

**KOMPARASI UNJUK KERJA *FILE TRANSFER*  
*PROTOCOL* PADA JARINGAN *TEST-BED* IPv6 VPN  
TERHADAP TEREEDO DAN IPv4 MURNI**

**SKRIPSI**

Oleh

**MOHAMMAD ISA**  
**04 03 03 071 3**



**DEPARTEMEN TEKNIK ELEKTRO  
FAKULTAS TEKNIK UNIVERSITAS INDONESIA  
GANJIL 2007/2008**

**KOMPARASI UNJUK KERJA *FILE TRANSFER*  
*PROTOCOL* PADA JARINGAN *TEST-BED* IPv6 VPN  
TERHADAP TEREEDO DAN IPv4 MURNI**

**SKRIPSI**

Oleh

**MOHAMMAD ISA**  
**04 03 03 071 3**



**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:

### **KOMPARASI UNJUK KERJA *FILE TRANSFER* *PROTOCOL* PADA JARINGAN *TEST-BED IPv6 VPN* TERHADAP *TEREDO* DAN *IPv4 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 Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 7 Januari 2008

Mohammad Isa

NPM. 040303 0713

## **PENGESAHAN**

Skripsi dengan judul:

**KOMPARASI UNJUK KERJA *FILE TRANSFER*  
*PROTOCOL* PADA JARINGAN *TEST-BED* IPv6 VPN  
TERHADAP TEREEDO DAN IPv4 MURNI**

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

Depok, 7 Januari 2008

Dosen Pembimbing

Ir. A. Endang Sriningsih MT

NIP. 130 781 318

## UCAPAN TERIMA KASIH

Puji syukur kepada Allah SWT atas segala rahmat dan karunia-Nya sehingga skripsi ini dapat diselesaikan. Shalawat dan salam semoga senantiasa tercurahkan kepada Nabi Muhammad SAW. Penulis ingin mengucapkan terima kasih kepada Ibu:

**Ir. A. Endang Sriningsih MT**

selaku dosen pembimbing yang telah banyak meluangkan waktunya untuk memberikan saran, bimbingan, dan pengarahan sehingga skripsi ini dapat diselesaikan dengan baik.

Mohammad Isa  
NPM 04 03 03 071 3  
Departemen Teknik Elektro

Dosen Pembimbing  
Ir. A. Endang Sriningsih MT

**KOMPARASI UNJUK KERJA *FILE TRANSFER*  
PROTOCOL PADA JARINGAN *TEST-BED* IPv6 VPN  
TERHADAP TEREEDO DAN IPv4 MURNI**

**ABSTRAK**

Mekanisme transisi IPv6 diperlukan untuk menjamin interoperabilitas jaringan antara IPv6 dengan IPv4 selama masa migrasi. *Network Address Translation* (NAT) dan sifatnya pada sebuah jaringan akan menjadi hambatan tersendiri bagi sebagian besar metode transisi IPv6, atau dikenal dengan istilah *proto-41 forwarding*. Diperlukan metode transisi khusus yang dapat menembus NAT untuk memberikan konektivitas IPv6 melalui infrastruktur IPv4. Teredo dan IPv6 VPN, dimana keduanya menggunakan tunneling berbasis UDP, merupakan solusi untuk menjawab kebutuhan tersebut.

Skripsi ini dibuat dengan tujuan untuk mengetahui metode dengan performa yang lebih baik antara Teredo dan IPv6 VPN, serta sejauh mana perbedaannya terhadap jaringan IPv4 murni (*existing*). Untuk itu dilakukan beberapa pengujian menggunakan jaringan *test-bed* secara lokal. Pengujian meliputi koneksi TCP dan UDP untuk memberikan gambaran umum performa jaringan, serta koneksi FTP untuk memberikan gambaran khusus performa aplikasi jaringan. Parameter yang diamati selama pengujian adalah *throughput* TCP, *frame loss* dan *jitter* UDP serta *latency*, *transfer time*, dan *throughput* FTP.

Hasil pengujian menunjukkan bahwa IPv6 VPN memiliki performa yang lebih baik di seluruh parameter pengujian, dibandingkan Teredo. Perbedaan yang terjadi berkisar antara 286.18% - 458.64% untuk *throughput* TCP, 23.64% - 2088.69% untuk *jitter* UDP, 168.07% - 267.57% untuk *latency* FTP, 279.41% - 447.36% untuk *transfer time* FTP, dan 257.35% - 391.21% untuk *throughput* FTP. Selain itu, metode IPv6 VPN memiliki performa yang tidak jauh berbeda dengan jaringan IPv4 murni. Hal ini menjadikan metode IPv6 VPN sebagai alternatif yang lebih baik dibandingkan metode Teredo, dalam memberikan konektivitas IPv6 bagi jaringan *private* yang berada dibalik NAT melalui infrastruktur jaringan IPv4.

**Kata Kunci : IPv6, NAT, proto-41 forwarding, Teredo, IPv6 VPN**

Mohammad Isa  
NPM 04 03 03 071 3  
Departemen Teknik Elektro

Dosen Pembimbing  
Ir. A. Endang Sriningsih MT

**FILE TRANSFER PROTOCOL PERFORMANCE COMPARATION  
OF IPv6 VPN TOWARDS TEREDO AND NATIVE IPv4  
IN TEST-BED NETWORK**

**ABSTRACT**

Transition mechanism is required to guarantee IPv4 and IPv6 interoperability during the migration period. Network Address Translation (NAT) and its behaviour would become a drawback to some usual IPv6 transition mechanism, this problem also known as proto-41 forwarding. Specific transition mechanism is required to pass NAT and gave IPv6 connectivity through current IPv4 backbone infrastructure. Teredo and IPv6 VPN, both are based on UDP tunneling mechanism, could exceed this problem.

This paper was made to examine which methods, between Teredo and IPv6 VPN, would have better performance. This paper also made to examine the effect of each method compare to existing IPv4 network. To answer that, some testing that based on local test-bed have been done. The test includes TCP and UDP connection to give an illustration of general network performance, and FTP connection to give an illustration of spesific internet application performance. Parameters to watch during the test are TCP throughput, UDP frame loss and jitter, then FTP latency, transfer time and throughput.

Overall result from the test indicates that IPv6 VPN gave a better performance than Teredo at all testing parameters. The differences occurred around 286.18% - 458.64% for TCP throughput, 23.64% - 2088.69% for UDP jitter, 168.07% - 267.57% for FTP latency, 279.41% - 447.36% for FTP transfer time and 257.35% - 391.21% for FTP throughput. IPv6 VPN's performances also closed enough comparing to the existing native IPv4 network performances. From this paper and the test result, gave indication that IPv6 VPN would become a better solution than Teredo in relations to give IPv6 connectivity for a private network that stand behind NAT devices through IPv4 network infrastructure.

**Keywords : IPv6, NAT, proto-41 frowarding, Teredo, IPv6 VPN**

## DAFTAR ISI

|  |      |
|--|------|
| JUDUL .....  | i    |
| PERNYATAAN KEASLIAN SKRIPSI.....                   | ii   |
| PENGESAHAN .....                                   | iii  |
| UCAPAN TERIMA KASIH.....                           | iv   |
| ABSTRAK .....                                      | v    |
| DAFTAR ISI.....                                    | vii  |
| DAFTAR GAMBAR .....                                | ix   |
| DAFTAR TABEL.....                                  | x    |
| DAFTAR LAMPIRAN.....                               | xi   |
| DAFTAR SINGKATAN .....                             | xii  |
| DAFTAR ISTILAH .....                               | xiii |
| BAB I  |      |
| PENDAHULUAN .....                                  | 1    |
| 1.1. LATAR BELAKANG .....                          | 1    |
| 1.2. TUJUAN .....                                  | 2    |
| 1.3. PEMBATAAN MASALAH.....                        | 2    |
| 1.4. METODOLOGI PENELITIAN.....                    | 2    |
| 1.5. SISTEMATIKA PENULISAN.....                    | 2    |
| BAB II   |      |
| IPv6 SERTA MEKANISME TRANSISI MENUJU IPv6 .....    | 3    |
| 2.1. LATAR BELAKANG IPv6.....                      | 3    |
| 2.2. PERBANDINGAN IPv6 DENGAN IPv4 .....           | 4    |
| 2.3. SPESIFIKASI TEKNIS IPv6.....                  | 5    |
| 2.3.1. Format <i>Header</i> IPv6.....              | 5    |
| 2.3.2. Metode Pengalamatan IPv6.....               | 8    |
| 2.4. TRANSISI IPv4 MENUJU IPv6 .....               | 10   |
| 2.4.1. Mekanisme Transisi <i>Dual Stack</i> .....  | 11   |
| 2.4.2. Mekanisme Transisi <i>Tunneling</i> .....   | 11   |
| 2.4.3. Mekanisme Transisi <i>Translation</i> ..... | 12   |
| 2.5. <i>NETWORK ADDRESS TRANSLATION</i> (NAT)..... | 13   |
| 2.5.1. Cara Kerja NAT .....                        | 14   |
| 2.5.2. Jenis-Jenis NAT .....                       | 14   |
| 2.5.3. Isu NAT dalam Proses Transisi IPv6.....     | 16   |
| 2.6. TEREDO .....                                  | 16   |
| 2.6.1. Cara Kerja Teredo .....                     | 17   |
| 2.6.2. Format Alamat Teredo .....                  | 17   |
| 2.6.3. Komponen-Komponen Teredo.....               | 19   |
| 2.7. IPv6 <i>VIRTUAL PRIVATE NETWORK</i> .....     | 20   |
| 2.7.1. Cara Kerja VPN .....                        | 21   |
| 2.7.2. Jenis-Jenis VPN .....                       | 22   |
| 2.7.3. Komponen-Komponen VPN.....                  | 23   |
| 2.8. <i>FILE TRANSFER PROTOCOL</i> (FTP).....      | 24   |



|   |    |
|---|----|
| <b>BAB III</b>  |    |
| KONFIGURASI JARINGAN DAN METODE PENGAMBILAN DATA .....                            | 26 |
| 3.1. TOPOLOGI JARINGAN.....   | 26 |
| 3.2. KONFIGURASI JARINGAN.....  | 27 |
| 3.2.1. Konfigurasi Jaringan IPv4 Murni .....                                      | 27 |
| 3.2.2. Konfigurasi Jaringan IPv6 VPN .....  | 28 |
| 3.2.3. Konfigurasi Jaringan Teredo.....   | 29 |
| 3.3. PERANGKAT LUNAK YANG DIGUNAKAN .....   | 30 |
| 3.3.1. Iperf.....   | 30 |
| 3.3.2. OpenVPN .....  | 31 |
| 3.3.3. Xlight FTP.....  | 31 |
| 3.3.4. Smart FTP .....  | 32 |
| 3.3.5. Miredo .....   | 33 |
| 3.3.6. Wireshark .....  | 33 |
| 3.4. METODE PENGAMBILAN DATA.....   | 34 |
| 3.4.1. Pengujian Performa Jaringan Dengan Paket TCP Dan UDP .....                 | 34 |
| 3.4.2. Pengujian Performa Jaringan Pada Aplikasi <i>File Transfer Protocol</i> .. | 35 |
| <br>  |    |
| <b>BAB IV</b>   |    |
| ANALISA .....   | 36 |
| 4.1. ANALISA TOPOLOGI .....   | 36 |
| 4.2. ANALISA PERFORMA JARINGAN DENGAN TCP DAN UDP.....                            | 37 |
| 4.2.1. Analisa TCP : <i>Throughput</i> .....                                      | 38 |
| 4.2.2. Analisa UDP : <i>Frame Loss</i> .....                                      | 39 |
| 4.2.3. Analisa UDP : <i>Jitter</i> .....  | 41 |
| 4.3. ANALISA PERFORMA JARINGAN PADA APLIKASI FTP.....                             | 43 |
| 4.3.1. Analisa <i>Latency</i> .....   | 44 |
| 4.3.2. Analisa <i>Transfer Time</i> .....   | 46 |
| 4.3.3. Analisa <i>Throughput</i> .....  | 47 |
| 4.4. ANALISA KESELURUHAN.....   | 49 |
| <br>  |    |
| <b>BAB V</b>  |    |
| KESIMPULAN .....  | 51 |
| <br>  |    |
| DAFTAR ACUAN .....  | 52 |
| <br>  |    |
| DAFTAR PUSTAKA .....  | 53 |
| <br>  |    |
| LAMPIRAN.....   | 55 |

## DAFTAR GAMBAR

|  |    |
|--|----|
| <b>Gambar 2.1</b> Perbandingan format <i>header</i> pada protokol IPv6 dengan IPv4.....          | 5  |
| <b>Gambar 2.2</b> Posisi <i>extension header</i> pada protokol IPv6.....                         | 7  |
| <b>Gambar 2.3</b> NAT diantara jaringan lokal dan jaringan publik.....                           | 14 |
| <b>Gambar 2.4</b> Format paket Teredo yang terenkapsulasi paket UDP IPv4.....                    | 17 |
| <b>Gambar 2.5</b> Format alamat <i>client</i> Teredo.....  | 17 |
| <b>Gambar 2.6</b> Konfigurasi jaringan transisi Teredo beserta komponennya.....                  | 19 |
| <b>Gambar 2.7</b> Berbagai macam format paket yang terenkapsulasi VPN [7].....                   | 22 |
| <b>Gambar 2.8</b> Jaringan <i>remote access via intranet</i> [8].....                            | 22 |
| <b>Gambar 2.9</b> Jaringan <i>site-to-site via intranet</i> [8].....                             | 22 |
| <b>Gambar 2.10</b> Jaringan <i>remote access via internet</i> [8].....                           | 23 |
| <b>Gambar 2.11</b> Jaringan <i>site-to-site via internet</i> [8].....                            | 23 |
| <b>Gambar 3.1</b> Topologi umum jaringan <i>test-bed</i> pengujian.....                          | 26 |
| <b>Gambar 3.2</b> Topologi dan konfigurasi <i>test-bed</i> IPv4 murni.....                       | 28 |
| <b>Gambar 3.3</b> Topologi dan konfigurasi <i>test-bed</i> IPv6 VPN.....                         | 29 |
| <b>Gambar 3.4</b> Topologi dan konfigurasi <i>test-bed</i> Teredo.....                           | 29 |
| <b>Gambar 3.5</b> Tampilan program Iperf.....  | 30 |
| <b>Gambar 3.6</b> Tampilan OpenVPN saat aktif.....   | 31 |
| <b>Gambar 3.7</b> Tampilan Xlight FTP pada sisi <i>server</i> .....                              | 32 |
| <b>Gambar 3.8</b> Tampilan program Smart FTP pada sisi <i>client</i> .....                       | 32 |
| <b>Gambar 3.9</b> Tampilan program Wireshark saat <i>meng-capture</i> paket.....                 | 33 |
| <b>Gambar 4.1</b> Grafik garis perbandingan <i>throughput</i> TCP.....                           | 39 |
| <b>Gambar 4.2</b> Grafik garis perbandingan <i>frame loss</i> UDP.....                           | 40 |
| <b>Gambar 4.3</b> Grafik garis perbandingan <i>jitter</i> UDP.....                               | 42 |
| <b>Gambar 4.4</b> Grafik garis perbandingan <i>latency</i> FTP.....                              | 45 |
| <b>Gambar 4.5</b> Grafik garis perbandingan <i>transfer time</i> FTP.....                        | 47 |
| <b>Gambar 4.6</b> Grafik garis perbandingan <i>throughput</i> FTP.....                           | 48 |
| <b>Gambar 4.7</b> Statistik IPv6 VPN (kiri) dan IPv4 murni (kanan) untuk ukuran file 128 MB..... | 49 |

## DAFTAR TABEL

|  |    |
|--|----|
| <b>Tabel 4.1</b> Data <i>throughput</i> TCP .....    | 38 |
| <b>Tabel 4.2</b> Data <i>frame loss</i> UDP.....     | 40 |
| <b>Tabel 4.3</b> Data <i>jitter</i> UDP.....         | 42 |
| <b>Tabel 4.4</b> Data <i>latency</i> FTP .....       | 44 |
| <b>Tabel 4.5</b> Data <i>transfer time</i> FTP ..... | 46 |
| <b>Tabel 4.6</b> Data <i>throughput</i> FTP.....     | 48 |

## DAFTAR LAMPIRAN

|            |   |    |
|------------|---|----|
| LAMPIRAN 1 | : KONFIGURASI IPv4 MURNI.....             | 55 |
| LAMPIRAN 2 | : KONFIGURASI TEREDO.....                 | 56 |
| LAMPIRAN 3 | : KONFIGURASI IPv6 VPN.....               | 58 |
| LAMPIRAN 4 | : DATA HASIL UJI TOPOLOGI IPv4 MURNI..... | 60 |
| LAMPIRAN 5 | : DATA HASIL UJI TOPOLOGI TEREDO .....    | 62 |
| LAMPIRAN 6 | : DATA HASIL UJI TOPOLOGI IPv6 VPN .....  | 64 |

## DAFTAR SINGKATAN

|         |  |
|---------|--|
| DHCPv6  | Dynamic Host Configuration Protocol version 6  |
| FTP     | File Transfer Protocol                         |
| GUI     | Graphical User Interface                       |
| IETF    | Internet Engineering Task Force                |
| IP      | Internet Protocol                              |
| IPng    | Internet Protocol Next Generation              |
| IPv4    | Internet Protocol version 4                    |
| IPv6    | Internet Protocol version 6                    |
| ISATAP  | Intra-Site Automatic Tunnel Access Protocol    |
| LAN     | Local Area Network                             |
| NAT     | Network Address Translation                    |
| NAT-PT  | Network Address Translation - Port Translation |
| NGTrans | Next Generation Transition                     |
| NIC     | Network Interface Card                         |
| OS      | Operating System                               |
| OSI     | Open System Interconnection                    |
| PC      | Personal Computer                              |
| QoS     | Quality of Service                             |
| RFC     | Request For Comments                           |
| TCP     | Transmission Control Protocol                  |
| UDP     | User Datagram Protocol                         |
| VPN     | Virtual Private Network                        |

## DAFTAR ISTILAH

|               |  |
|---------------|--|
| Dekapsulasi   | Proses penghapusan <i>header</i> protokol tertentu pada sebuah paket.  |
| Dual Stack    | Mekanisme dimana IPv6 dan IPv4 dalam suatu perangkat jaringan aktif secara bersamaan.  |
| Enkapsulasi   | Proses penambahan <i>header</i> protokol tertentu pada sebuah paket.   |
| Frame Loss    | Parameter yang menunjukkan persentase jumlah paket yang hilang selama perjalanan terhadap keseluruhan paket yang ditransfer. |
| Header        | Bagian dari sebuah paket yang berisi informasi protokol dan informasi <i>routing</i> dari paket tersebut.                    |
| Jitter        | Parameter yang menunjukkan variasi <i>delay</i> paket-paket dari aliran data yang sama.                                      |
| Latency       | Parameter yang menunjukkan waktu yang dibutuhkan sebuah paket untuk tiba di terminal penerima.                               |
| Tunneling     | Mekanisme transmisi paket dimana setiap paket dienkapsulasi dengan <i>header</i> protokol lain.                              |
| Throughput    | Parameter yang menunjukkan besaran data rata-rata yang dapat dikirim melalui jaringan tiap satuan waktu.                     |
| Transfer Time | Parameter yang menunjukkan jumlah waktu yang dibutuhkan untuk mentransfer satu file utuh pada FTP.                           |

# BAB I

## PENDAHULUAN

### 1.1. LATAR BELAKANG

*Internet Protocol* atau disebut IP merupakan salah satu protokol yang banyak digunakan di dunia. Pada mulanya IP dikembangkan pada jaringan berskala kecil, dengan aplikasi-aplikasi sederhana seperti *mail transfer*. Namun, saat ini IP telah berkembang pesat baik dari skala jaringan maupun jenis aplikasi yang mampu dijalankan melalui internet. Saat ini telah banyak layanan-layanan seperti *e-mail*, *file transfer*, *audio-video streaming* bahkan *video conference* telah dapat dijalankan melalui internet. Tidak heran apabila saat ini jumlah pengguna internet terus meningkat di seluruh dunia. Protokol IP yang dirancang dengan mengadaptasi *OSI layer* menyebabkan protokol ini mudah diimplementasi dan memiliki interoperabilitas yang sangat baik.

