks Fourth Edition*, (New Jersey : Pearson Education Inc., – Prentice Hall PTR, 2003).

"Teredo tunneling," *Wikipedia, the free ensyclopedia*, 18 Agustus 2007. Diakses 7 September 2007, dari wikipedia. http://en.wikipedia.org/Teredo_tunneling

"Teredo Overview," *Microsoft TechNet*, 15 Mei 2007. Diakses 7 September 2007, dari Microsoft TechNet. http://www.microsoft.com/

Tyson, Jeff, "Introduction How Network Address Translation Works," *How Network Address Translation Works*. Diakses 24 Oktober 2007, dari howstuffworks. http://computer.howstuffworks.com/nat.htm

LAMPIRAN

Lampiran 1 Perintah konfigurasi jaringan pada topologi Teredo

1.1. Perintah untuk mengkonfigurasi Teredo *client* pada Windows XP

```
C:\> netsh int ipv6 set teredo client 202.154.0.1
```

1.2. Perintah untuk mengkonfigurasi *router* NAT pada Ubuntu 6.06

1.2.1. Konfigurasi inisialisasi pada NAT

```
#!/bin/bash

echo "1" > /proc/sys/net/ipv4/ip_forward
#echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route add default gw 167.205.0.1
/sbin/iptables -t nat -A POSTROUTING -o atas -j MASQUERADE
/sbin/iptables -t nat -A PREROUTING -i atas -j DNAT --to-destination 192.168.0.2

#route -A inet6 add default gw 2001:b::1
```

1.2.2. Konfigurasi teredo pada NAT (miredo)

```
#~/bin/bash

/usr/local/sbin/miredo -f
```

1.3. Perintah untuk mengkonfigurasi *router* 1 pada Ubuntu 6.06

```
#!/bin/bash

echo "1" > /proc/sys/net/ipv4/ip_forward
#echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

#route add default gw 202.154.0.1
#route -A inet6 add default gw 2001:b::1
```

1.4. Perintah untuk mengkonfigurasi Teredo *server*

1.4.1. Konfigurasi Teredo dengan menggunakan tools miredo

```
miredo.conf

#! /usr/local/sbin/miredo -f -c
#
# Sample configuration file for Miredo
# $Id: miredo.conf-in 1715 2006-08-24 16:05:53Z remi $

# Please refer to the miredo.conf(5) man page for details.
```

```
# Miredo can safely run as a Teredo client, which is the default.
#RelayType client

# Name of the network tunneling interface.
InterfaceName      teredo

# Depending on the local firewall/NAT rules, you might need to force
# Miredo to use a fixed UDP port and or IPv4 address.
BindPort  3545
BindAddres      202.154.0.2

#SyslogFacility    user

## CLIENT-SPECIFIC OPTIONS
# The hostname or primary IPv4 address of the Teredo server.
# This setting is required if Miredo runs as a Teredo client.
#*********************************************************************#
#      teredo.remlab.net is an experimental service for testing only.      #
# Please use another server for production and/or large scale deployments. #
#*********************************************************************#
#ServerAddress teredo.remlab.net
#ServerAddress2 teredo2.remlab.net

## RELAY-SPECIFIC OPTIONS
#Prefix 2001:0::
Prefix 3ffe:831f::
#InterfaceMTU 1280
```

miredo-server.conf

```
Prefix 2001::
InterfaceMTU 1280
ServerBindAddress 202.154.0.2
ServerBindAddress2 202.154.0.3
```

memulai miredo

```
#!/bin/bash

/usr/local/sbin/miredo-server
```

1.4.2. Konfigurasi addressing, routing dan forwarding

```
#!/bin/bash
# Fungsinya untuk menginisialisi routing dan ip forwarding

# Enable forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

```
# Konfigurasi route static
route add default gw 202.154.0.1
route -A inet6 add default gw 2001:b::1
```

1.5. Perintah untuk mengkonfigurasi Teredo *relay*

1.5.1. Konfigurasi Teredo dengan menggunakan tools miredo

```
#! /usr/local/sbin/miredo -f -c
#
# Sample configuration file for Miredo
# $Id: miredo.conf-in 1715 2006-08-24 16:05:53Z remi $