Sebagian besar protokol IP yang digunakan saat ini mengikuti standar protokol IPv4. Seiring dengan meningkatnya pengguna internet, IPv4 diprediksi tidak akan mampu memenuhi kebutuhan alamat IP. Sehingga dibuatlah standar protokol baru yang disebut IPv6. Tujuan utama pengembangan protokol IPv6 adalah untuk memenuhi kebutuhan alamat IP untuk jangka panjang sekaligus menyempurnakan berbagai kelemahan yang ada pada IPv4. Hingga saat ini IPv6 yang mulai dikembangkan sejak tahun 1994 telah banyak digunakan di berbagai negara maju seperti Amerika Serikat, Jepang, Korea dan Eropa.

Salah satu isu utama yang banyak dikemukakan sejak awal munculnya IPv6 adalah interoperabilitasnya dengan jaringan IPv4. Keberadaan IPv6 tidak akan membuat IPv4 ditinggalkan secara serempak begitu saja. Adopsi IPv6 oleh seluruh jaringan internet dilakukan secara perlahan dan bertahap. Keberadaan mekanisme transisi menjamin interoperabilitas antara IPv6 dan IPv4 tersebut. Salah satu metode transisi IPv6 yang banyak dikenal adalah menggunakan metode transisi *tunneling* seperti 6to4, 6over4, dan ISATAP. Namun, keberadaan metode transisi *tunneling* yang sudah ada belum dapat menjawab semua tantangan. Salah satu hambatan dalam metode transisi *tunneling* adalah keberadaan *Network Address Translation* (NAT) dalam jaringan IPv4. Beberapa metode transisi

*tunneling* yang umum digunakan tidak dirancang untuk dapat menembus NAT. Oleh karenanya diperlukan suatu mekanisme transisi IPv6 yang dirancang untuk mengatasi permasalahan NAT tersebut. Beberapa solusi permasalahan NAT tersebut adalah Teredo dan IPv6 VPN.

## **1.2. TUJUAN**

Penulisan skripsi ini bertujuan untuk menguji serta membandingkan dua buah metode transisi yang dapat memberikan konektivitas IPv6 untuk jaringan *private* yang berada dibalik NAT, yaitu Teredo dan IPv6 VPN, khususnya untuk aplikasi *File Transfer Protocol*. Selain itu dilihat pula sejauh mana perbedaan performa kedua metode transisi tersebut dengan jaringan IPv4 murni.

## **1.3. PEMBATASAN MASALAH**

Permasalahan yang dibahas pada skripsi ini dibatasi pada pengujian dua buah metode transisi yang dapat memberikan konektivitas IPv6 untuk jaringan *private* yang berada di balik NAT, yaitu Teredo dan IPv6 VPN, pada aplikasi *File Transfer Protocol* serta sejauh mana perbedaannya dengan jaringan IPv4 murni.

## **1.4. METODOLOGI PENELITIAN**

Metode yang digunakan pada pengujian adalah dengan pengujian menggunakan jaringan lokal berskala kecil (*test-bed*). Parameter uji yang digunakan sebagai bahan perbandingan dan analisis adalah *throughput* (untuk pengujian TCP), *frame loss* dan *jitter* (untuk pengujian UDP) serta *throughput*, *latency* dan *transfer time* (untuk pengujian FTP).

## **1.5. SISTEMATIKA PENULISAN**

Penulisan skripsi dibagi menjadi 5 bab dengan pembagian sebagai berikut. Bab 1 berisi pendahuluan. Bab 2 berisi penjelasan dasar teori mengenai IPv6 beserta mekanisme transisinya. Bab 3 berisi penjelasan tentang topologi dan konfigurasi jaringan beserta metode pengujian dan pengambilan data. Bab 4 berisi hasil pengambilan data serta penjelasan analisis. Bab 5 berisi kesimpulan yang didapat dari hasil pengujian.



## **BAB II**

### **IPv6 SERTA MEKANISME TRANSISI MENUJU IPv6**

#### **2.1. LATAR BELAKANG IPv6**

IP (*Internet Protocol*) merupakan salah satu standar protokol yang paling dikenal dan berhasil diimplementasikan di dunia. Hal tersebut dapat dibuktikan dengan terus meningkatnya jumlah pengguna internet diseluruh dunia. Keberhasilan tersebut disebabkan antara lain karena protokol IP bersifat terbuka (*open platform*) sehingga mudah diadaptasi oleh berbagai platform teknologi lainnya.

Walaupun demikian protokol IP yang berlaku saat ini, yaitu IP versi 4 (IPv4), bukan tanpa kelemahan. Seiring dengan meningkatnya jumlah pengguna internet, dunia mulai kekurangan alamat IP. Alamat IP berfungsi sebagai identitas bagi tiap komputer yang terhubung dengan jaringan internet. Dengan demikian untuk menghindari kesalahan pengiriman paket IP maka duplikasi alamat IP tidak boleh terjadi. Oleh sebab itu alamat IP dikatakan bersifat unik untuk tiap-tiap komputer yang terhubung dengan jaringan internet. Masalahnya adalah standar IPv4 yang berlaku saat ini hanya mendukung pengalamatan (*addressing*) sebanyak 32 bit, sehingga jumlah alamat IPv4 yang dapat diberikan hanya sekitar  $2^{32}$  (= 4.296.967.296) alamat saja. Jumlah tersebut dirasa tidak mencukupi untuk memenuhi kebutuhan di masa yang akan datang, terlebih lagi dengan akan digunakannya IP sebagai basis teknologi pada jaringan *core platform* NGN (*Next Generation Network*).

Untuk menjawab tantangan kebutuhan tersebut, maka para ahli yang tergabung dalam sebuah organisasi bernama IETF (*Internet Engineering Task Force*) mempersiapkan sebuah standar protokol baru sebagai penerus IPv4. Langkah tersebut dimulai dengan merumuskan protokol IPng (*IP Next Generation*) pada periode awal 1990-an sebagai cikal bakal IPv6. Hingga akhirnya protokol IPv6 secara resmi ditetapkan menjadi salah satu standar IETF pada tanggal 10 Agustus 1998 melalui RFC 2460.

## 2.2. PERBANDINGAN IPv6 DENGAN IPv4

Sebagai evolusi dari IPv4, IPv6 memiliki berbagai kelebihan yang merupakan penyempurnaan dari IPv4. Berikut adalah beberapa kelebihan utama IPv6 bila dibandingkan dengan IPv4.

- Perluasan ruang alamat

Ruang alamat mengalami peningkatan dari 32 bit pada IPv4 menjadi 128 bit pada IPv6. Hal ini berarti IPv6 dapat menyediakan alamat  $2^{96}$  kali lebih banyak dibandingkan IPv4. Dengan demikian kebutuhan akan alamat IP di masa-masa mendatang dapat terpenuhi.

- Penyederhanaan format *header*

IPv6 memiliki ukuran *header* dua kali lebih besar dari *header* IPv4, terutama untuk mendukung ruang alamat 128 bit. Walaupun demikian format *headernya* mengalami penyederhanaan dengan menghilangkan beberapa *field* yang dirasa tidak efisien. Sebagai gantinya dibuatkan *header* tambahan yang disebut *extension header* sebagai opsi dalam pengiriman paket IPv6.

- Konfigurasi alamat *stateless* dan *statefull*

Konfigurasi alamat *host* pada IPv6 dibuat menjadi lebih fleksibel. Konfigurasi alamat *host* dapat dilakukan secara *statefull* dengan bantuan *server* DHCPv6 (konfigurasi manual) maupun secara *stateless* tanpa bantuan *server* DHCPv6 (konfigurasi otomatis). Pada konfigurasi *stateless*, *host* IPv6 mendapatkan informasi *prefix* dari router IPv6 di jaringan kemudian membuat sebuah alamat IPv6 identik yang disebut *link-local addresses*.

- Dukungan terhadap QoS (*Quality of Services*)

Keberadaan *field flow label* sebagai *field* baru pada *header* standar IPv6 menunjukkan dukungan IPv6 terhadap QoS. *Field flow label* digunakan untuk mengidentifikasi paket-paket yang membutuhkan perlakuan khusus dalam jaringan. Dengan dukungan terhadap QoS, IPv6 akan menjadi jaringan yang lebih *reliable* terutama untuk data-data yang bersifat *real time*.

- Dukungan keamanan yang lebih baik

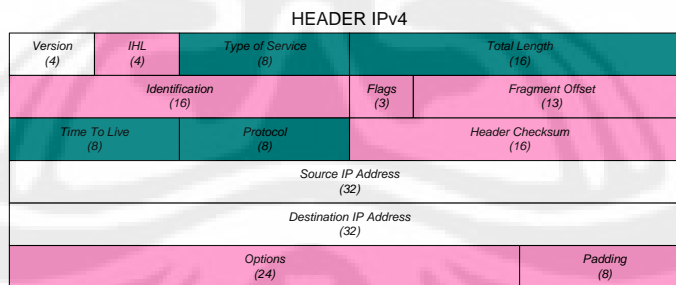
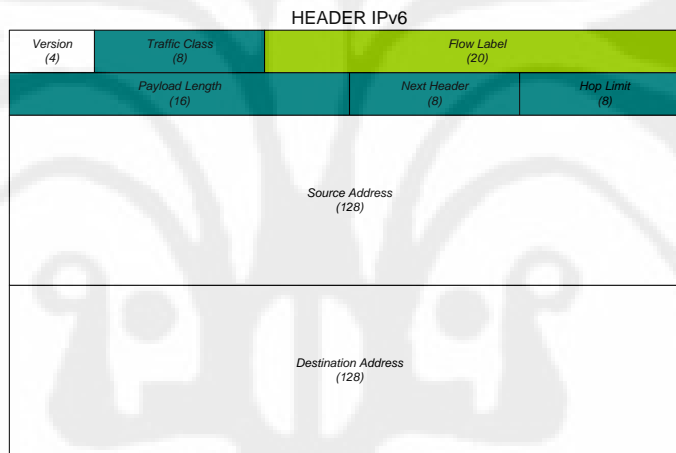
IPv6 telah dirancang dengan mendukung IPSec sehingga dapat dikatakan IPv6 memiliki keamanan yang lebih baik. Komponen IPSec yang terdapat pada IPv6 terdiri dari 2 *extension header* yaitu *authentication header* dan *Encapsulating*

Security Payload (ESP) header serta sebuah protokol Internet Key Exchange (IKE) yang digunakan dalam komunikasi *unicast*.

### 2.3. SPESIFIKASI TEKNIS IPv6

Pada dasarnya IPv6 memang dibuat untuk mengatasi masalah keterbatasan jumlah alamat IP yang tersedia pada IPv4. Namun demikian, perubahan yang diberikan oleh IPv6 tidak hanya pada peningkatan jumlah alamat IP saja. Berikut adalah beberapa detail perubahan yang diberikan pada IPv6.

#### 2.3.1. Format Header IPv6



**KETERANGAN**

- |   |   |
|---|---|
| <p><span style="display: inline-block; width: 15px; height: 15px; border: 1px solid black; background-color: #e0e0e0; margin-right: 5px;"></span> : Field yang dipertahankan pada IPv6</p> <p><span style="display: inline-block; width: 15px; height: 15px; border: 1px solid black; background-color: #008080; margin-right: 5px;"></span> : Field yang berubah nama / posisi pada IPv6</p> | <p><span style="display: inline-block; width: 15px; height: 15px; border: 1px solid black; background-color: #ff69b4; margin-right: 5px;"></span> : Field yang dihilangkan pada IPv6</p> <p><span style="display: inline-block; width: 15px; height: 15px; border: 1px solid black; background-color: #90ee90; margin-right: 5px;"></span> : Field baru pada IPv6</p> |
|---|---|

**Gambar 2.1** Perbandingan format *header* pada protokol IPv6 dengan IPv4

Secara sederhana *header* IPv6 dibagi menjadi 2 bagian, yaitu *header* standar (*default*) dan *header* tambahan (*extension*). *Header* standar (*default*)

adalah *field-field* yang selalu ada dalam setiap paket IPv6, sedangkan *header* tambahan (*extension*) adalah *field-field* IPv6 yang ditambahkan hanya saat diperlukan saja, letaknya berada diantara *header* standar IPv6 dengan *upper-layer header*. Berdasarkan Gambar 2.1 diatas, dapat dilihat *header* pada IPv6 memiliki ukuran yang lebih besar daripada *header* pada IPv4. Hal tersebut tentu saja untuk mendukung pengalamatan IPv6 yang mencapai 128 bit. Walaupun demikian, *header* IPv6 memiliki format yang lebih sederhana dibandingkan dengan *header* IPv4. Ini disebabkan adanya beberapa *field* pada *header* IPv4 yang dihilangkan pada *header* IPv6. *Field-field* pada IPv4 yang dihilangkan pada *header* IPv6 yaitu *Internet Header Length (IHL)*, *Identification*, *Flags*, *Fragment Offset*, *Header Checksum*, *Options*, serta *Padding*, sedangkan *field Version*, *Source Address*, dan *Destination Address* tetap dipertahankan pada IPv6. Pada *header* IPv6 sendiri diberikan *field* baru yang bernama *Flow Label*. Fungsi dari masing-masing *field* pada *header* standar IPv6 dijelaskan sebagai berikut.

- **Version** merupakan 4 bit *field* yang menunjukkan versi protokol IP yang digunakan (bernilai 6 bila menggunakan IPv6).
- **Traffic Class** merupakan 8 bit *field* yang berfungsi untuk menentukan skala prioritas antar paket atau mengidentifikasi paket-paket yang membutuhkan penanganan khusus. *Field* ini menggantikan *field Type of Service* pada IPv4.
- **Payload Length** merupakan 16 bit *field* yang berfungsi untuk menunjukkan panjang bit data (*payload*) yang dibawa oleh setiap paket IPv6. *Field* ini serupa dengan *field Total Length* pada IPv4, perbedaannya panjang bit *header* IPv6 tidak diikutsertakan dalam perhitungan.
- **Next Header** merupakan 8 bit *field* yang menunjukkan jenis protokol dari paket IPv6 tersebut. Selain itu *field* ini juga berfungsi untuk mengidentifikasi adanya *extension header* beserta jenisnya pada sebuah paket IPv6. *Field* ini serupa dengan *field Protocol Type* pada *header* IPv4.
- **Hop Limit** merupakan 8 bit *field* yang menunjukkan jumlah hop maksimum yang dapat dilewati paket tersebut sebelum di-*discard* dari jaringan. Apabila sebuah router (*hop*) menerima sebuah paket dengan nilai *hop limit* “1”, maka router tersebut akan mengurangi 1 nilai *field hop limit* menjadi bernilai “0”.

Karena *field hop limit* mencapai nilai “0” maka paket tersebut akan di-*discard* dan router mengirim pesan ICMPv6. *Field* ini serupa dengan *field Time To Live* (TTL) pada *header* IPv4.

- **Source Address** merupakan 128 bit *field* yang menunjukkan alamat IPv6 dari *node* asal paket.
- **Destination Address** merupakan 128 bit *field* yang menunjukkan alamat IPv6 dari *node* tujuan paket.
- **Flow Label** merupakan 20 bit *field* yang berfungsi mengidentifikasi paket-paket real-time yang membutuhkan perlakuan yang sama atau dianggap memiliki alur data yang sama. *Flow label* merupakan *field* baru yang ditambahkan pada *header* IPv6 dan sebelumnya tidak ada pada *header* IPv4.

*Extension header* pada IPv6 merupakan *header* yang menggantikan fungsi *field Option* pada *header* IPv4. *Extension header* merupakan *header* tambahan diluar dari *header* standar IPv6, artinya sebuah paket IPv6 bisa memiliki *extension header* bisa juga tidak. Paket yang tidak memiliki *extension header* akan diproses lebih cepat dibandingkan dengan paket yang memiliki *extension header*. Berbeda dengan IPv4 dimana *field Option* menjadi bagian dari format *header* standar IPv4 sehingga setiap *node* pada jaringan memproses paket lebih lama. Dengan demikian keberadaan *extension header* sebagai pilihan pada IPv6 dapat meningkatkan efisiensi proses *routing* jaringan IPv6. Posisi *extension header* pada IPv6 dapat dilihat pada Gambar 2.2 berikut.



**Gambar 2.2** Posisi *extension header* pada protokol IPv6

Setiap paket IPv6 dapat terdiri dari nol, satu ataupun beberapa *extension header* sekaligus. Berdasarkan RFC 2460 terdapat 6 jenis *extension header* IPv6, yaitu :

- **Hop-by-hop Options Header** digunakan untuk mengidentifikasi paket yang harus diproses disetiap router jaringan yang dilewati. *Hop-by-hop Options header* memiliki nilai *next header* = 0.

- **Destination Options Header** digunakan untuk memuat informasi tambahan untuk diproses pada *node* tujuan. Apabila *destination options header* muncul sebelum *routing header*, maka *header* tersebut harus diproses oleh router yang tercantum pada *routing header*. Apabila *destination options header* muncul sebelum *upper-layer header*, maka *header* tersebut harus diproses oleh *node* tujuan paket. *Destination options header* memiliki nilai *next header* = 60.
- **Routing Header** digunakan untuk mencantumkan satu atau lebih *node intermediate* yang harus dilewati paket sebelum sampai ke tujuannya. Atau dengan kata lain *header* ini dapat digunakan untuk menentukan jalur *routing* sebuah paket IPv6. *Routing header* memiliki nilai *next header* = 43.
- **Fragment Header** digunakan oleh *node* tujuan untuk mengidentifikasi apakah paket merupakan bagian dari suatu *fragment* atau tidak. Berbeda dengan IPv4, pada IPv6 router *intermediate* tidak diperbolehkan melakukan fragmentasi paket. Fragmentasi paket hanya dapat dilakukan oleh *node* pengirim setelah mengetahui ukuran maksimum MTU (*Maximum Transfer Unit*) yang dapat didukung jaringan sampai *node* yang dituju. *Fragment header* memiliki nilai *next header* = 44.
- **Authentication Header** digunakan untuk mengidentifikasi autentikasi, integritas data serta *anti-replay protection*. *Authentication header* memiliki nilai *next header* = 51.
- **Encapsulating Security Payload Header** digunakan untuk mengidentifikasi autentikasi, integritas data serta *anti-replay protection* khusus untuk paket yang dienkapsulasi. *Encapsulating Security Payload header* memiliki nilai *next header* = 50.

### 2.3.2. Metode Pengalamatan IPv6

Salah satu kelebihan yang dimiliki IPv6 adalah ruang alamat IP yang lebih besar dari pendahulunya, IPv4. Alamat IPv4 yang terdiri dari 32 bit hanya mampu menyediakan sebanyak  $2^{32}$  ( $\pm 4,3 \times 10^9$ ) alamat, sementara alamat IPv6 terdiri dari 128 bit sehingga mampu menyediakan sebanyak  $2^{128}$  ( $\pm 3,4 \times 10^{38}$ ) alamat. Jumlah tersebut  $2^{96}$  kali lebih banyak dari yang dapat disediakan oleh IPv4. Dengan

jumlah alamat IPv6 tersebut diharapkan dapat memenuhi kebutuhan akan alamat IP di masa depan.

Penulisan alamat IPv6 memiliki format yang berbeda dengan alamat IPv4. Alamat IPv4 terdiri atas 32 bit biner ditulis dalam 4 oktet masing-masing 8 bit, dimana antar oktet dipisahkan dengan notasi titik (.). Setiap oktet nantinya diterjemahkan menjadi bilangan desimal dengan nilai 0-255. Contoh penulisan alamat IPv4 adalah seperti berikut ini.

```
11000000.10101000.00000001.00001011
192 . 168 . 1 . 11
```

Alamat IPv6 terdiri atas 128 bit biner ditulis dalam 8 blok masing-masing 16 bit, dimana antar blok dipisahkan dengan notasi *colon* (:). Tiap blok nantinya diterjemahkan menjadi 4 bit bilangan heksadesimal dengan nilai antara 0000-FFFF. Untuk memudahkan penulisan angka “0” maka blok yang bernilai “0000” dapat dituliskan dengan sebuah “0” saja, sedangkan untuk beberapa blok yang bernilai 0 berurutan penulisannya dapat digantikan dengan notasi *dual colon* (::). Notasi *dual colon* hanya boleh dituliskan satu kali pada setiap alamat IPv6. Contoh penulisan alamat IPv6 adalah seperti berikut ini.

```
0010000111011010 0000000000000000 0000000000000000 0010111100111011
21DA : 0000 : 0000 : 2F3B
0000001010101010 0000000001111111 1111111100010100 0100111000101101
02AA : 00FF : FE28 : 9C5A
```

Bentuk tersebut dapat disederhanakan menjadi

```
21DA:0:0:2F3B:2AA:FF:FE28:9C5A
21DA:: 2F3B:2AA:FF:FE28:9C5A
```

Selain penulisan alamat yang berbeda dengan IPv4, metode pengalamatan pada IPv6 juga mengalami perubahan. Metode pengalamatan *broadcast* yang umum digunakan pada IPv4 tidak lagi digunakan pada IPv6. Metode *broadcast* mengirimkan duplikasi paket ke seluruh jaringan yang berada dalam satu *broadcast domain* atau disebut *broadcast storm*. Hal ini dapat menurunkan kinerja

jaringan secara keseluruhan sehingga dianggap tidak efisien. Oleh karenanya pada IPv6 digunakan 3 metode pengalamatan untuk kebutuhan yang berbeda yaitu *unicast*, *multicast* dan *anycast*.

- ***Unicast***

*Unicast* merupakan metode pengalamatan untuk *interface* tunggal. Paket yang dikirimkan ke suatu alamat *unicast* akan ditujukan ke sebuah *interface* yang diidentifikasi oleh alamat *unicast* tersebut. Alamat *unicast* biasanya digunakan dalam komunikasi *one-to-one* atau *peer-to-peer*.

- ***Multicast***

*Multicast* merupakan metode pengalamatan untuk beberapa *interface* sekaligus. Paket yang ditujukan ke suatu alamat *multicast* akan dikirimkan ke seluruh *interface* yang diidentifikasi oleh alamat *multicast* tersebut. Alamat *multicast* biasanya digunakan dalam komunikasi *one-to-many*.

- ***Anycast***

*Anycast* merupakan metode pengalamatan untuk beberapa *interface* sekaligus. Paket yang ditujukan ke suatu alamat *anycast* akan dikirimkan ke salah satu *interface* terdekat dari beberapa *interface* yang diidentifikasi oleh alamat *anycast* tersebut. Alamat *anycast* biasa digunakan dalam komunikasi *one-to-many* dan merupakan metode pengalamatan yang baru diperkenalkan pada IPv6.

#### **2.4. TRANSISI IPv4 MENUJU IPv6**

Migrasi jaringan IP secara keseluruhan dari IPv4 menuju IPv6 tidak dapat dilakukan dengan mudah. Jaringan internet yang umumnya berbasis IP saat ini memegang peranan krusial dalam komunikasi di dunia. Banyak sektor-sektor penting seperti bisnis, perdagangan, perbankan dan lain-lain yang bertumpu pada jaringan IP. Sehingga adanya perubahan dalam teknologi jaringan IP sedikit banyak akan membawa implikasi negatif yang tidak dapat ditoleransi. Selain itu, adopsi sebuah teknologi baru diatas teknologi yang sudah cukup matang tentu membutuhkan investasi yang tidak kecil. Hal tersebut tentu akan menjadi pertimbangan tersendiri bagi perusahaan-perusahaan besar. Oleh karenanya diperlukan mekanisme-mekanisme transisi IPv6 yang aman serta tidak mengganggu jaringan yang sudah ada saat ini. NGTrans, sebuah kelompok yang



bergerak di bidang internet menawarkan beberapa mekanisme transisi seperti diusulkan dalam RFC 4213. Mekanisme transisi yang ditawarkan tersebut adalah *dual stack*, *tunneling*, dan *translation*.

#### 2.4.1. Mekanisme Transisi *Dual Stack*

Pada metode transisi *dual stack*, tiap router dan *host* dalam jaringan harus mendukung baik protokol IPv4 maupun IPv6. Setiap *node* dalam jaringan masing-masing akan memiliki dua alamat yaitu alamat IPv4 dan alamat IPv6. Apabila *node* tersebut berkomunikasi dengan *host* IPv4 maka *node* tersebut akan menggunakan alamat IPv4-nya dan beroperasi seperti umumnya *node* IPv4. Selanjutnya bila *node* tersebut berkomunikasi dengan *node* IPv6 maka *node* tersebut akan menggunakan alamat IPv6-nya dan beroperasi seperti umumnya *node* IPv6. Untuk mendukung transisi *dual stack*, maka tiap router dalam jaringan harus mengaktifkan mekanisme *forwarding* untuk IPv4 dan IPv6. Termasuk didalamnya *upgrade* perangkat lunak yang mendukung kedua protokol serta konfigurasi jaringan (*routing* protokol, dll) harus dibuat untuk masing-masing IPv4 dan IPv6. Sehingga mekanisme ini berpotensi untuk menghabiskan lebih banyak memori dan daya pada peralatan jaringan. Selain itu karena tiap *node* juga harus memiliki alamat IPv4, maka mekanisme transisi ini tidak dapat memecahkan masalah keterbatasan alamat pada IPv4 [1].

#### 2.4.2. Mekanisme Transisi *Tunneling*

*Tunnel* adalah sebuah mekanisme enkapsulasi suatu protokol dengan protokol lainnya untuk dapat melewati jaringan yang belum dapat dilewati protokol tersebut secara normal. Metode transisi *tunneling* bekerja dengan menghubungkan 2 buah *node* IPv6 melalui jaringan IPv4 yang sudah ada. Dalam hal ini, jaringan IPv4 berperan sebagai perantara diantara kedua *host* IPv6 dengan membentuk semacam *virtual tunnel* di dalam jaringannya. Saat paket IPv6 yang dikirim tiba di router *ingress* dari jaringan IPv4, paket terlebih dulu dienkapsulasi ke dalam paket IPv4. Alamat asal dan tujuan paket yang tertulis dalam *header* IPv4 adalah alamat IPv4 dari router *ingress* (asal) dan router *egress* (tujuan), sedangkan *Field Protocol* pada *header* IPv4 diberi nilai 41 yang menunjukkan

adanya paket IPv6 terenkapsulasi. Selama berada dalam jaringan IPv4, proses *routing* paket dilakukan berdasarkan kedua alamat tersebut. Setibanya di router *outgress* dari jaringan IPv4, paket didekapsulasi menjadi paket IPv6 kembali. Karena router *ingress* dan *outgress* terletak diantara jaringan IPv6 dan IPv4 (*end-point*), maka kedua router tersebut harus berperan sebagai *dual stack node* atau dengan kata lain memiliki alamat IPv6 maupun IPv4 aktif.

Dalam metode transisi *tunneling* dikenal dua macam mekanisme *tunneling* yang berbeda, yaitu *tunneling* terkonfigurasi manual dan *tunneling* otomatis. Pada metode *tunneling* terkonfigurasi manual, proses *routing* paket IPv6 didalam jaringan IPv4 mulai dari router *ingress* hingga router *egress* telah ditentukan secara manual. Pada metode *tunneling* otomatis, proses *routing* paket IPv6 didalam jaringan dilakukan secara dinamis. Keduanya memiliki kelebihan masing-masing. Metode *tunneling* terkonfigurasi manual memberikan tingkat keamanan yang lebih baik, sedangkan metode *tunneling* otomatis memberikan kemudahan dari sisi administratif. Beberapa implementasi dari metode transisi *tunneling* yang umum dikenal antara lain adalah 6to4, 6over4, ISATAP, serta Teredo.

Kelebihan dari metode transisi ini adalah fleksibilitasnya sehingga dapat dengan cepat dan mudah diimplementasikan. Untuk menghubungkan antar *node* IPv6 tidak diperlukan infrastruktur *backbone* yang mendukung IPv6, karena jaringan *backbone* IPv4 dapat digunakan sebagai *tunnel*. Kekurangan metode transisi ini adalah memberikan beban komputasi lebih kepada router-router yang digunakan sebagai *ingress* dan *egress* sehingga mengkonsumsi lebih banyak daya serta berpotensi sebagai titik lemah jaringan [1].

#### **2.4.3. Mekanisme Transisi Translation**

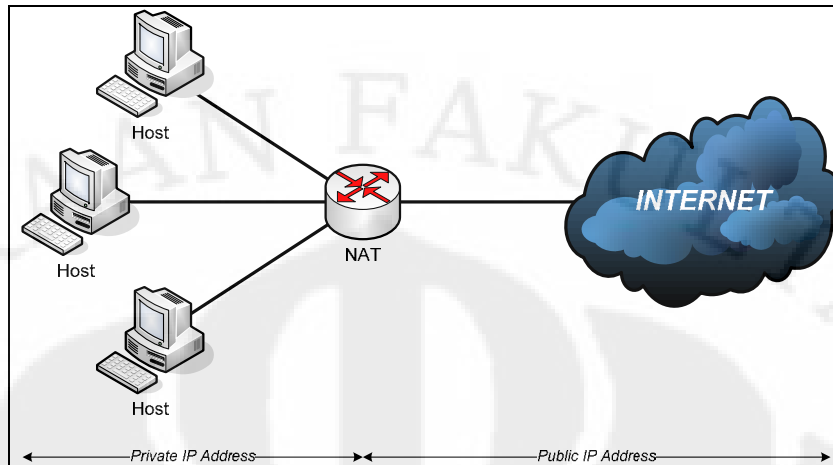
Selain metode transisi *dual-stack* dan *tunneling* dikenal juga metode transisi translasi. Pada metode transisi translasi, format *header* dan alamat dari paket yang dikirimkan akan mengalami perubahan. Fungsi pengubah format tersebut dijalankan oleh sebuah *node* yang berperan sebagai penerjemah yang letaknya diantara kedua *node* yang berkomunikasi. Saat *node* penerjemah menerima paket IPv4 dengan tujuan IPv6, maka penerjemah akan mentraslasikan

informasi dari *header* IPv4 ke *header* IPv6, menghilangkan *header* IPv4 dari paket kemudian menggantikannya dengan *header* IPv6 hasil translasi. Cara kerjanya menyerupai NAT (*Network Address Translation*) dalam jaringan IPv4, hanya saja metode transisi ini mampu bekerja diantara jaringan IPv4 dan IPv6. Salah satu implementasi dari mekanisme transisi translasi adalah NAT-PT (*Network Address Translation – Port Translation*) yang dispesifikasikan dalam RFC 2766.

Kelebihan dari metode transisi ini adalah secara langsung dapat menghubungkan jaringan IPv4 dengan jaringan IPv6, begitu pula sebaliknya. Kekurangan dari metode translasi ini adalah tidak dapat mengoptimalkan kelebihan IPv6 secara penuh seperti *end-to-end security*, tidak mendukung beberapa aplikasi yang membutuhkan pengalamatan langsung, serta berpotensi menjadi titik lemah dalam jaringan. Karena banyaknya keterbatasan metode translasi ini, maka metode ini sebaiknya digunakan sebagai pilihan terakhir [1].

## **2.5. NETWORK ADDRESS TRANSLATION (NAT)**

NAT (*Network Address Translation*) adalah teknik yang umum digunakan pada jaringan IPv4 untuk mengatasi keterbatasan alamat IPv4 publik. Tujuan dari NAT adalah agar suatu alamat IP (publik) dapat digunakan secara bersama-sama oleh beberapa *host* sekaligus. Alamat IP publik adalah alamat IP yang dikenal dan dapat digunakan secara global di internet. Jumlahnya terbatas dan alokasinya sudah diatur sedemikian rupa sehingga alamat IP publik dapat menjadi identitas dalam jaringan internet. Alamat IP *private* memiliki cakupan yang berbeda. Alamat IP *private* bebas digunakan oleh setiap *host* tetapi cakupannya hanya boleh terbatas dalam suatu jaringan lokal saja. Alamat IP *private* telah diberikan alokasi khusus yaitu pada 192.168.0.0/24, 172.16.0.0/16, 10.0.0.0/8. Ketiga subnet alamat IP *private* ini tidak dikenal dalam jaringan internet sehingga tidak dapat digunakan untuk berkomunikasi di jaringan internet. NAT memungkinkan translasi dari alamat IP *private* menjadi alamat IP publik sehingga *host* dalam jaringan lokal dapat terhubung dengan jaringan internet. Posisi NAT dalam jaringan dapat dilihat pada Gambar 2.3 berikut.



**Gambar 2.3** NAT diantara jaringan lokal dan jaringan publik

### 2.5.1. Cara Kerja NAT

NAT biasanya terletak diantara suatu jaringan lokal (dalam) dengan jaringan internet (luar) dan berfungsi sebagai *gateway* bagi jaringan lokal tersebut. Perbedaan antara NAT dengan sebuah *gateway* router atau *firewall* adalah dari kemampuannya menangani paket. NAT tidak hanya mem-*forward* atau men-*discard* paket, NAT memiliki opsi untuk mengubah informasi dalam *header* sebuah paket. NAT adalah sebuah penerjemah IP *header* atau lebih spesifik penerjemah alamat IP. Setiap paket memiliki informasi alamat IP asal serta alamat IP tujuan. Saat suatu paket masuk ke suatu perangkat NAT dari arah dalam menuju luar, maka NAT akan mencocokkan alamat IP asal paket dengan tabel translasinya. Apabila alamat IP asal terdapat dalam tabel translasi, maka NAT akan mengubah alamat IP asal paket (yang umumnya alamat IP *private*) menjadi alamat IP publik dari NAT, setelah itu paket tersebut di-*forward* ke jaringan internet. Apabila alamat yang dimaksud tidak terdapat dalam tabel translasi, maka paket akan di-*discard*. NAT juga bekerja dalam arah yang berlawanan, yaitu saat paket datang dari jaringan internet menuju jaringan lokal [2].

### 2.5.2. Jenis-Jenis NAT

Secara mekanisme pemetaannya, NAT dapat dibagi menjadi 2 jenis yaitu statik dan dinamik.

- **NAT Statik**

NAT statik terjadi ketika sebuah alamat lokal (*private*) dipetakan ke sebuah alamat global/internet (publik). Alamat lokal dan global dipetakan satu lawan satu secara statik [3].

- **NAT Dinamik**

NAT dinamis dibagi lagi menjadi dua, yaitu NAT *pool* dan NAT *overload*. NAT *pool* terjadi ketika beberapa kelompok alamat lokal (*private*) dipetakan ke beberapa kelompok alamat global (publik). NAT *overload* terjadi ketika beberapa kelompok alamat lokal (*private*) dipetakan ke sebuah alamat global (publik) [3].

Selain berdasarkan mekanisme pemetaannya, NAT juga dapat dibedakan berdasarkan caranya mem-*filter* dan mem-*forward* paket. Berdasarkan cara NAT mem-*filter* dan mem-*forward* paket, NAT dibagi menjadi *full cone*, *restricted cone*, *port restricted cone* serta *symmetric*.

- **Full Cone**

Mentranslasikan alamat dan *port* internal dari *host* yang berada di belakang perangkat NAT ke sebuah alamat dan *port* eksternal, jadi semua trafik yang berasal dari alamat di luar perangkat NAT akan dapat diteruskan ke *host* yang berada di belakang NAT [3].