# Please refer to the miredo.conf(5) man page for details.

# Miredo can safely run as a Teredo client, which is the default.
RelayType relay

# Name of the network tunneling interface.
InterfaceName      teredo

# Depending on the local firewall/NAT rules, you might need to force
# Miredo to use a fixed UDP port and or IPv4 address.
BindPort  3545
BindAddress        222.124.0.2

#SyslogFacility    user

## CLIENT-SPECIFIC OPTIONS
# The hostname or primary IPv4 address of the Teredo server.
# This setting is required if Miredo runs as a Teredo client.
#*************************************************************************#
#      teredo.remlab.net is an experimental service for testing only.      #
# Please use another server for production and/or large scale deployments. #
#*************************************************************************#
#ServerAddress teredo.remlab.net
#ServerAddress2 teredo2.remlab.net

## RELAY-SPECIFIC OPTIONS
Prefix 2001:0::
InterfaceMTU 1280

memulai miredo

#!/bin/bash

modprobe tun
#sysctl -w net.ipv6.conf.all.forwarding=1
/usr/local/sbin/miredo
```

1.5.2. Konfigurasi addressing, routing dan forwarding

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

```
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route add default gw 222.124.0.1
route -A inet6 add default gw 2001:c::1
```

1.6. Perintah untuk mengkonfigurasi *router* 2 pada Ubuntu 6.06

```
#!/bin/bash

echo "1" > /proc/sys/net/ipv4/ip_forward
echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route -A inet6 add 2001::/32 gw 2001:c::2

#route add default gw 222.124.0.1
#route -A inet6 add default gw 2001:c::1
```

1.7. Perintah untuk mengkonfigurasi FTP *server* (*host* IPv6) pada Ubuntu 6.06

```
#!/bin/bash

ifconfig eth0 up

ifconfig eth0 inet6 add 2001:a::2/64

route -A inet6 add default gw 2001:a::1
```

Lampiran 2 Perintah konfigurasi jaringan pada topologi NAT IPv4 murni

2.1. Perintah untuk mengkonfigurasi NAT pada Ubuntu 6.06

Sama seperti pada topologi Teredo

2.2. Perintah untuk mengkonfigurasi *router* 1 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig bawah 167.205.0.1 netmask 255.255.255.0
ifconfig atas 152.118.1.2 netmask 255.255.255.0

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

route add -net 152.118.2.0 netmask 255.255.255.0 gw 152.118.1.3
route add -net 152.118.3.0 netmask 255.255.255.0 gw 152.118.1.3
```

2.3. Perintah untuk mengkonfigurasi *router* 2 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig atas 152.118.1.3 netmask 255.255.255.0
ifconfig bawah 152.118.2.3 netmask 255.255.255.0

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

route add -net 167.205.0.0 netmask 255.255.255.0 gw 152.118.1.2
route add -net 152.118.3.0 netmask 255.255.255.0 gw 152.118.2.4
```

2.4. Perintah untuk mengkonfigurasi *router* 3 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig tengah 152.118.2.4 netmask 255.255.255.0
ifconfig atas 152.118.3.5 netmask 255.255.255.0

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

route add -net 152.118.1.0 netmask 255.255.255.0 gw 152.118.2.3
route add -net 167.205.0.0 netmask 255.255.255.0 gw 152.118.2.3
```

2.5. Perintah untuk mengkonfigurasi FTP *server* pada Ubuntu 6.06

```
#!/bin/bash

ifconfig eth0 152.118.3.6 netmask 255.255.255.0

route add default gw 152.118.3.5
```

Lampiran 3 Perintah konfigurasi jaringan pada topologi IPv6 murni

3.1. Perintah untuk mengkonfigurasi FTP *client* pada Ubuntu 6.06

```
C:\> netsh int ipv6 add address atas 2001:e::2
C:\> netsh int ipv6 add route atas 2001:e::1
```

3.2. Perintah untuk mengkonfigurasi *router* 1 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig atas up
ifconfig bawah up

ifconfig bawah inet6 add 2001:e::1/64
ifconfig atas inet6 add 2001:d::2/64

echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route -A inet6 add 2001:c::/64 gw 2001:d::1
route -A inet6 add 2001:b::/64 gw 2001:d::1
route -A inet6 add 2001:a::/64 gw 2001:d::1
```

3.3. Perintah untuk mengkonfigurasi *router* 2 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig atas up
ifconfig bawah up

ifconfig atas inet6 add 2001:c::2/64
ifconfig bawah inet6 add 2001:d::1/64

tc qdisc add dev atas root netem delay 50ms
tc qdisc add dev bawah root netem delay 50ms

echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route -A inet6 add 2001:e::/64 gw 2001:d::2
route -A inet6 add 2001:b::/64 gw 2001:c::1
route -A inet6 add 2001:a::/64 gw 2001:c::1
```

3.4. Perintah untuk mengkonfigurasi *router* 3 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig atas up
ifconfig bawah up

ifconfig bawah inet6 add 2001:b::2/64
ifconfig atas inet6 add 2001:c::1/64

echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

```
route -A inet6 add 2001:a::/64 gw 2001:b::1
route -A inet6 add 2001:d::/64 gw 2001:c::2
route -A inet6 add 2001:e::/64 gw 2001:c::2
```

3.5. Perintah untuk mengkonfigurasi *router* 4 pada Ubuntu 6.06

```
#!/bin/bash

ifconfig atas up
ifconfig tengah up

ifconfig atas inet6 add 2001:a::1/64
ifconfig tengah inet6 add 2001:b::1/64

echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

route -A inet6 add 2001:c::/64 gw 2001:b::2
route -A inet6 add 2001:d::/64 gw 2001:b::2
route -A inet6 add 2001:e::/64 gw 2001:b::2
```

3.6. Perintah untuk mengkonfigurasi FTP *server* pada Ubuntu 6.06

```
#!/bin/bash

ifconfig eth0 up

ifconfig eth0 inet6 add 2001:a::2/64

route -A inet6 add default gw 2001:a::1
```

Lampiran 4 Konfigurasi FTP *server*

```
# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
#listen=YES ;(NAT IPv4 murni, pagar dihilangkan)
#
# Run standalone with IPv6?
# Like the listen parameter, except vsftpd will listen on an IPv6 socket
# instead of an IPv4 one. This parameter and the listen parameter are mutually
# exclusive.
#listen_ipv6=YES (pada topologi teredo dan IPv6 murni, pagar dihilangkan)
#
#listen_address6=2001:a::2
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES
#
anon_root=/home/ftp
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
#
# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.
anon_mkdir_write_enable=YES
#
# Activate directory messages - messages given to remote users when they
# go into a certain directory.
```

```
dirmessage_enable=YES
#
# Activate logging of uploads/downloads.
xferlog_enable=YES
#
# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES
#
# If you want, you can arrange for uploaded anonymous files to be owned by
# a different user. Note! Using "root" for uploaded files is not
# recommended!
#chown_uploads=YES
#chown_username=whoever
#
# You may override where the log file goes if you like. The default is shown
# below.
#xferlog_file=/var/log/vsftpd.log
#
# If you want, you can have your log file in standard ftpd xferlog format
#xferlog_std_format=YES
#
# You may change the default value for timing out an idle session.
#idle_session_timeout=600
#
# You may change the default value for timing out a data connection.
#data_connection_timeout=120
#
# It is recommended that you define on your system a unique user which the
# ftp server can use as a totally isolated and unprivileged user.
#nopriv_user=ftpsecure
#
# Enable this and the server will recognise asynchronous ABOR requests. Not
# recommended for security (the code is non-trivial). Not enabling it,
# however, may confuse older FTP clients.
#async_abor_enable=YES
#
# By default the server will pretend to allow ASCII mode but in fact ignore
# the request. Turn on the below options to have the server actually do ASCII
# mangling on files when in ASCII mode.
# Beware that on some FTP servers, ASCII support allows a denial of service
# attack (DoS) via the command "SIZE /big/file" in ASCII mode. vsftpd
# predicted this attack and has always been safe, reporting the size of the
# raw file.
# ASCII mangling is a horrible feature of the protocol.
#ascii_upload_enable=YES
#ascii_download_enable=YES
#
# You may fully customise the login banner string:
ftpd_banner=Welcome to blah FTP service.
```

```
#
# You may specify a file of disallowed anonymous e-mail addresses. Apparently
# useful for combatting certain DoS attacks.
#deny_email_enable=YES
# (default follows)
#banned_email_file=/etc/vsftpd.banned_emails
#
# You may restrict local users to their home directories.  See the FAQ for
# the possible risks in this before using chroot_local_user or
# chroot_list_enable below.
#chroot_local_user=YES
#
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().
#chroot_list_enable=YES
# (default follows)
#chroot_list_file=/etc/vsftpd.chroot_list
#
# You may activate the "-R" option to the builtin ls. This is disabled by
# default to avoid remote users being able to cause excessive I/O on large
# sites. However, some broken FTP clients such as "ncftp" and "mirror" assume
# the presence of the "-R" option, so there is a strong case for enabling it.
#ls_recurse_enable=YES
#
#
# Debian customization
#
# Some of vsftpd's settings don't fit the Debian filesystem layout by
# default.  These settings are more Debian-friendly.
#
# This option should be the name of a directory which is empty.  Also, the
# directory should not be writable by the ftp user. This directory is used
# as a secure chroot() jail at times vsftpd does not require filesystem
# access.
secure_chroot_dir=/var/run/vsftpd
#
# This string is the name of the PAM service vsftpd will use.
pam_service_name=vsftpd
#
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
#
# This option specifies the location of the RSA key to use for SSL
# encrypted connections.
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

Lampiran 5 Konfigurasi Network Emulator