- **Restricted Cone**

Mentranslasikan alamat dan *port* internal dari *host* yang berada di belakang perangkat NAT ke suatu alamat dan *port* eksternal. Alamat tujuan dari paket yang dikirim oleh *host* yang berada di belakang perangkat NAT akan disimpan dalam tabel NAT. Trafik yang berasal dari alamat di luar perangkat NAT hanya akan diteruskan apabila alamat tersebut terdapat di dalam tabel NAT [3].

- **Port Restricted Cone**

Tipe ini menambah larangan dalam penerimaan paket yang dikirim oleh *host* di jaringan eksternal. *Restricted Cone* NAT hanya mengamati *host* jaringan luar, akan tetapi *Port Restricted Cone* NAT juga mengamati *port* yang digunakan untuk dapat melalui NAT, paket yang dikirimkan oleh *host* dari jaringan luar tidak hanya harus dikirim dari *host* yang menjadi tujuan komunikasi yang dimulai oleh

*host* internal, tetapi juga harus dikirim melalui *port* yang menjadi tujuan komunikasi, jika tidak maka semua paket akan ditolak [3].

- **Symmetric**

Mentranslasikan sebuah alamat dan *port* internal yang sama ke suatu alamat eksternal dengan *port* yang berbeda-beda [3].

### 2.5.3. Isu NAT dalam Proses Transisi IPv6

Fungsi NAT yang dapat mentranslasikan alamat IP *private* menjadi alamat IP publik menjadikan NAT sebagai pilihan yang umum dalam jaringan IPv4. NAT telah banyak berperan dalam memberikan akses internet kepada *host-host* yang tidak mendapatkan alokasi alamat IP publik sekaligus berperan dalam menghemat penggunaan alamat IPv4 publik. Keberadaan NAT dalam proses transisi jaringan IPv4 menuju IPv6 tidak dapat diabaikan begitu saja. NAT yang juga berfungsi sebagai firewall umumnya dikonfigurasi untuk hanya melewatkan paket TCP dan UDP. *Header* IPv4 yang mengenkapsulasi paket IPv6 memiliki nilai *field* protokol 41 (IPv6), bukan TCP (=6) ataupun UDP (=17). NAT memberikan permasalahan tersendiri dalam transisi jaringan IPv6. Sebagian besar implementasi NAT yang ada saat ini belum mendukung translasi protokol 41 atau dikenal dengan istilah *proto-41 forwarding*, skenario dimana *tunneling* IPv6 bekerja. Hal ini menyebabkan beberapa mekanisme transisi *tunneling* seperti 6to4, 6over4 ataupun ISATAP tidak dapat menembus NAT.

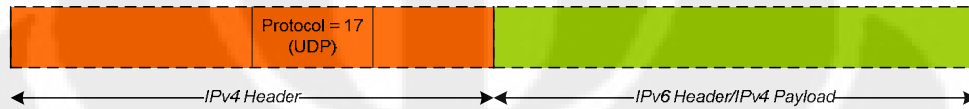
Hingga saat ini telah ada beberapa mekanisme transisi yang dirancang untuk dapat menembus NAT. Dua diantara mekanisme tersebut adalah Teredo dan IPv6 VPN.

### 2.6. TEREDO

Teredo merupakan salah satu implementasi metode transisi menggunakan *tunneling* otomatis selain 6to4, 6over4 dan ISATAP. Hal yang membedakan Teredo dari implementasi *tunneling* lainnya adalah Teredo dipergunakan khusus untuk skenario dimana *host* IPv4 berada di balik NAT. Teredo mengatasi masalah keterbatasan tersebut dengan mengenkapsulasi paket IPv6 dengan paket UDP IPv4 sehingga dapat menembus NAT.

### 2.6.1. Cara Kerja Teredo

Prinsip kerja Teredo adalah dengan mengenkapsulasi paket IPv6 kedalam paket UDP IPv4. Dengan mengenkapsulasi paket-paket IPv6 kedalam paket UDP IPv4, Teredo dapat melewati sebagian besar NAT, kecuali NAT simetris. Disebabkan NAT simetris mengalokasikan port secara dinamis dan terus berubah sehingga tidak dapat diprediksi oleh Teredo. Gambar 2.4 berikut adalah bentuk paket IPv6 yang terenkapsulasi paket UDP IPv4.

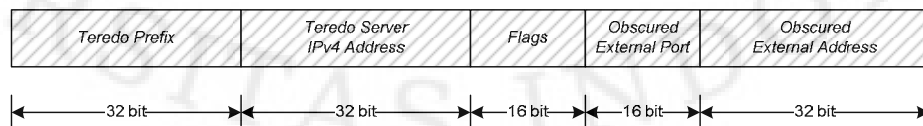


Gambar 2.4 Format paket Teredo yang terenkapsulasi paket UDP IPv4

Proses kerjanya dimulai ketika Teredo *client*, sebagai pihak yang menginisiasi koneksi, mengirimkan paket request (*router solicitation*) ke alamat IP publik Teredo *server*. Teredo *server* merespon dengan cara mengirim paket (*router advertisement*) serta melakukan proses yang disebut “kualifikasi” terhadap Teredo *client*. Proses “kualifikasi” tersebut untuk mengetahui jenis NAT yang terdapat didalam jaringan. Apabila Teredo *client* tidak berada dibalik NAT simetris, maka *client* dianggap memenuhi syarat. Selanjutnya *client* akan menyusun alamat IPv6 Teredo berdasarkan *router advertisement* yang diterimanya. Setelah mendapatkan alamat IPv6 Teredo, maka *client* dapat berkomunikasi dengan *client* IPv6 lainnya melalui Teredo *relay*.

### 2.6.2. Format Alamat Teredo

Teredo memiliki format alamat tertentu yang membuat setiap paket Teredo dikenali dalam jaringan. Format alamat teredo tersusun atas *field-field* seperti Gambar 2.5 berikut ini [4].



Gambar 2.5 Format alamat *client* Teredo

### ***Teredo Prefix***

*Field* ini merupakan 32 bit pertama dalam susunan alamat Teredo. Berdasarkan spesifikasi RFC 4380, IANA mengalokasikan 2001::/32 sebagai alamat *prefix* Teredo. *Prefix* ini sekaligus sebagai identitas bahwa paket IPv6 tersebut merupakan paket Teredo.

### ***Teredo Server IPv4 Address***

*Field* ini berisi alamat IPv4 publik dari Teredo *server* setelah dikonversi kedalam bentuk heksadesimal.

### ***Flags***

*Field* ini dipakai untuk mengidentifikasi tipe NAT yang terdapat didalam jaringan. Saat tipe NAT yang digunakan adalah *full cone*, maka *field* ini akan bernilai heksadesimal 8000. Selain dari *full cone*, maka *field* ini bernilai 0.

### ***Obscured External Port***

*Field* ini menunjukkan alamat port UDP yang dialokasikan oleh NAT setelah dilakukan operasi XOR dengan FFFF.

### ***Obscured External Address***

*Field* ini menunjukkan alamat IP publik dari NAT yang digunakan oleh Teredo *client* untuk berkomunikasi dengan Teredo *server* setelah dilakukan operasi XOR dengan FFFFFFFF.

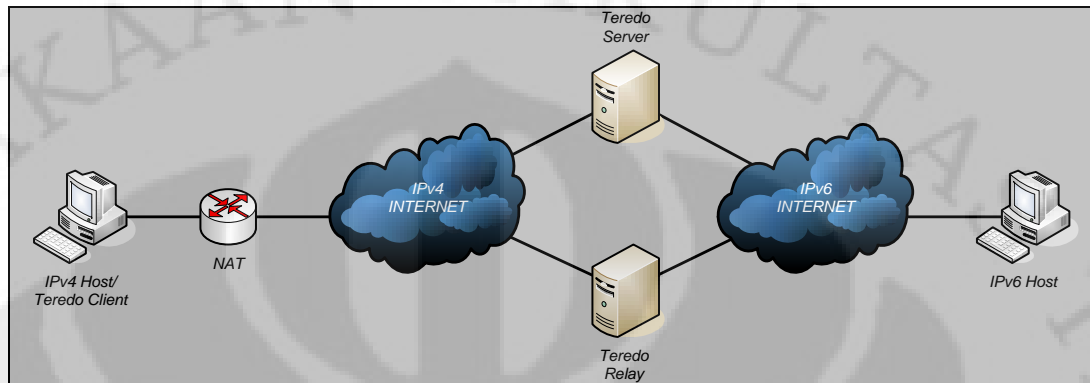
Sebagai contoh, apabila sebuah Teredo *client* (192.168.0.2) berkomunikasi dengan Teredo *server* (202.154.0.2) melalui alamat IP publik milik *full cone* NAT (167.205.0.2) port 2048.

|                            |                          |             |
|----------------------------|--------------------------|-------------|
| Teredo Server IPv4 Address | = 202.154.0.2            | → CA9A:0002 |
| Flags                      | = 8000                   | → 8000      |
| Obscured External Port     | = 2048 ⊕ FFFF            | → F7FF      |
| Obscured External Address  | = 167.205.0.2 ⊕ FFFFFFFF | → 5832:FFFD |

Sehingga alamat IPv6 yang didapatkan oleh Teredo *client* dari Teredo *server* adalah 2001::CA9A:0002:8000:F7FF:5832:FFFD.



### 2.6.3. Komponen-Komponen Teredo



**Gambar 2.6** Konfigurasi jaringan transisi Teredo beserta komponennya

Komponen-komponen jaringan utama yang bekerja dalam metode transisi Teredo adalah seperti ditunjukkan pada Gambar 2.6 diatas, yaitu :

- **IPv6 Host**

IPv6 *host* adalah komputer *host* yang terdapat didalam jaringan IPv6 murni (*native*) yang akan dituju oleh paket dari IPv4. Komputer ini hanya memiliki alamat IPv6 saja.

- **IPv4 Host / Teredo Client**

IPv4 *host* adalah komputer *host* didalam jaringan IPv4 murni (*native*) yang ingin terhubung dengan jaringan IPv6 sehingga disebut dengan Teredo *client*. Komputer ini hanya memiliki alamat IPv4 *private* saja.

- **NAT**

NAT adalah suatu perangkat jaringan baik berupa router maupun komputer yang berfungsi mentranslasikan alamat IPv4 *private* menjadi alamat IPv4 publik, begitu juga sebaliknya. NAT memiliki satu *interface* yang terhubung dengan jaringan LAN dengan alamat IP *private* dan satu *interface* lagi terhubung dengan internet dengan alamat IP publik.

- **Teredo Server**

Teredo *server* merupakan suatu perangkat jaringan khusus pada jaringan *tunneling* Teredo. Fungsi utamanya adalah untuk membantu Teredo *client* (IPv4 *host*) membentuk alamat IPv6-nya. Selain itu Teredo *server* juga berperan dalam proses

inisialisasi awal antara Teredo *client* dengan IPv6 *host*. Teredo *server* terletak diantara jaringan IPv4 dan jaringan IPv6. Teredo *server* memiliki 2 buah *interface* dimana sebuah *interface* terhubung dengan jaringan IPv4 dan sebuah *interface* lainnya terhubung dengan jaringan IPv6.

- **Teredo Relay**

Teredo *relay* merupakan suatu perangkat jaringan khusus pada jaringan *tunneling* Teredo selain Teredo *server*. Fungsi utamanya adalah untuk mem-*forward* paket-paket antara jaringan IPv4 dengan jaringan IPv6. Proses enkapsulasi dan dekapsulasi paket-paket juga menjadi tanggung jawab dari Teredo *relay*. Teredo *relay* mengenkapsulasi paket-paket IPv6 dari *host* IPv6 yang ditujukan kepada Teredo *client* kedalam paket UDP IPv4. Sebaliknya untuk paket-paket yang berasal dari Teredo *client* menuju *host* IPv6, Teredo *relay* mendekapsulasi paket UDP IPv4 tersebut menjadi paket IPv6. Sama seperti Teredo *server*, Teredo *relay* juga memiliki 2 buah *interface* yang terhubung dengan jaringan IPv4 dan jaringan IPv6.

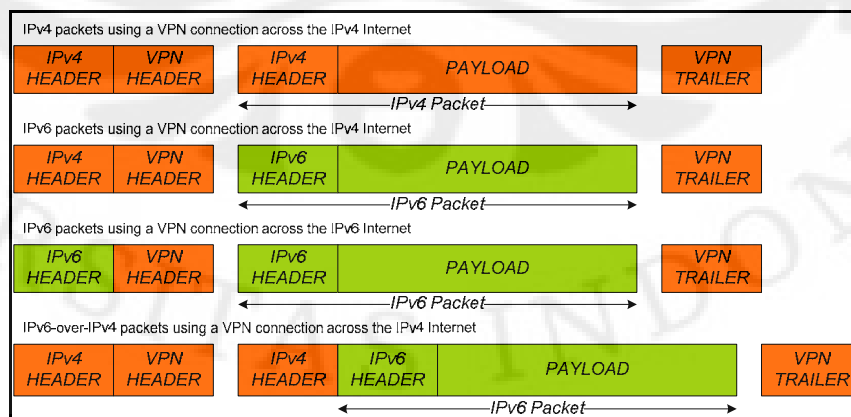
## 2.7. IPv6 VIRTUAL PRIVATE NETWORK

*Virtual Private Network* (VPN) adalah suatu bentuk jaringan atau layanan yang mampu menghasilkan kondisi jaringan lokal *private* melalui infrastruktur jaringan internet publik [5]. VPN umumnya digunakan oleh suatu organisasi untuk menghubungkan jaringan komputer antar lokasi atau jaringan lokal dengan jaringan *remote*-nya. Dalam mengirimkan data melalui jaringan internet publik tersebut VPN menggunakan semacam *virtual tunnel* untuk menghubungkan kedua lokasi. Saat kedua jaringan telah terkoneksi melalui VPN, maka tiap komputer yang terhubung dengan jaringan tersebut akan dianggap sebagai komputer lokal sehingga semua aplikasi dalam jaringan dapat diakses oleh komputer tersebut. IPv6 VPN bukanlah suatu solusi yang dirancang khusus dalam mekanisme transisi IPv6 seperti ISATAP, 6to4 atau Teredo. IPv6 VPN lebih bisa dikatakan sebagai pemanfaatan solusi VPN untuk mengirimkan data IPv6 melalui infrastruktur existing (IPv4) [6]. Khususnya bila dikaitkan dengan Teredo, IPv6 VPN memiliki kemampuan yang sama untuk membawa data IPv6 menembus NAT.

### 2.7.1. Cara Kerja VPN

Proses VPN dimulai dengan pembuatan virtual *tunnel* diantara dua lokasi yang ingin dihubungkan. *Virtual tunnel* ini bersifat *point-to-point*, dan tiap *tunnel end-point* harus didefinisikan secara jelas pada VPN. Hubungan koneksi antara kedua *tunnel end-point* dapat berupa *peer-to-peer* ataupun *client-server*. Pada koneksi *peer-to-peer* kedua *node* yang terlibat memiliki hak yang sama, keduanya sama-sama menginisiasi koneksi VPN. Pada koneksi *client-server* salah satu *node* bertindak sebagai penginisiasi koneksi (*client*) dan *node* lainnya bertindak menunggu koneksi (*server*). Apabila koneksi antar kedua *tunnel end-point* telah terjadi, maka *virtual tunnel* telah terbentuk dan kedua jaringan terhubung.

Serupa dengan mekanisme *tunneling* pada umumnya, VPN juga menggunakan proses enkapsulasi dalam mengirimkan data. Data yang akan dikirim awalnya akan dienkapsulasi dengan *header* VPN. Selanjutnya data yang telah terenkapsulasi *header* VPN akan dienkapsulasi kembali dengan *header* protokol yang akan digunakan selama pengiriman. Dalam hal ini VPN memiliki format *header* sendiri yang digunakan untuk menyimpan informasi *tunnel* VPN. Paket yang dikirimkan melalui koneksi VPN dapat diamankan melalui proses enkripsi. Paket yang dikirimkan tidak dapat didekripsi pada *tunnel end-point* apabila *password* diantara keduanya tidak sesuai. Karena VPN memberikan porsi tersendiri untuk metode enkripsi, maka keseluruhan *header* VPN akan memiliki ukuran yang lebih besar dibandingkan paket aslinya. Kelebihan ukuran inilah yang pada *traffic* jaringan disebut sebagai *overhead*.



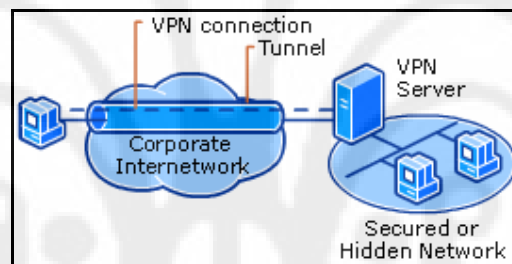
Gambar 2.7 Berbagai macam format paket yang terenkapsulasi VPN [7]

### 2.7.2. Jenis-Jenis VPN

VPN dapat digunakan untuk menghubungkan dua jaringan yang terpisah baik melalui infrastruktur intranet maupun internet. Berdasarkan karakteristik tersebut, VPN dapat dibagi menjadi 4 macam yaitu *remote access via intranet*, *site-to-site access via intranet*, *remote access via internet* serta *site-to-site via internet*.

- **Remote Access via Intranet**

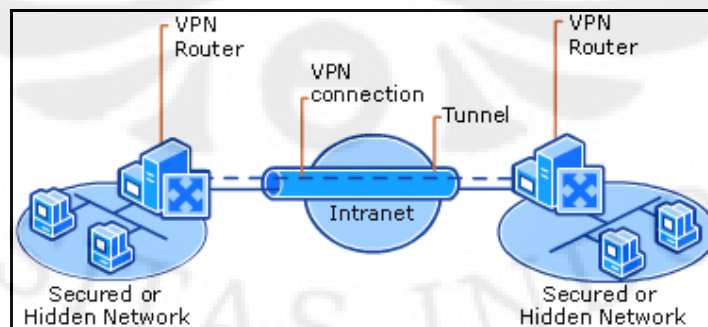
Pada jenis koneksi ini VPN memberikan akses kepada *client* agar mendapatkan akses ke jaringan yang berbeda segmen melalui jaringan intranet. Jaringan *remote access* VPN melalui intranet dapat dilihat pada Gambar 2.8 berikut.