```
#!/bin/bash

tc qdisc del dev atas root
tc qdisc del dev tengah root
tc qdisc del dev bawah root

tc qdisc add dev atas root netem delay 50ms
tc qdisc add dev tengah root netem delay 50ms
tc qdisc add dev bawah root netem delay 50ms
```

Lampiran 6 Spesifikasi Perangkat Keras

Topologi	Node	Processor	Memory	LAN card	OS
Teredo	FTP client	AMD Athlon XP-M 1.50GHz	512	Ethernet 10/100 Mbps	Windows XP SP 2
	Router NAT	Intel(R) Pentium(R) III CPU 500MHz	128	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 1	Intel(R) Pentium(R) 4 CPU 1.70GHz	256	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Teredo relay	Intel(R) Pentium(R) 4 CPU 1.70GHz	384	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 2	Intel(R) Pentium(R) 4 CPU 2.40GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	FTP server	Intel(R) Pentium(R) 4 CPU 2.26GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Teredo server	Intel(R) Pentium(R) 4 CPU 2.00GHz	256	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
NAT IPv4 murni	FTP client	AMD Athlon XP-M 1.50GHz	512	Ethernet 10/100 Mbps	Windows XP SP 2
	Router NAT	Intel(R) Pentium(R) III CPU 500MHz	128	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 1	Intel(R) Pentium(R) 4 CPU 1.70GHz	256	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 2	Intel(R) Pentium(R) 4 CPU 1.70GHz	384	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 3	Intel(R) Pentium(R) 4 CPU 2.40GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	FTP server	Intel(R) Pentium(R) 4 CPU 2.26GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
IPv6 murni	FTP client	AMD Athlon XP-M 1.50GHz	512	Ethernet 10/100 Mbps	Windows XP SP 2
	Router 1	Intel(R) Pentium(R) III CPU 500MHz	128	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 2	Intel(R) Pentium(R) 4 CPU 1.70GHz	256	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 3	Intel(R) Pentium(R) 4 CPU 1.70GHz	384	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	Router 4	Intel(R) Pentium(R) 4 CPU 2.40GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS
	FTP server	Intel(R) Pentium(R) 4 CPU 2.26GHz	512	Ethernet 10/100 Mbps	Ubuntu Linux 6.06 LTS

Lampiran 7 Hasil Pengambilan Data

Data tanpa simulasi jaringan sebenarnya

file1.txt

file1.txt	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	0.58	245.8815264	0.672	214.4924707	0.769	198.5536377
2	0.587	241.8549814	0.74	194.6797256	0.796	191.2448506
3	0.73	196.5666826	0.637	225.7544912	0.915	166.805
4	0.763	186.0022813	0.729	196.8504014	0.745	204.8620811
5	0.628	227.4394072	0.637	225.7693916	0.821	185.5555996
6	0.638	224.7569404	0.584	245.5428691	0.769	198.4262295
7	0.66	214.9879111	0.746	192.9293252	0.799	190.6292656
8	0.922	155.0525293	0.704	204.7566445	0.849	179.4613438
9	0.955	149.015875	0.718	200.223749	0.876	173.9618896
10	0.99	144.773252	0.729	197.011915	0.761	200.4145723
Rata-rata	0.7453	198.6331387	0.6896	209.8010983	0.81	188.991447

file2.mp3

file2.mp3	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	1.36	4728.013443	1.508	4271.457818	7.208	948.6242207
2	1.6	3749.76601	1.543	4175.105659	7.365	929.4910547
3	1.625	3954.276747	1.724	3741.220327	6.812	1005.21123