Gambar 2.8 Jaringan *remote access via intranet* [8]

- **Site-to-site Access via Intranet**

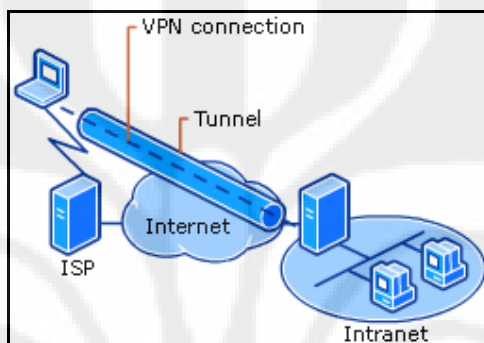
Pada jenis koneksi ini VPN menghubungkan dua buah jaringan yang berbeda segmen agar dapat mengakses satu sama lain melalui jaringan intranet. Jaringan *site-to-site* VPN melalui intranet dapat dilihat pada Gambar 2.9 berikut.



Gambar 2.9 Jaringan *site-to-site via intranet* [8]

- **Remote Access via Internet**

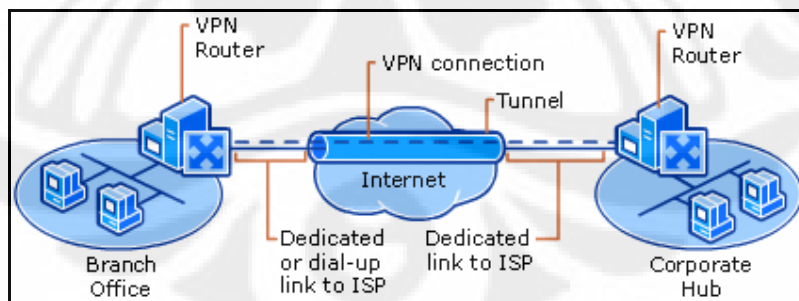
Pada jenis koneksi ini VPN memberikan akses kepada *client* agar mendapatkan akses ke suatu jaringan *remote* yang berbeda lokasi melalui jaringan intranet. *Client* bisa mendapatkan akses internet melalui ISP lokal sebelum melakukan koneksi ke *server* VPN. Jaringan *remote access* VPN melalui internet dapat dilihat pada Gambar 2.10 berikut.



Gambar 2.10 Jaringan *remote access* via internet [8]

- **Site-to-site via Internet**

Pada jenis koneksi ini VPN menghubungkan dua jaringan yang berbeda lokasi agar dapat mengakses satu sama lain melalui jaringan intranet. Jaringan *site-to-site* VPN melalui internet dapat dilihat pada Gambar 2.11 berikut.



Gambar 2.11 Jaringan *site-to-site* via internet [8]

### 2.7.3. Komponen-Komponen VPN

- **VPN Server**

VPN *server* adalah *tunnel end-point* yang berfungsi untuk menunggu adanya permintaan koneksi VPN. VPN *server* dapat berupa router dalam jenis koneksi site-to-site ataupun *host* dalam koneksi *host-to-host*. VPN *server* bertugas melakukan identifikasi dan autentikasi terhadap permintaan koneksi VPN yang masuk.

- **VPN Client**

VPN *client* adalah *tunnel end-point* yang berfungsi untuk menginisiasi koneksi VPN ke VPN *server* yang dituju. VPN *client* dapat berupa router dalam jenis koneksi site-to-site ataupun *host* dalam koneksi *host-to-host*. Saat melakukan inisiasi, VPN *client* akan memperkenalkan diri terlebih dulu kepada VPN *server*.

## **2.8. FILE TRANSFER PROTOCOL (FTP)**

Semakin maraknya penggunaan internet mengakibatkan makin berkembangnya aplikasi-aplikasi jaringan baru. Saat ini internet tidak lagi hanya digunakan untuk pertukaran informasi melalui halaman web ataupun email saja. Internet juga telah berkembang sebagai media pertukaran file, *streaming* audio dan video, maupun *video conference*. *File Server* merupakan salah satu aplikasi paling populer di internet saat ini selain *Web Server* dan *Mail Server*. Hal ini dapat dibuktikan dengan makin banyaknya domain-domain di internet yang menyediakan fasilitas *uploading* dan *downloading* file, seperti rapidshare. *File Server* bekerja dengan memanfaatkan *File Transfer Protocol* (FTP). FTP digunakan dalam proses pengiriman/transfer file baik upload maupun download melalui jaringan TCP/IP.

Proses pengiriman file melalui FTP melibatkan dua *host* dimana satu *host* berperan sebagai *client* dan satu *host* lainnya berperan sebagai *server*. FTP *client* merupakan *host* yang mengirim permintaan *download/upload* ke FTP *server*. Setelah menerima permintaan tersebut, FTP *server* akan mengautentikasi user. Apabila diterima maka koneksi FTP akan terjadi dan selanjutnya *client* dapat mengakses penyimpanan data yang disiapkan oleh *server*. Hak-hak *client* sangat ditentukan oleh *privilege* yang diberikan oleh *server* seperti *download*, *upload*, bahkan merubah struktur data.

Berdasarkan autentikasi atau tingkat keamanannya, FTP dapat dibagi menjadi dua modus yaitu *anonymous* dan *authenticated*. Pada modus *anonymous*, FTP server dapat diakses oleh setiap *client* di jaringan tanpa harus memiliki ID serta *password*. Sementara pada modus *authenticated*, FTP server hanya dapat diakses oleh *client* yang memiliki ID serta *password* yang terdaftar didalam database dari FTP server.



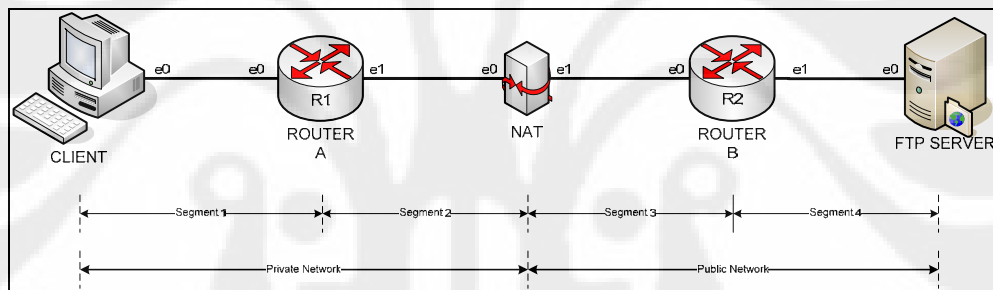
# BAB III

## KONFIGURASI JARINGAN

### DAN METODE PENGAMBILAN DATA

#### 3.1. TOPOLOGI JARINGAN

Jaringan *test-bed* yang digunakan merupakan simulasi jaringan yang terdiri dari 5 buah komputer untuk jaringan IPv4 murni dan jaringan IPv6 VPN serta 7 komputer untuk jaringan Teredo. Komputer Router A, Router B dan NAT beroperasi dibawah *platform* Linux Ubuntu LTS 6.06 sedangkan komputer *Client* dan *FTP Server* beroperasi dibawah *platform* Windows XP Service Pack 2. Bentuk umum topologi jaringan *test-bed* yang digunakan untuk pengujian ditunjukkan oleh Gambar 3.1.



Gambar 3.1 Topologi umum jaringan *test-bed* pengujian

Spesifikasi perangkat keras yang digunakan untuk jaringan *test-bed* adalah sebagai berikut :

- *Client*
  - Processor : AMD Athlon XP 1.5 GHz
  - RAM : 256 Mbytes DDR
  - NIC : Ethernet 10/100 Mbps
  - OS : Windows XP Service Pack 2
- *FTP Server*
  - Processor : Intel Pentium IV 2.26 GHz
  - RAM : 512 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Windows XP Service Pack 2



- Router 1
  - Processor : Intel Pentium IV 1.7 GHz
  - RAM : 256 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Linux Ubuntu LTS 6.06
- Router 2
  - Processor : Intel Pentium 2.4 GHz
  - RAM : 512 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Linux Ubuntu LTS 6.06
- NAT
  - Processor : Pentium III 500 MHz
  - RAM : 128 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Linux Ubuntu LTS 6.06

Perangkat tambahan untuk *test-bed* Teredo

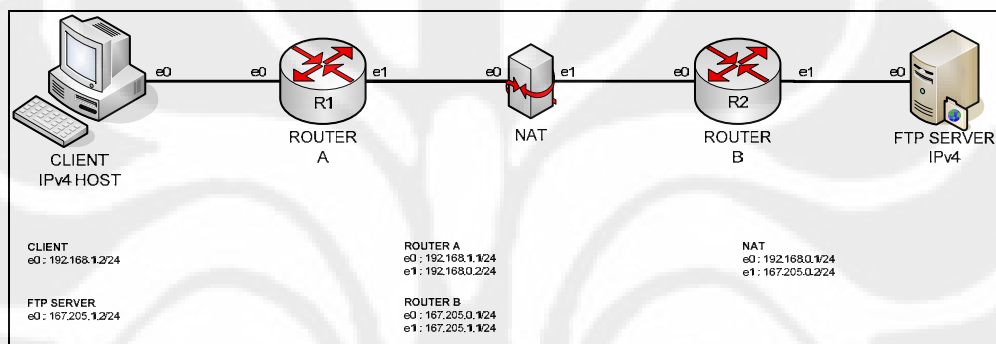
- Teredo *Server*
  - Processor : Intel Pentium IV 2.00 GHz
  - RAM : 256 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Linux Ubuntu LTS 6.06
- Teredo *Relay*
  - Processor : Pentium IV 1.7 GHz
  - RAM : 384 Mbytes
  - NIC : Ethernet 10/100 Mbps
  - OS : Linux Ubuntu LTS 6.06

## 3.2. KONFIGURASI JARINGAN

### 3.2.1. Konfigurasi Jaringan IPv4 Murni

Jaringan *test-bed* IPv4 murni tersusun dari 5 komputer yang keseluruhannya diberi alamat IPv4. Pemberian alamat IPv4 pada tiap *interface*

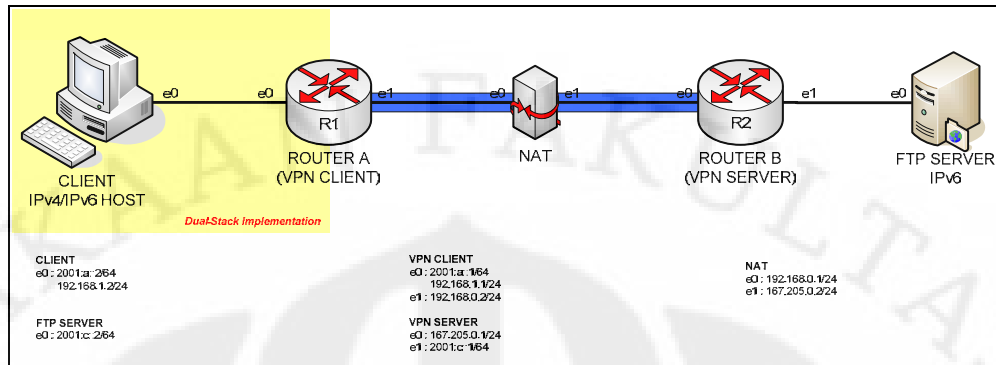
dilakukan secara statik yang berarti alamat IPv4 dimasukkan secara manual. Untuk perangkat jaringan yang berada dibalik Router A diberikan alamat IPv4 *private* yaitu 192.168.1.0/24, sedangkan untuk perangkat yang berada dibalik Router B diberikan alamat IPv4 publik yaitu 167.205.1.0/24. Perangkat NAT diberikan alamat IP *private* untuk NIC yang terhubung dengan jaringan *private* dan alamat IP publik untuk NIC yang terhubung dengan jaringan publik. Topologi serta konfigurasi alamat untuk jaringan IPv4 murni dapat dilihat pada Gambar 3.2 dan Lampiran 1.



Gambar 3.2 Topologi dan konfigurasi *test-bed* IPv4 murni

### 3.2.2. Konfigurasi Jaringan IPv6 VPN

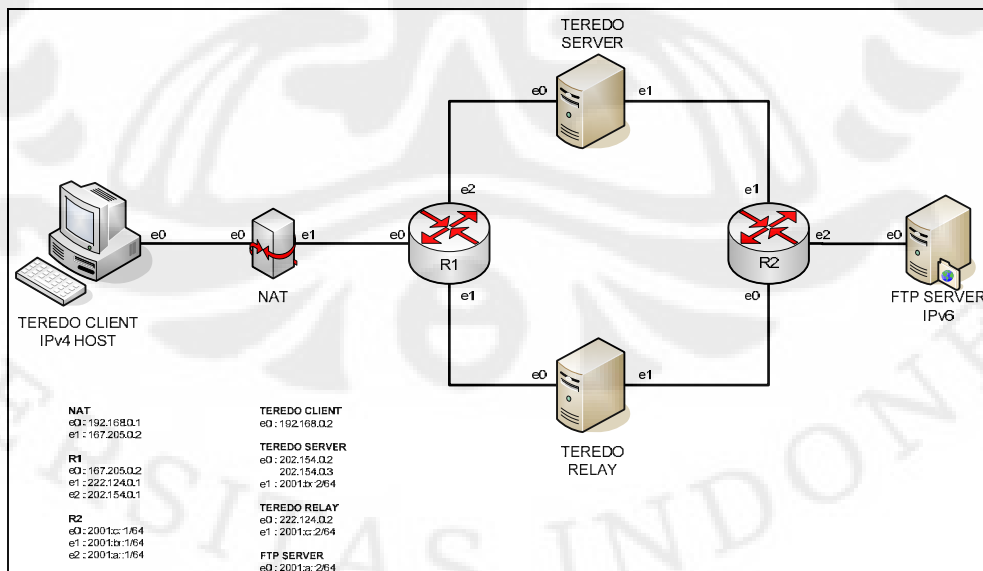
Jaringan *test-bed* IPv6 VPN tersusun dari 5 komputer dengan bentuk topologi yang serupa dengan *test-bed* IPv4 murni. Pemberian alamat IPv4 dan IPv6 pada tiap *interface* dilakukan secara statik dimana alamat IP dimasukkan secara manual. Perangkat jaringan yang berada dibalik Router A dikonfigurasi *dual stack* agar jaringan *private* IPv4 yang sudah ada tetap dapat digunakan. Alamat IP yang diberikan untuk segmen jaringan ini adalah 192.168.0.0/24 untuk IPv4 dan 2001:a::/64 untuk IPv6. Perangkat jaringan yang berada dibalik Router B merupakan jaringan IPv6 murni dengan alokasi alamat IPv6 2001:c::/64. Perangkat lunak VPN yang digunakan yaitu OpenVPN. Sebagai *tunnel end-point* adalah Router A (VPN *client*) dan Router B (VPN *server*), dengan alokasi alamat untuk virtual *tunnel* adalah 2001:b::/64. Perangkat NAT diberikan alamat IP *private* untuk NIC yang terhubung dengan jaringan *private* dan alamat IP publik untuk NIC yang terhubung dengan jaringan publik. Topologi serta konfigurasi alamat untuk jaringan IPv6 VPN dapat dilihat pada Gambar 3.3 dan Lampiran 3.



Gambar 3.3 Topologi dan konfigurasi *test-bed* IPv6 VPN

### 3.2.3. Konfigurasi Jaringan Teredo

Jaringan Teredo digunakan sebagai pembanding terhadap solusi IPv6 VPN yang digunakan untuk jaringan *private* yang berada di balik NAT. Jaringan *test-bed* Teredo tersusun dari 7 buah komputer dengan tambahan 2 komputer dari kedua topologi sebelumnya sebagai Teredo *server* dan Teredo *relay*. Pemberian alamat IPv4 dan IPv6 pada tiap *interface* dilakukan secara statik atau manual, kecuali alamat IPv6 pada Teredo *client* yang akan diberikan secara otomatis oleh Teredo *server*. *Prefix* IPv6 untuk Teredo *client* dialokasikan 2001::/32, sesuai dengan spesifikasi Teredo pada RFC 4380. Topologi serta konfigurasi alamat untuk jaringan teredo dapat dilihat pada Gambar 3.4 dan Lampiran 2.



Gambar 3.4 Topologi dan konfigurasi *test-bed* Teredo

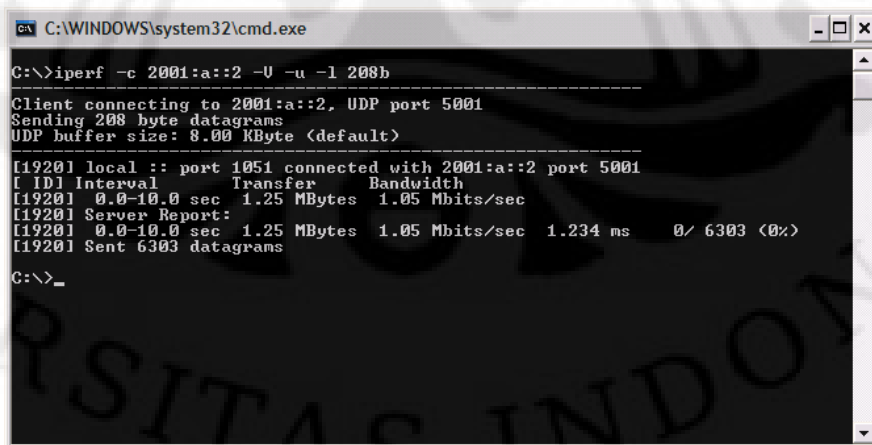
### 3.3. PERANGKAT LUNAK YANG DIGUNAKAN

Sistem operasi yang digunakan pada pengujian menggunakan *test-bed* adalah Windows XP SP2 dan Linux Ubuntu 6.06. Sistem operasi Linux Ubuntu 6.06 dipergunakan khusus untuk perangkat-perangkat jaringan selain *host* dan *server*. Pertimbangan penggunaan sistem operasi Linux adalah karena sistem operasi ini dikenal dengan kehandalan dan kestabilannya untuk bekerja nonstop sehingga cocok digunakan untuk perangkat jaringan seperti router. Sistem operasi Windows XP SP2 dipergunakan pada *host* dan *server*. Pertimbangan penggunaan Windows XP adalah sistem operasi ini paling banyak digunakan oleh kebanyakan user serta memiliki banyak aplikasi pendukung yang sudah stabil.

Selain sistem operasi, digunakan pula beberapa perangkat lunak khusus untuk mendukung skenario topologi yang digunakan. Berikut adalah perangkat-perangkat lunak pendukung yang digunakan pada pengujian menggunakan *test-bed*.

#### 3.3.1. Iperf

Iperf adalah program yang berfungsi untuk menghasilkan paket secara otomatis. Paket yang dapat dihasilkan oleh Iperf adalah paket TCP atau UDP. Program Iperf dijalankan pada ujung-ujung jaringan yang akan diukur performanya. Karena belum memiliki GUI, maka pengaturan parameter dilakukan melalui command prompt. Tampilan Iperf ditunjukkan pada Gambar 3.5.