4	1.383	4578.137313	1.624	3963.2963	6.442	1059.057224
5	1.266	5077.839821	1.646	3910.355466	6.866	994.7534629
6	1.554	4138.795397	1.633	3949.318041	6.294	1089.067935
7	1.383	4426.304327	1.733	3718.608438	6.816	1005.148267
8	1.6	4018.76972	1.537	4196.610924	7.608	897.7379199
9	1.43	4498.215787	1.667	3862.012628	5.821	1174.174399
10	1.489	3820.214329	1.584	4064.07003	6.171	1105.636362
Rata-rata	1.469	4299.033289	1.6199	3985.205563	6.7403	1020.890208

file3.zip

file3.zip	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	0.782	664.4259268	0.839	621.8359902	1.15	480.4332061
2	0.68	767.8743633	0.754	693.3427314	1.175	471.5668672
3	0.615	843.8767842	0.83	627.3884473	1.038	532.688917
4	0.708	733.6034619	0.767	680.3702529	1.018	545.4511475
5	0.748	696.1908408	0.824	635.5526309	1.225	451.1973477
6	0.793	657.3142275	0.772	677.9166963	1.206	459.3954141
7	0.574	907.0764346	0.786	667.0796533	1.437	385.1994834
8	0.753	691.372293	0.738	707.8489404	1.198	462.1643193
9	0.757	687.8494277	0.729	719.5006631	1.006	551.737125
10	0.927	560.4226133	0.843	620.9988936	1.378	401.6099424
Rata-rata	0.7337	721.0006373	0.7882	665.1834899	1.1831	474.144377

file4.pdf

file4.pdf	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	1.24	1656.032083	0.894	2303.999249	2.8	780.4022783
2	0.887	2313.878181	0.893	2304.913204	2.511	870.2811807
3	0.913	2245.920225	0.966	2129.566636	2.429	895.5232461
4	1.602	1281.323625	1.106	1863.626224	2.805	779.0633174
5	0.905	2264.740189	0.893	2298.015567	2.99	734.3238076
6	1.292	1587.53886	1.146	1794.452964	2.755	793.4389209
7	1.003	2045.254556	0.949	2172.817495	2.094	1044.975512
8	1.045	1965.317991	1.067	1924.814424	3.66	597.9075566
9	1.206	1703.939427	1.085	1891.461704	3.473	628.2031748
10	1.871	1097.055626	0.917	2243.046147	4.116	529.6090088
Rata-rata	1.1964	1816.100076	0.9916	2092.671361	2.9633	765.3728003

file5.iso

file5.iso	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	18.816	1913.406755	10.538	4821.064938	49.246	1206.885395
2	11.577	3362.211076	12.63	4235.67506	44.178	1341.717928
3	14.911	2505.98885	10.489	4750.590165	48.435	1225.366465
4	12.218	2680.782375	11.292	4522.511236	44.436	1334.908355
5	14.12	2524.173406	10.649	4788.696979	44.378	1335.481516
6	15.055	2734.8688	10.36	4607.350682	50.443	1177.968537
7	19.028	1851.646395	11.964	4274.043846	36.862	1605.44273
8	17.779	2084.867019	12.308	4263.994378	45.085	1314.888565
9	10.168	3686.418316	11.643	4429.672776	48.752	1217.743748
10	11.954	3453.681152	11.368	4588.695002	46.073	1287.014582
Rata-rata	14.5626	2679.804414	11.3241	4528.229506	45.7888	1304.741782

Data dengan simulasi jaringan sebenarnya

file1.txt

file1.txt	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	3.505	40.67891211	2.668	54.15884961	3.841	39.74991699
2	2.403	59.24143262	2.662	54.25901563	4.197	34.078
3	2.507	56.83084277	2.589	55.64550195	2.895	52.77896777
4	2.104	67.63702539	2.547	56.58761523	2.912	52.46915918
5	2.087	68.23249023	2.735	52.5303916	3.006	49.5593457
6	2.068	68.83509766	2.605	55.23959961	2.708	56.29080176
7	2.125	66.99414844	2.41	59.68945898	3.159	48.36758789
8	2.11	67.43327734	2.426	59.34597949	3.312	46.16476758