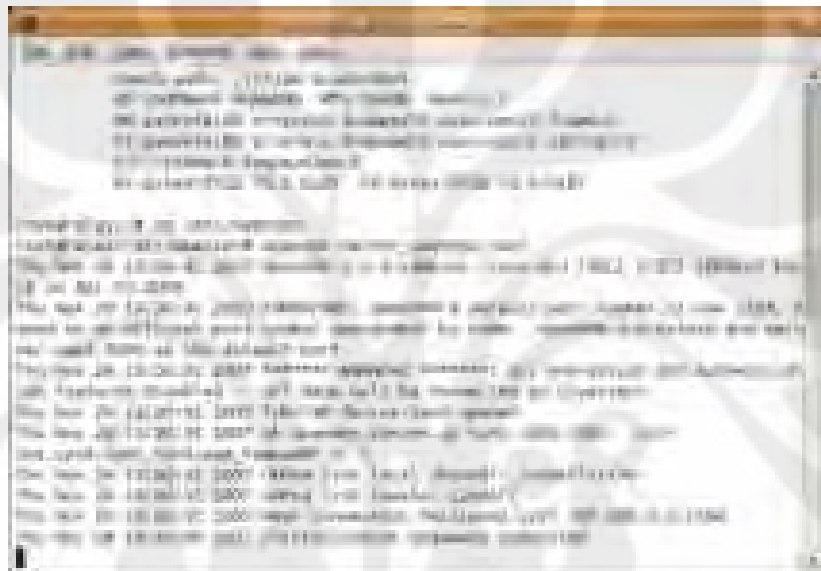


```
C:\WINDOWS\system32\cmd.exe
C:\>iperf -c 2001:a::2 -U -u -l 200b
-----
Client connecting to 2001:a::2, UDP port 5001
Sending 208 byte datagrams
UDP buffer size: 8.00 KByte (default)
-----
[1920] local :: port 1051 connected with 2001:a::2 port 5001
[ ID] Interval      Transfer      Bandwidth
[1920] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[1920] Server Report:
[1920] 0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  1.234 ms   0/ 6303 (0%)
[1920] Sent 6303 datagrams
C:\>_
```

Gambar 3.5 Tampilan program Iperf

### 3.3.2. OpenVPN

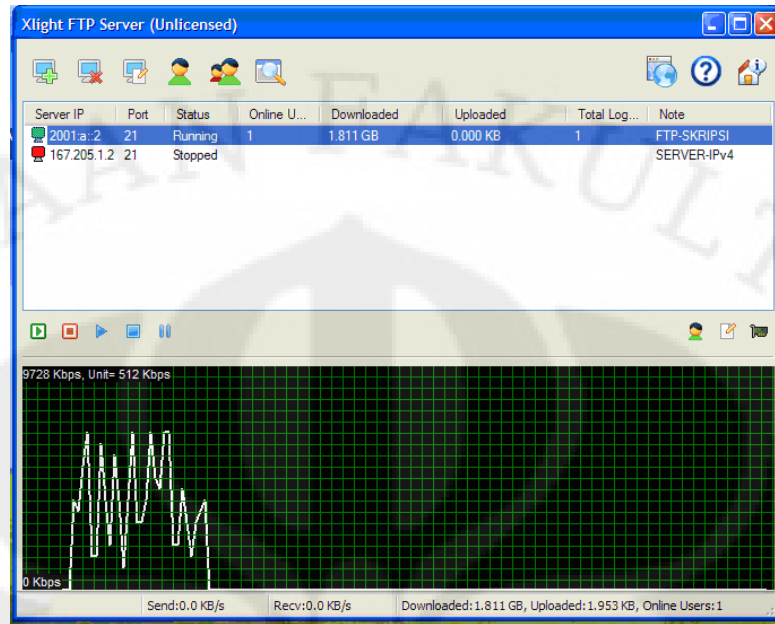
OpenVPN adalah perangkat lunak yang ditujukan khusus sebagai solusi VPN. Program ini bersifat *open source* dan *multi platform* sehingga mudah didapat dan fleksibel dalam penggunaannya. Pada pengujian ini digunakan OpenVPN 2.0.9 berbasis sistem operasi Linux. OpenVPN di-*install* pada *tunnel end-point* (Router A & Router B). Tampilan OpenVPN pada Linux masih berbasis terminal seperti ditunjukkan pada Gambar 3.6.



Gambar 3.6 Tampilan OpenVPN saat aktif

### 3.3.3. Xlight FTP

Xlight FTP merupakan perangkat lunak yang berfungsi sebagai *server* FTP. *Server* FTP digunakan untuk menyimpan file-file yang akan di-*download* oleh *FTP client*. Xlight FTP *Server* bekerja pada *platform* Windows, telah memiliki GUI yang memudahkan manajemen oleh administrator serta telah mendukung IPv6. Tampilan Xlight FTP dapat dilihat pada Gambar 3.7.



**Gambar 3.7** Tampilan Xlight FTP pada sisi *server*

### 3.3.4. Smart FTP

Smart FTP merupakan perangkat lunak yang berfungsi sebagai *client* FTP. FTP *client* digunakan oleh user yang ingin melakukan *download/upload* data dari FTP *server*. Smart FTP yang bekerja pada *platform* Windows, telah memiliki GUI serta telah mendukung IPv6. Tampilan Smart FTP dapat dilihat pada Gambar 3.8.



**Gambar 3.8** Tampilan program Smart FTP pada sisi *client*

### 3.3.5. Miredo

Miredo merupakan perangkat lunak implementasi Teredo yang bekerja pada *platform* sistem operasi berbasis Unix. Karena berbasis Unix, maka Miredo dapat didistribusikan secara bebas. Miredo dapat bekerja baik sebagai Teredo *server* maupun Teredo *relay*, tergantung pada konfigurasi yang dibutuhkan.

### 3.3.6. Wireshark

Wireshark merupakan perangkat lunak yang digunakan untuk mengamati paket-paket yang melalui suatu *interface* tertentu. Wireshark merupakan pengembangan dari perangkat lunak serupa yang bernama Ethereal. Wireshark dapat berjalan baik pada sistem operasi Windows maupun Linux. Tampilan Wireshark saat mengambil paket dapat dilihat pada Gambar 3.9.



**Gambar 3.9** Tampilan program Wireshark saat meng-*capture* paket

### 3.4. METODE PENGAMBILAN DATA

Tujuan utama dari pengujian jaringan *test-bed* adalah mendapatkan parameter-parameter yang dibutuhkan untuk perbandingan performa antara topologi IPv6 VPN dengan topologi Teredo. Dari perbandingan tersebut akan dapat ditentukan mana topologi yang lebih baik dalam memberikan konektivitas IPv6 untuk jaringan *private* yang berada dibalik NAT. Kemudian, untuk melihat sejauh mana perbedaan kedua topologi tersebut dengan jaringan *existing* (IPv4) maka dilakukan juga pengujian pada topologi IPv4 murni.

Pengujian jaringan *test-bed* dilakukan dengan dua metode. Metode pertama bertujuan untuk mengukur kemampuan dari masing-masing topologi jaringan secara umum, sedangkan metode kedua bertujuan untuk mengukur performa masing-masing topologi jaringan saat digunakan pada aplikasi FTP.

#### 3.4.1. Pengujian Performa Jaringan Dengan Paket TCP Dan UDP

Pengujian performa *tunneling* dilakukan dengan mengirimkan paket-paket TCP dan UDP dari komputer *client* ke komputer *server*. Parameter yang diambil dari pengujian ini adalah *throughput* jaringan, persentase *frame loss* dan *jitter* paket. Pengukuran *throughput* jaringan dilakukan dengan mengirimkan paket-paket TCP dengan *window size* yang telah ditentukan selama selang periode tertentu, dalam pengujian ini 10 detik. Variasi yang dilakukan adalah pada ukuran *window size* dari TCP. Tujuannya untuk melihat adakah hubungan antara *window size* dengan *throughput* jaringan. *Window size* yang digunakan bervariasi yaitu 8, 16, 32, 64 dan 128 KB. Pengukuran *frame loss* dan *jitter* dalam jaringan dilakukan dengan mengirimkan paket-paket UDP dengan ukuran yang telah ditentukan. Tujuannya adalah untuk melihat adakah hubungan antara ukuran paket UDP yang dikirimkan dengan *frame loss* dan *jitter* selama transmisi. Ukuran paket UDP divariasikan yaitu sebesar 16, 80, 208, 464, 720, 976, 1232 dan 1452 byte.

Pengambilan data untuk setiap pengujian TCP dan UDP dilakukan sebanyak 5 kali untuk setiap variasi yang dilakukan pada masing-masing topologi jaringan. Pengujian dilakukan dengan bantuan program Iperf yang dijalankan bersamaan pada komputer *client* dan *server*. Berikut adalah sintaks yang dimasukkan dalam melakukan pengujian ini.



Komputer *Server*

```
C:> iperf -s -V -u -l <ukuran paket UDP> -i 1
```

Komputer *Client*

```
C:> iperf -c <alamat IP server> -V -u -l <ukuran paket UDP> -P 5
```

Contoh tampilan Iperf saat digunakan dapat dilihat pada Gambar 3.5 yang terdapat di halaman 30.

Pengukuran tambahan dengan aplikasi pengamat paket seperti Wireshark dapat dilakukan pada sisi client. Fungsinya adalah untuk menjamin kestabilan jumlah paket yang dikirimkan oleh aplikasi *packet generator* Iperf. Terutama dalam pengujian koneksi UDP dimana Iperf harus mengirimkan paket dengan *rate* yang konstan untuk setiap variasi pengujian yang dilakukan. Sementara dalam pengujian koneksi TCP hal tersebut tidak dapat dilakukan karena *rate* pengiriman paket TCP sangat bergantung pada performa setiap jaringan.

### 3.4.2. Pengujian Performa Jaringan Pada Aplikasi File Transfer Protocol

Pengujian performa *File Transfer Protocol* dilakukan dengan menjalankan aplikasi FTP *client-server* pada masing-masing topologi yang diujikan. Parameter yang diambil dari pengujian ini adalah *throughput*, *transfer time* dan *latency* paket dalam jaringan. File yang akan ditransfer melalui FTP bervariasi ukurannya yaitu 8, 16, 32, 64 dan 128 Mbyte. Tujuannya adalah untuk melihat adakah hubungan antara ukuran file yang dikirim dengan parameter-parameter tersebut diatas. Pengambilan data dilakukan sebanyak 10 kali untuk setiap variasi ukuran file pada masing-masing topologi. Pengujian dilakukan dengan bantuan aplikasi Xlight FTP pada komputer *server* serta Smart FTP pada komputer *client*. Program penangkap paket, Wireshark, di-*install* di komputer *client* dan digunakan untuk mendapatkan parameter-parameter yang dibutuhkan.

Total jumlah pengambilan data yang dilakukan untuk keseluruhan skenario pengujian dan topologi adalah sebanyak 345 kali. Pengujian dilakukan dengan menjaga agar lingkungan *test-bed* semirip dan sestabil mungkin untuk setiap topologi yang diujikan. Hal ini sangat penting untuk bisa mendapatkan parameter perbandingan yang *valid*, sehingga dapat diambil kesimpulan yang baik dan dapat dipertanggung jawabkan.

## BAB IV ANALISA

### 4.1. ANALISA TOPOLOGI

Mekanisme pengujian dilakukan dengan pembuatan test-bed untuk setiap topologi yang diujikan. Pada topologi test-bed IPv4 murni dan IPv6 VPN digunakan 5 buah PC dengan 2 PC sebagai *host*, 2 PC sebagai router dan 1 PC sebagai NAT. Pada pengujian topologi Teredo digunakan 7 buah PC dengan tambahan 1 PC sebagai Teredo *relay* dan 1 PC sebagai Teredo *server*. Hasil perbandingan yang lebih akurat akan didapatkan apabila pada topologi IPv4 murni dan IPv6 VPN ditambahkan sebuah *intermediary router* lagi sehingga didapatkan jumlah *next hop* yang sama dengan topologi Teredo, namun karena keterbatasan jumlah komputer yang tersedia hal tersebut tidak dapat dilakukan.

Topologi Teredo memiliki 2 perangkat jaringan dengan fungsi khusus pada skenario Teredo, yaitu Teredo *relay* dan Teredo *server*. Teredo *server* berfungsi memberikan alamat IPv6 kepada Teredo *client*. Karena NAT yang digunakan merupakan *full cone* NAT, maka setelah alamat berhasil diberikan Teredo *relay* akan memegang peranan dalam proses transmisi data selanjutnya. Teredo bekerja dengan metode *tunneling* berbasis UDP, artinya setiap paket yang memasuki tunnel akan dienkapsulasi dengan paket UDP. Pada topologi Teredo *tunneling* selama transmisi data akan terjadi antara Teredo *client* - NAT - Router1 - Teredo *relay*.

Topologi IPv6 VPN juga memiliki 2 perangkat jaringan dengan fungsi khusus, yaitu VPN *client* dan VPN *server*. Kedua perangkat tersebut berfungsi sebagai *tunnel end-point* yaitu perangkat dimana proses enkapsulasi/dekapsulasi berlangsung. Router1 yang juga berfungsi sebagai VPN *client* menginisiasi koneksi VPN dengan Router2 sebagai VPN *server*. Berbeda dengan Teredo yang hanya bekerja dengan *tunneling* UDP, VPN memiliki opsi untuk menggunakan *tunneling* berbasis TCP ataupun UDP. Namun, dengan alasan performa maka *tunneling* UDP lebih sering digunakan.

Topologi IPv4 murni berbeda dengan topologi Teredo ataupun IPv6 VPN. Topologi IPv4 murni tidak memiliki perangkat jaringan khusus dan juga tidak

menggunakan metode *tunneling*. Proses yang terjadi selama transmisi data hanya *routing* dan *forwarding* paket seperti jaringan pada umumnya.

#### 4.2. ANALISA PERFORMA JARINGAN DENGAN TCP DAN UDP

Pengujian pertama yang dilakukan adalah pengujian koneksi dengan paket TCP dan UDP untuk masing-masing topologi. TCP dan UDP adalah dua protokol yang banyak digunakan dalam jaringan internet berbasis IP. Keduanya dibuat dengan tujuan yang berbeda. TCP (*Transmission Control Protocol*) misalnya, bersifat *connection oriented*, artinya protokol ini memiliki kemampuan untuk menjamin transfer dan kontrol data hingga *node* tujuan. Sebaliknya UDP (*User Datagram Protocol*) bersifat *connectionless oriented*, yang berarti protokol ini tidak memiliki mekanisme yang dapat menjamin sampainya paket ke *node* tujuan. Pengujian dilakukan menggunakan bantuan program *packet generator*, Iperf, yang dapat menghasilkan dan mengirimkan paket-paket TCP dan UDP sesuai skenario pengujian. Saat pengujian, Iperf sebagai *packet generator* akan mengirimkan paket TCP atau UDP sebanyak-banyaknya selama rentang waktu tertentu (10s).

Penggunaan Iperf pada modus TCP akan menghasilkan keluaran parameter *throughput* jaringan. Variasi yang dilakukan pada pengujian koneksi TCP adalah pada ukuran *window size*. Pada koneksi TCP, *window size* menentukan jumlah maksimum data yang dapat berada dalam jaringan pada saat yang bersamaan. Variasi *window size* merupakan prosedur standar dalam proses *tuning* koneksi TCP untuk mendapatkan *bandwidth* atau *throughput* yang maksimum. Apabila ukuran *window size* terlalu kecil maka jaringan akan *idle* untuk waktu tertentu sehingga didapatkan performa jaringan yang buruk [9].

Penggunaan Iperf pada modus UDP akan menghasilkan keluaran parameter *frame loss* dan *jitter*. Variasi yang dilakukan pada pengujian koneksi UDP adalah pada ukuran datagram UDP yang dikirim. Variasi ukuran datagram UDP merupakan prosedur standar dalam proses *tuning* koneksi UDP untuk mendapatkan persentase *frame loss* dan *jitter* yang minimum. Ukuran datagram UDP biasanya akan disesuaikan dengan ukuran datagram yang dihasilkan oleh aplikasi yang digunakan. Secara default Iperf akan mengirimkan datagram UDP berukuran 1470 bytes. Data lengkap hasil pengujian untuk topologi IPv4 murni,

Teredo dan IPv6 VPN dapat dilihat masing-masing Lampiran 4, Lampiran 5, dan Lampiran 6.

#### 4.2.1. Analisa TCP : *Throughput*

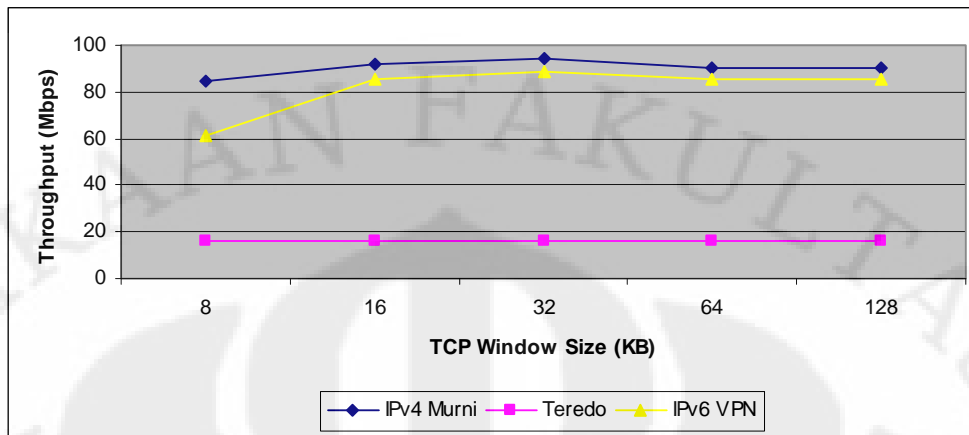
*Throughput* merupakan parameter yang menunjukkan jumlah bit rata-rata data yang dapat ditransfer dari satu *node* jaringan ke *node* jaringan lainnya setiap detik. *Throughput* TCP diukur dengan membandingkan jumlah byte data TCP yang terkirim melalui jaringan dengan rentang waktu pengiriman. Jaringan dengan performa yang baik adalah jaringan dengan *throughput* tinggi. Jaringan yang memiliki *throughput* tinggi akan mampu mentransfer data lebih banyak untuk waktu yang sama.

Dari hasil pengujian untuk lima kali pengambilan data, didapatkan rata-rata *throughput* untuk setiap variasi *window size* seperti dijabarkan pada Tabel 4.1 berikut.