9	2.159	66.03688574	2.463	58.34079004	3.21	47.65407031
10	2.205	64.56406445	2.535	56.82216895	3.599	42.40797949
Rata-rata	2.3273	62.64841768	2.564	56.26193711	3.2839	46.95205967

file2.mp3

file2.mp3	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	13.799	465.0570098	44.624	144.4383867	49.504	137.5567012
2	14.81	433.3750186	48.276	133.7097988	51.02	133.4053555
3	16.73	383.6588564	44.542	144.2131719	50.642	134.384835
4	16.724	383.7969678	45.957	140.3301836	48.179	141.3373242
5	14.649	437.7397578	44.525	144.5034297	49.761	136.6359746
6	16.804	381.9237852	47.748	135.0595488	49.195	138.7367217
7	13.619	471.4546914	44.825	143.9751064	50.434	135.2647256
8	15.022	427.2791348	44.673	143.5845107	49.476	137.52196
9	17.164	374.9368691	47.715	135.0677256	51.508	132.2654102
10	14.849	64.56406445	44.669	144.4999746	48.078	141.5725576
Rata-rata	15.417	382.3786155	45.7554	140.9381837	49.7797	136.8681565

file3.zip

file3.zip	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	2.763	188.1404121	5.118	102.4029482	6.179	89.38590039
2	2.763	188.239332	4.972	105.3625117	5.584	99.0469082
3	2.723	190.8982588	5.112	102.481124	5.576	99.09048633
4	2.814	184.7957256	5.154	101.6840439	8.391	65.9640127
5	2.981	174.3858721	5.308	99.19594922	6.64	83.31334082
6	2.768	187.9180869	5.007	104.376332	8.625	65.82404004
7	2.755	188.7223623	6.015	87.12336133	6.491	86.34003125
8	2.78	187.0835117	5.105	102.7539766	5.426	101.9190654
9	2.82	184.3794326	5.115	102.431499	5.67	97.34822754
10	2.808	64.56406445	4.984	105.2225332	6.428	86.01293262
Rata-rata	2.7975	173.9127059	5.189	101.3034279	6.501	87.42449453

file4.pdf

file4.pdf	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	6.615	309.6490693	16.108	127.4215781	17.523	124.3876846
2	8.22	249.159792	15.444	133.4559668	16.162	134.3262666
3	8.011	255.7103984	15.738	130.9761943	18.212	119.7132061
4	6.009	341.0422754	15.393	133.8886758	17.665	123.2197002
5	5.783	354.1066074	15.44	132.8360605	16.105	134.8744834
6	7.32	280.0855996	17.306	118.7079609	15.924	136.6903467
7	6.256	327.577124	15.347	133.5950811	17.115	127.0003125
8	6.945	295.239749	15.179	134.9902881	16.356	132.958166
9	8.937	229.1928164	15.368	134.1414404	16.271	133.5005078
10	7.984	64.56406445	15.323	133.8482305	16.905	128.8009775
Rata-rata	7.208	270.6327496	15.6646	131.3861477	16.8238	129.5471651

file5.iso

file5.iso	ipv4		ipv6		teredo	
No. Data	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)	latency (s)	throughput (KBytes/s)
1	111.689	499.2838477	380.775	146.9114697	420.081	140.926335
2	112.503	495.8179297	378.328	147.6132949	411.495	143.7816055
3	114.154	488.5899658	383.525	146.4662861	411.576	143.8186074
4	114.027	489.216125	379.144	147.7593359	408.698	144.7419473
5	113.571	491.095999	380.377	146.9477031	424.354	139.5200215
6	106.854	521.9950215	380.949	147.2658564	415.708	142.3567529
7	115.766	481.7951533	381.455	146.61404	413.003	143.2991865
8	118.755	469.5597061	381.439	146.7129385	409.965	144.2985508
9	111.423	500.3178086	379.672	147.1286025	423.128	139.9448838
10	109.959	64.56406445	388.463	144.3592607	498.427	118.6252998
Rata-rata	112.8701	450.2235621	381.4127	146.7778788	423.6435	140.131319