**Tabel 4.1** Data *throughput* TCP

| WINDOW<br>SIZE (KB) | THROUGHPUT (Mbps) |        |          |
|---------------------|-------------------|--------|----------|
|                     | IPv4 Murni        | Teredo | IPv6 VPN |
| 8                   | 84.44             | 15.92  | 61.48    |
| 16                  | 92.24             | 16.08  | 85.12    |
| 32                  | 94.20             | 15.86  | 88.60    |
| 64                  | 90.46             | 15.96  | 85.68    |
| 128                 | 90.32             | 16.00  | 85.48    |

Pada Tabel 4.1 diatas dapat dilihat bahwa besarnya *throughput* TCP ikut berubah saat ukuran *window size* berubah. Perubahan *throughput* tersebut tidak berbanding lurus ataupun berbanding terbalik dengan ukuran *window size*. Sehingga, *throughput* yang maksimum tidak pasti didapatkan saat ukuran *window size* maksimum. Artinya memang dibutuhkan sedikit *trial & error* dalam mengatur ukuran *window size* untuk mendapatkan *throughput* yang maksimum. Dalam hubungannya dengan *tuning* TCP, tabel hasil pengujian menunjukkan bahwa rata-rata *throughput* maksimum pada topologi IPv4 dan IPv6 VPN didapatkan saat *window size* berukuran 32 KB, sedangkan untuk topologi Teredo maksimum didapatkan saat *window size* berukuran 16 KB.



**Gambar 4.1** Grafik garis perbandingan *throughput* TCP

Berdasarkan grafik yang ditunjukkan pada Gambar 4.1, terlihat bahwa topologi IPv4 murni memiliki rata-rata *throughput* TCP tertinggi untuk setiap variasi *window size*. Topologi IPv6 VPN sendiri memiliki rata-rata *throughput* berkisar antara 61.48 Mbps – 88.60 Mbps. Apabila dibandingkan dengan rata-rata *throughput* topologi IPv4 murni, nilai tersebut lebih buruk antara 5.28% - 27.19% untuk setiap variasi pengujian. Saat topologi IPv6 VPN dibandingkan dengan topologi Teredo, didapatkan rata-rata *throughput* TCP IPv6 VPN lebih baik antara 286.18 % - 458.64 % untuk setiap variasi pengujian. Kedua perbandingan diatas menunjukkan bahwa dari sisi *throughput* TCP, IPv4 murni sedikit lebih baik dari IPv6 VPN sedangkan IPv6 VPN jauh lebih baik dari Teredo.

#### 4.2.2. Analisa UDP : *Frame Loss*

*Frame loss* merupakan parameter yang menunjukkan persentase paket yang hilang selama proses transfer data. *Frame loss* dideteksi oleh *server* melalui ID yang terdapat dalam setiap datagram. Saat pengiriman, biasanya datagram UDP dibagi menjadi beberapa paket IP (terfragmentasi). Apabila salah satu paket tersebut hilang selama perjalanan maka keseluruhan datagram UDP akan hilang. Berbeda dengan TCP yang memiliki mekanisme *acknowledgement* dan *retransmission*. Itulah sebabnya koneksi UDP sangat rentan terhadap *packet/frame loss*. Agar tidak salah mengukur *packet loss* dan bukan *datagram loss*, maka ukuran datagram UDP yang digunakan saat pengujian diusahakan

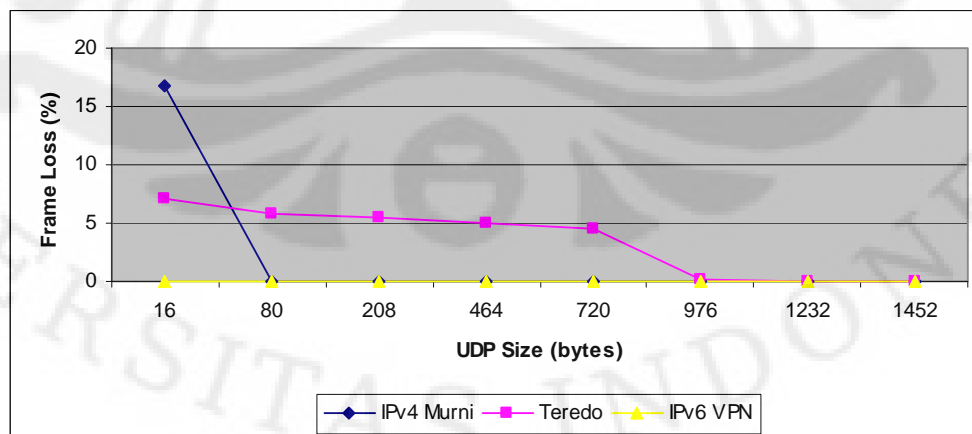
sekecil mungkin agar tidak terfragmentasi. Jaringan dengan persentase *frame loss* yang kecil menunjukkan bahwa jaringan tersebut memiliki reliabilitas tinggi.

Dari hasil pengujian untuk lima kali pengambilan data, didapatkan rata-rata *frame loss* untuk setiap variasi ukuran datagram UDP seperti dijabarkan pada Tabel 4.2 berikut.

**Tabel 4.2** Data *frame loss* UDP

| UDP SIZE (byte) | FRAME LOSS (%) |        |          |
|-----------------|----------------|--------|----------|
|                 | IPv4 Murni     | Teredo | IPv6 VPN |
| 16              | 16.80          | 7.04   | 0.02     |
| 80              | 0.00           | 5.78   | 0.00     |
| 208             | 0.02           | 5.54   | 0.00     |
| 464             | 0.00           | 4.92   | 0.00     |
| 720             | 0.00           | 4.52   | 0.00     |
| 976             | 0.00           | 0.21   | 0.00     |
| 1232            | 0.00           | 0.00   | 0.00     |
| 1452            | 0.00           | 0.00   | 0.00     |

Pada Tabel 4.2 diatas dapat dilihat bahwa persentase *frame loss* UDP ikut berubah saat ukuran datagram berubah. Persentase *frame loss* cenderung tinggi saat datagram yang dikirim berukuran kecil. Dalam kaitannya dengan *tuning* koneksi UDP, maka ukuran datagram UDP yang baik yaitu  $\geq 80$  byte untuk topologi IPv4 murni,  $\geq 976$  byte untuk topologi Teredo, dan  $\geq 16$  byte untuk topologi IPv6 VPN. Sama seperti proses *tuning* TCP, untuk mendapatkan persentase *frame loss* yang minimum pada UDP dibutuhkan *trial & error*.



**Gambar 4.2** Grafik garis perbandingan *frame loss* UDP

Berdasarkan grafik yang ditunjukkan pada Gambar 4.2, terlihat bahwa topologi IPv6 VPN memiliki persentase *frame loss* terendah diantara kedua topologi lainnya. Persentase *frame loss* pada topologi IPv6 VPN cenderung tidak signifikan, yaitu hanya berkisar antara 0% - 0.02% untuk setiap variasi ukuran datagram UDP yang dikirim. Bila dibandingkan dengan topologi IPv4 murni, *frame loss* yang cukup signifikan hanya terjadi saat datagram UDP yang dikirim berukuran 16 byte, yaitu sebesar 16.80%. Sementara topologi Teredo menghasilkan *frame loss* yang cukup signifikan saat datagram UDP yang dikirim berukuran 16, 80, 208, 464 dan 720 byte. Persentase *frame loss* yang tinggi umumnya mengindikasikan adanya kongesti yang terjadi dalam jaringan. Kongesti dalam jaringan dapat terjadi karena beberapa faktor salah satunya adalah tidak seimbangannya kecepatan pengiriman paket dengan waktu pemrosesan paket di salah satu *node* jaringan. Hal tersebut mengakibatkan adanya *bottleneck* pada salah satu segmen jaringan. Dari hasil pengujian didapatkan bahwa topologi IPv6 VPN memiliki tingkat reliabilitas tertinggi dibandingkan topologi IPv4 murni dan Teredo. Khusus untuk topologi Teredo, *frame loss* lebih rentan terjadi untuk lebih banyak variasi ukuran datagram UDP. Hal ini menunjukkan bahwa kemungkinan terjadinya kongesti lebih besar pada topologi Teredo disebabkan metode enkapsulasi/dekapsulasi paket yang dipergunakan memakan waktu lebih lama.

#### 4.2.3. Analisa UDP : Jitter

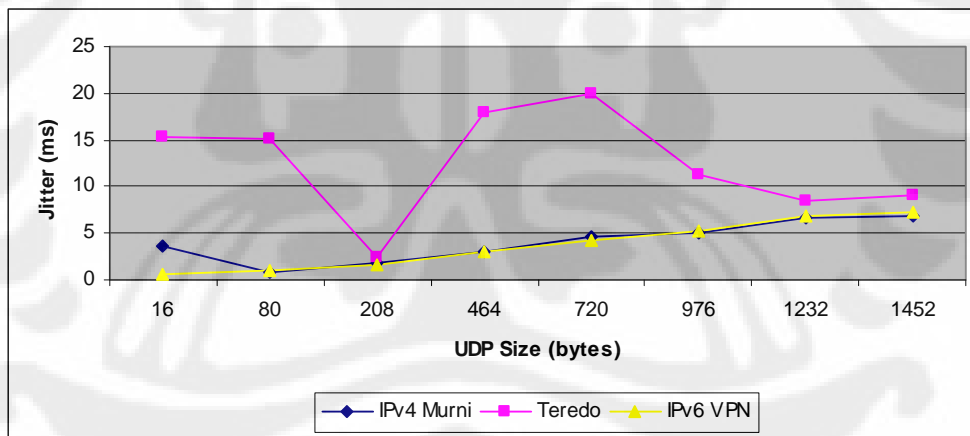
*Jitter* atau *delay invariant* merupakan parameter yang menunjukkan variasi delay antar paket dalam satu pengiriman data yang sama. Jaringan yang baik adalah jaringan dengan *jitter* yang kecil, karena dengan *jitter* tinggi berarti paket-paket tidak akan diterima secara bersamaan. Hal tersebut akan sangat mengganggu terutama untuk aplikasi-aplikasi multimedia melalui jaringan. Penghitungan *jitter* oleh Iperf dilakukan secara kontinu, dimana *server* akan menghitung waktu transit relatif (waktu penerimaan *server* – waktu pengiriman *client*) untuk tiap paket milik data yang sama [9].

Dari hasil pengujian untuk lima kali pengambilan data, didapatkan rata-rata *jitter* paket untuk setiap ukuran file seperti dijabarkan pada Tabel 4.3 berikut.

**Tabel 4.3** Data *jitter* UDP

| UDP<br>SIZE (byte) | JITTER (ms) |        |          |
|--------------------|-------------|--------|----------|
|                    | IPv4 Murni  | Teredo | IPv6 VPN |
| 16                 | 3.67        | 15.36  | 0.70     |
| 80                 | 0.90        | 15.04  | 1.06     |
| 208                | 1.76        | 2.37   | 1.68     |
| 464                | 2.96        | 17.90  | 2.99     |
| 720                | 4.58        | 20.00  | 4.18     |
| 976                | 4.99        | 11.19  | 5.26     |
| 1232               | 6.58        | 8.50   | 6.88     |
| 1452               | 6.91        | 9.00   | 7.26     |

Pada Tabel 4.3 diatas dapat dilihat bahwa besar *jitter* ikut berubah saat ukuran datagram berubah. Secara umum hasil pengujian menunjukkan bahwa *jitter* meningkat seiring dengan meningkatnya ukuran datagram UDP yang dikirimkan. Khusus untuk topologi Teredo, *jitter* yang timbul cenderung fluktuatif dimana hal ini menunjukkan adanya antrian paket pada salah satu *node* jaringan yang dapat menimbulkan kongesti [10]. Antrian paket tersebut menyebabkan waktu transfer paket tidak dapat diprediksi sehingga *jitter* paket meningkat.



**Gambar 4.3** Grafik garis perbandingan *jitter* UDP

Berdasarkan grafik yang ditunjukkan oleh Gambar 4.3, terlihat bahwa topologi IPv4 murni dan IPv6 VPN memiliki *jitter* yang tidak jauh berbeda. Secara umum *jitter* topologi IPv4 murni masih lebih baik dibandingkan topologi IPv6 VPN. *Jitter* pada topologi IPv6 VPN lebih buruk 0.9% - 15.05% pada ukuran datagram 80, 464, 972, 1232 dan 1452 byte kecuali pada ukuran datagram



16, 208 dan 720 byte dimana IPv6 VPN lebih baik 5.02% - 423.31% dibandingkan topologi IPv4 murni. Saat dibandingkan dengan topologi Teredo, *jitter* pada topologi IPv6 VPN masih lebih baik 23.64% - 2088.69% untuk setiap variasi ukuran datagram. Fluktuatif dan tingginya *jitter* pada topologi Teredo mengindikasikan bahwa pemrosesan paket pada Teredo cenderung tidak stabil sehingga berpotensi menimbulkan kongesti pada jaringan.

#### 4.3. ANALISA PERFORMA JARINGAN PADA APLIKASI FTP

*File Transfer Protocol* (FTP) merupakan suatu protokol yang banyak digunakan dalam men-*download/upload* data melalui jaringan internet. FTP merupakan salah satu protokol yang bekerja dengan memanfaatkan protokol TCP/IP, seperti juga HTTP. Pada prosesnya, suatu koneksi FTP memiliki bentuk koneksi *client-server*. FTP *client* adalah komputer yang akan men-*download* dari *server* atau meng-*upload* ke *server*. FTP *server* adalah komputer dimana file-file yang di-*download/upload* tersebut akan disimpan. Komputer yang berperan sebagai *client* akan melakukan *request* kepada FTP *server*. Proses tersebut diawali dengan *three way handshaking*, yang merupakan prosedur umum dalam sebuah koneksi TCP. Setelah koneksi antar keduanya terjadi dan FTP *server* mengautentikasi *client* maka *client* berhak mendapatkan layanan dari FTP *server*. Modus FTP yang digunakan pada pengujian adalah modus *anonymous*, dimana setiap FTP *client* dapat mengakses FTP *server* tanpa perlu terdaftar.

Parameter yang diambil dalam pengujian ini adalah *throughput*, *transfer time* serta *latency*. Parameter-parameter tersebut merupakan besaran yang dirasa mampu menunjukkan performa aplikasi FTP dalam suatu jaringan. Setiap parameter saling berhubungan satu dengan lainnya. Misalkan untuk suatu file dengan ukuran sama, makin besar *throughput* jaringan berarti *transfer time* dan *latency* seharusnya akan semakin kecil. Begitu pula sebaliknya, apabila *latency* makin besar, maka umumnya *transfer time* akan semakin besar sehingga *throughput* jaringan akan mengecil.

Pengujian dilakukan dengan memvariasikan ukuran file yang akan di-*download* oleh FTP *client*. File-file yang diujikan tersebut adalah sebagai berikut.

Data[8MB].mp3 ⇒ Ukuran 8,196 megabytes  
 Data[16MB].pdf ⇒ Ukuran 17,345 megabytes  
 Data[32MB].exe ⇒ Ukuran 32,919 megabytes  
 Data[64MB].rar ⇒ Ukuran 64,846 megabytes  
 Data[128MB].avi ⇒ Ukuran 131,340 megabytes

Tipe file tersebut dipilih sesuai dengan jenis file yang umum di-*download* oleh user melalui internet. Sedangkan ukurannya diatur sedemikian rupa sehingga dapat dilihat keterkaitan antara ukuran file dengan parameter-parameter yang diambil. Untuk mendapatkan parameter-parameter yang diperlukan tersebut digunakan program *packet capture* bernama Wireshark. Data lengkap hasil pengujian untuk topologi IPv4 murni, Teredo dan IPv6 VPN dapat dilihat masing-masing Lampiran 4, Lampiran 5, dan Lampiran 6.

#### 4.3.1. Analisa Latency

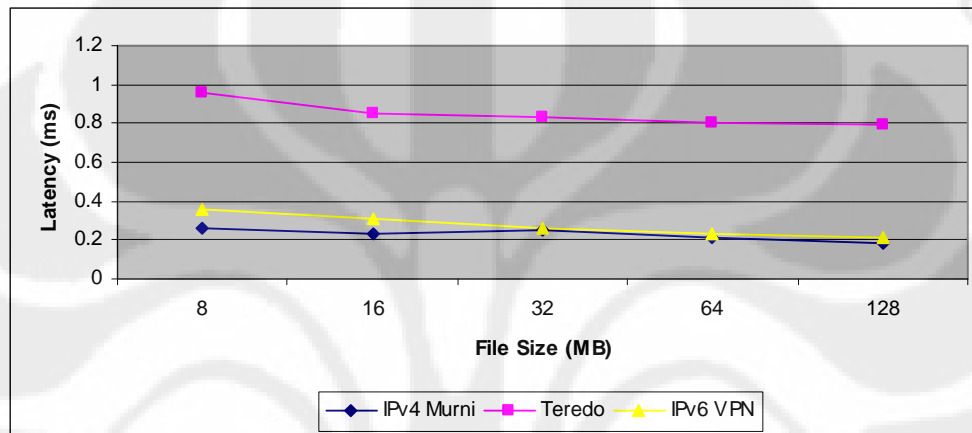
*Latency* atau *delay* adalah waktu yang dibutuhkan sebuah paket mulai dari dikirim hingga sampai ke tujuan. Karena waktu pengiriman dihitung per paket, maka dengan perhitungan *latency* dapat merepresentasikan perlakuan paket yang berbeda pada tiap topologi. Salah satu proses yang dapat menambah *latency* paket dalam jaringan selain *routing* adalah *tunneling*. Dengan mengamati *latency* pada ketiga topologi maka dapat dibandingkan sejauh mana performa *tunneling* tiap topologi yang diujikan.

Dari hasil pengujian untuk 10 kali pengambilan data, didapatkan rata-rata *latency* untuk setiap ukuran file seperti dijabarkan pada Tabel 4.4 berikut.

**Tabel 4.4** Data *latency* FTP

| FILE SIZE<br>(MB) | LATENCY (ms) |        |          |
|-------------------|--------------|--------|----------|
|                   | IPv4 Murni   | Teredo | IPv6 VPN |
| 8                 | 0.26         | 0.95   | 0.36     |
| 16                | 0.24         | 0.85   | 0.31     |
| 32                | 0.25         | 0.83   | 0.26     |
| 64                | 0.21         | 0.81   | 0.24     |
| 128               | 0.18         | 0.79   | 0.22     |

Pada Tabel 4.4 diatas dapat dilihat bahwa nilai rata-rata *latency* cenderung akan semakin kecil seiring dengan makin besarnya ukuran file yang ditransfer. *Latency* atau *delay* menunjukkan waktu yang dibutuhkan tiap paket untuk ditransfer melewati jaringan. Makin kecil nilai *latency*, maka akan semakin baik performa jaringan tersebut. Urutan rata-rata *latency* pada pengujian dari yang terbaik hingga terburuk adalah IPv4 Murni, IPv6 VPN dan Teredo.



**Gambar 4.4** Grafik garis perbandingan *latency* FTP

Berdasarkan grafik yang ditunjukkan oleh Gambar 4.4, terlihat bahwa topologi IPv4 murni memiliki rata-rata nilai *latency* yang terkecil dibandingkan dua topologi lainnya. Sedangkan rata-rata *latency* untuk topologi IPv6 VPN berkisar antara 0.22 s – 0.36 s tergantung dari besar ukuran file. Besar rata-rata *latency* IPv6 VPN tersebut lebih buruk 4.13% - 36.50% dibandingkan dengan rata-rata *latency* untuk topologi IPv4 murni untuk setiap variasi pengujian. *Latency* IPv6 VPN yang lebih besar dibandingkan *latency* IPv4 murni disebabkan proses enkapsulasi/dekapsulasi yang terjadi pada topologi IPv6 VPN. Proses enkapsulasi/ dekapsulasi yang terjadi pada tiap pengiriman paket tersebut akan memberikan *overhead* tersendiri dalam jaringan. Saat *latency* topologi IPv6 VPN dibandingkan dengan *latency* topologi Teredo, didapatkan *latency* yang lebih baik antara 168.07% - 267.57% untuk IPv6 VPN pada setiap variasi pengujian. Walaupun IPv6 VPN dan Teredo sama-sama menggunakan proses enkapsulasi/dekapsulasi, *latency* pada IPv6 VPN lebih kecil dibandingkan pada

IPv4 murni. Nilai *latency* IPv6 VPN yang lebih kecil menunjukkan performa enkapsulasi/dekapsulasi pada IPv6 VPN jauh lebih baik dibandingkan pada Teredo.

#### 4.3.2. Analisa Transfer Time

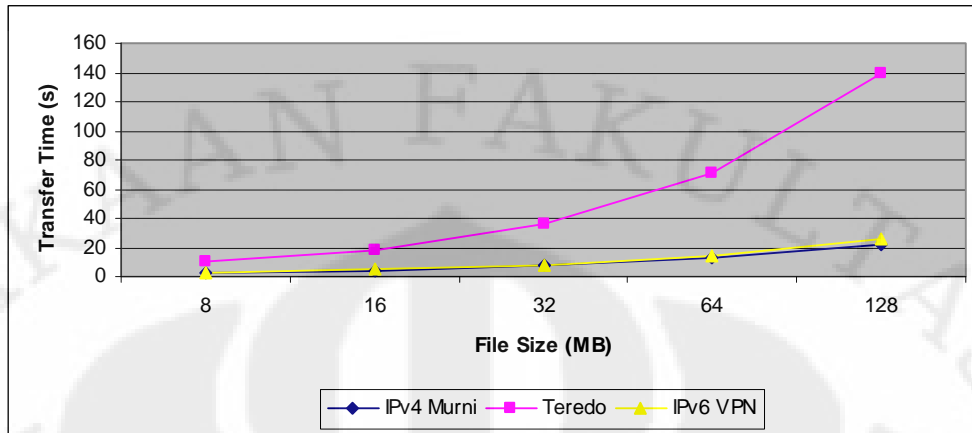
*Transfer time* merupakan jumlah total waktu yang dibutuhkan untuk mentransfer suatu file dari FTP server ke FTP client. Besarnya *transfer time* akan sangat dipengaruhi oleh *throughput* jaringan. Apabila *throughput* jaringan semakin besar, maka untuk ukuran file yang sama *transfer time* yang dibutuhkan akan semakin kecil. Perhitungan *transfer time* pada FTP dimulai saat FTP client melakukan *request* file ke FTP server hingga keseluruhan file selesai ditransfer.

Dari hasil pengujian untuk 10 kali pengambilan data, didapatkan rata-rata *transfer time* untuk setiap ukuran file seperti dijabarkan pada Tabel 4.5 berikut.

Tabel 4.5 Data transfer time FTP

| FILE SIZE<br>(MB) | TRANSFER TIME (s) |        |          |
|-------------------|-------------------|--------|----------|
|                   | IPv4 Murni        | Teredo | IPv6 VPN |
| 8                 | 2.42              | 10.56  | 2.78     |
| 16                | 3.56              | 18.26  | 4.70     |
| 32                | 7.96              | 36.74  | 7.83     |
| 64                | 13.14             | 70.47  | 14.20    |
| 128               | 22.01             | 139.96 | 25.57    |

Pada Tabel 4.4 diatas dapat dilihat bahwa nilai rata-rata *transfer time* cenderung akan semakin besar seiring dengan makin besarnya ukuran file yang ditransfer. *Transfer time* menunjukkan waktu yang dibutuhkan keseluruhan file untuk ditransfer melewati jaringan. Makin kecil nilai *transfer time*, maka akan semakin baik performa jaringan tersebut. Urutan rata-rata *transfer time* pada pengujian dari yang terbaik hingga terburuk adalah IPv4 Murni, IPv6 VPN dan Teredo.



Gambar 4.5 Grafik garis perbandingan *transfer time* FTP

Berdasarkan grafik yang ditunjukkan pada Gambar 4.5, topologi IPv4 murni memiliki nilai rata-rata *transfer time* yang terkecil dibandingkan dua topologi lainnya. Sedangkan rata-rata *transfer time* untuk topologi IPv6 VPN berkisar antara 2.78 s – 25.57 s meningkat seiring makin besarnya ukuran file. Nilai *transfer time* pada IPv6 VPN tersebut lebih buruk 8.09% - 31.99% dibandingkan *transfer time* pada IPv4 murni untuk setiap variasi pengujian . Hal ini berhubungan dengan *latency* pada IPv6 VPN yang juga lebih besar daripada *latency* IPv4 murni. Karena waktu pemrosesan untuk setiap paket pada IPv6 VPN lebih lama, maka keseluruhan transfer file juga akan berlangsung lebih lama. Saat IPv6 VPN dibandingkan dengan Teredo, didapatkan nilai *transfer time* IPv6 VPN lebih baik antara 279.41% - 447.36% untuk setiap variasi pengujian . Serupa dengan saat dibandingkan dengan IPv4 murni, jauh lebih besarnya *transfer time* pada Teredo disebabkan *latency* pada Teredo yang juga jauh lebih besar. Hal ini disebabkan proses enkapsulasi/dekapsulasi pada Teredo yang lebih buruk dibandingkan dengan IPv6 VPN.

#### 4.3.3. Analisa *Throughput*

*Throughput* menunjukkan kecepatan transfer data, yaitu besaran data rata-rata yang mampu ditransfer melewati jaringan setiap detik. *Throughput* pada pengujian FTP berbeda dengan *throughput* TCP. Pada pengujian TCP, pengujian *throughput* dilakukan dengan pengiriman paket TCP selama rentang waktu 10

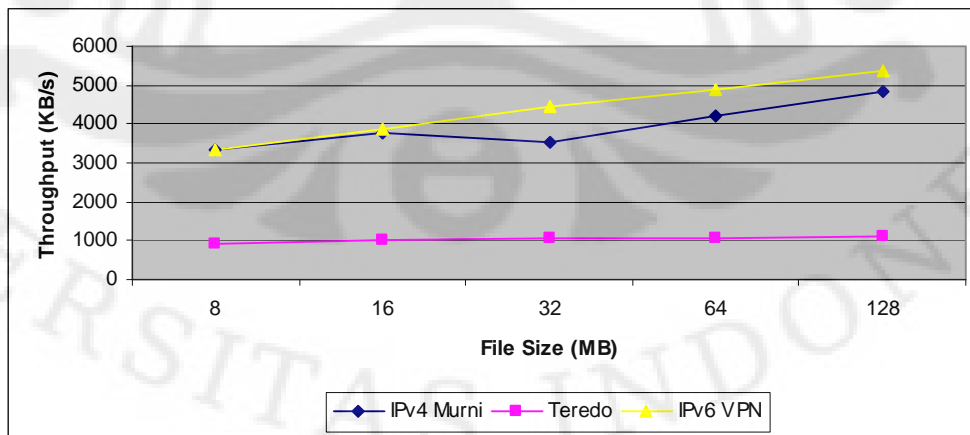
detik. Pada pengujian FTP, data yang dikirim merupakan file yang terfragmentasi menjadi paket-paket sehingga pada saat diterima pun perlu dilakukan penyusunan ulang paket menjadi file utuh. Secara sederhana dapat dijelaskan bahwa pengujian FTP akan melibatkan mekanisme-mekanisme yang lebih rumit dibandingkan pengujian koneksi TCP/UDP.

Dari hasil pengujian untuk 10 kali pengambilan data, didapatkan rata-rata *throughput* aplikasi FTP untuk setiap ukuran file seperti dijabarkan pada Tabel 4.6 berikut.

**Tabel 4.6** Data *throughput* FTP

| FILE SIZE<br>(MB) | THROUGHPUT (KB/s) |          |          |
|-------------------|-------------------|----------|----------|
|                   | IPv4 Murni        | Teredo   | IPv6 VPN |
| 8                 | 3,358.83          | 930.24   | 3,324.19 |
| 16                | 3,779.34          | 1,028.40 | 3,876.83 |
| 32                | 3,543.07          | 1,044.76 | 4,471.38 |
| 64                | 4,197.34          | 1,073.00 | 4,878.00 |
| 128               | 4,847.78          | 1,093.00 | 5,368.86 |

Pada Tabel 4.6 diatas dapat dilihat bahwa nilai rata-rata *throughput* cenderung akan semakin besar seiring dengan makin besarnya ukuran file yang ditransfer. Karena *throughput* menunjukkan kecepatan transfer data, maka makin besar nilai *throughput* akan semakin baik performa jaringan tersebut. Urutan rata-rata *throughput* pada pengujian dari yang terbaik hingga terburuk adalah IPv6 VPN, IPv4 Murni dan Teredo.



**Gambar 4.6** Grafik garis perbandingan *throughput* FTP

Berdasarkan grafik pada Gambar 4.6, terlihat bahwa topologi IPv6 VPN memiliki rata-rata *throughput* tertinggi diantara ketiga topologi yang diujikan. Rata-rata *throughput* topologi IPv6 VPN yang didapat saat pengujian berkisar antara 3,324.19 KB/s - 5,368.86 KB/s tergantung ukuran file yang ditransfer. Rata-rata *throughput* topologi IPv6 VPN lebih baik 2.58% – 26.20% bila dibandingkan dengan rata-rata *throughput* topologi IPv4 murni untuk setiap variasi pengujian. Mengacu pada hasil pengukuran *latency*, dimana *latency* pada topologi IPv6 VPN lebih besar daripada *latency* topologi IPv4 murni, tetapi topologi IPv6 VPN memiliki *throughput* yang lebih baik. Hal ini disebabkan rata-rata ukuran paket yang dikirim pada topologi IPv6 VPN lebih besar daripada rata-rata ukuran paket topologi IPv4 murni, seperti ditunjukkan hasil *summary* Wireshark pada Gambar 4.7 dibawah. Pengecualian terjadi pada saat file yang dikirim berukuran 8 Mbyte dimana IPv6 VPN lebih buruk 1.03%. Bila dibandingkan dengan rata-rata *throughput* topologi Teredo, maka rata-rata *throughput* topologi IPv6 VPN lebih baik antara 257.35% – 391.21% untuk setiap variasi pengujian. Hal ini sangat beralasan, karena *transfer time* pada Teredo jauh lebih besar daripada *transfer time* pada IPv6 VPN. Karena *transfer time* yang besar tersebut, maka untuk ukuran file yang sama, rata-rata *throughput* Teredo akan jauh lebih kecil.

| Traffic                       | Captured       | Displayed      | Traffic                       | Captured      | Displayed     |
|-------------------------------|----------------|----------------|-------------------------------|---------------|---------------|
| Between first and last packet | 22.073 sec     | 22.073 sec     | Between first and last packet | 22.977 sec    | 22.977 sec    |
| Packets                       | 113389         | 113387         | Packets                       | 129797        | 129797        |
| Avg. packets/sec              | 5136.953       | 5136.863       | Avg. packets/sec              | 5649.093      | 5649.093      |
| Avg. packet size              | 1142.000 bytes | 1142.243 bytes | Avg. packet size              | 873.000 bytes | 873.173 bytes |
| Bytes                         | 129515733      | 129515561      | Bytes                         | 113335189     | 113335189     |
| Avg. bytes/sec                | 5867555.564    | 5867547.772    | Avg. bytes/sec                | 4932633.403   | 4932633.403   |
| Avg. MBit/sec                 | 46.940         | 46.940         | Avg. MBit/sec                 | 39.461        | 39.461        |

**Gambar 4.7** Statistik IPv6 VPN (kiri) dan IPv4 murni (kanan) untuk ukuran file 128 MB

#### 4.4. ANALISA KESELURUHAN

Setelah melakukan perbandingan dari hasil pengujian yang didapatkan untuk ketiga topologi jaringan, dapat disimpulkan bahwa topologi IPv6 VPN memiliki performa yang tidak jauh berbeda dibandingkan topologi IPv4 murni.

IPv6 VPN lebih unggul dibandingkan IPv4 murni pada sisi *frame loss* UDP dan *throughput* FTP. Sedangkan dibandingkan IPv6 VPN, IPv4 murni memiliki keunggulan pada sisi *throughput* TCP, *jitter* UDP, *delay/latency* dan total *transfer time* FTP. Bila IPv6 VPN dibandingkan dengan Teredo, dimana keduanya sama-sama merupakan mekanisme yang mampu memberikan konektivitas IPv6 untuk jaringan *private* yang berada di balik NAT infrastruktur IPv4. Hasil pengujian menunjukkan bahwa IPv6 VPN memiliki keunggulan pada setiap parameter pengujian. IPv6 VPN memberikan *throughput* yang lebih tinggi baik aplikasi TCP. Pada aplikasi UDP pun IPv6 VPN menghasilkan persentase *frame loss* dan *jitter* yang jauh lebih baik dan stabil. Saat kedua topologi digunakan untuk aplikasi internet yang spesifik, yaitu *File Transfer Protocol*, hasil pengujian pun masih menunjukkan bahwa IPv6 VPN masih jauh lebih baik dibandingkan Teredo. Walaupun keduanya sama-sama menggunakan *tunneling* dengan enkapsulasi UDP untuk dapat menembus NAT, performa *tunneling* keduanya jauh berbeda. IPv6 VPN memiliki proses *tunneling* yang lebih baik, hal ini dapat dilihat dari rata-rata *latency* IPv6 VPN yang lebih kecil dibandingkan Teredo (Tabel 4.4). Kecilnya angka *latency* akan memberikan total *transfer time* yang lebih kecil pula, sehingga untuk ukuran file yang sama memberikan *throughput* yang jauh lebih besar.

Berdasarkan hasil pengujian tersebut, IPv6 VPN akan menjadi mekanisme alternatif yang lebih baik dibandingkan Teredo dalam memberikan konektivitas IPv6 untuk jaringan *private* yang berada dibalik NAT.



## **BAB V**

### **KESIMPULAN**

1. *Throughput* TCP untuk IPv6 VPN pada ukuran *window* 16 KB lebih baik 429.35% dari Teredo tetapi lebih buruk 7.72% dari IPv4 murni, sedangkan pada ukuran *window* 32 KB IPv6 VPN lebih baik 436.84% dari Teredo tetapi lebih buruk 5.28% dari IPv4 murni.
2. Topologi IPv6 VPN memiliki reliabilitas yang lebih tinggi dan stabil dengan persentase *frame loss* UDP antara 0% - 0.02%, sementara Teredo 0% - 7.04% dan IPv4 murni 0% - 16.8%.
3. *Jitter* UDP untuk IPv6 VPN pada ukuran UDP 464 byte lebih baik 499.46% dari Teredo tetapi lebih buruk 0.90% dari IPv4 murni, sedangkan pada ukuran UDP 976 byte IPv6 VPN lebih baik 112.85% dari Teredo tetapi lebih buruk 5.02% dari IPv4 murni.
4. *Latency* FTP untuk IPv6 VPN pada ukuran file 16 MB lebih baik 173.35% dari Teredo tetapi lebih buruk 31.77% dari IPv4 murni, sedangkan pada ukuran file 64 MB IPv6 VPN lebih baik 242.95% dari Teredo tetapi lebih buruk 10.29% dari IPv4 murni.
5. *Transfer time* FTP untuk IPv6 VPN pada ukuran file 16 MB lebih baik 288.81% dari Teredo tetapi lebih buruk 31.99% dari IPv4 murni, sedangkan pada ukuran file 64 MB IPv6 VPN lebih baik 396.22% dari Teredo tetapi lebih buruk 8.09% dari IPv4 murni.
6. *Throughput* FTP untuk IPv6 VPN pada ukuran file 16 MB lebih baik 276.98% dari Teredo dan lebih baik 2.58% dari IPv4 murni, sedangkan pada ukuran file 64 MB IPv6 VPN lebih baik 354.61% dari Teredo dan lebih baik 16.22% dari IPv4 murni.
7. Hasil pengujian menunjukkan bahwa IPv6 VPN memiliki performa yang tidak jauh berbeda dengan IPv4 murni.
8. Hasil pengujian menunjukkan bahwa sebagai metode yang mampu memberikan konektivitas IPv6 bagi jaringan *private* yang berada dibalik NAT, topologi IPv6 VPN memiliki performa yang jauh lebih baik dibandingkan topologi Teredo.

## DAFTAR ACUAN

- [1] Silvia Hagen, *IPv6 Essentials : Second Edition*, (USA : O'Reilly, Mei 2006), Section 10.4.
- [2] Geoff Huston, "Anatomy : A Look Inside Network Address Translator", *The Internet Protocol Journal – Volume 7 Number 3*, Agustus 2004, hal. 2. Diakses 5 Oktober 2007, dari Cisco.com.  
[http://www.cisco.com/web/about/ac123/ac147/archived\\_issues/ipj\\_7-3/anatomy.html](http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html)
- [3] Kusuma Ayu L., "Analisa Interkoneksi Internet Protocol Security (IPsec) pada Jaringan Berbasis Network Address Translation (NAT)", *Tugas Besar Kelompok Keahlian Teknologi Informasi Sekolah Teknik Elektro dan Informatika ITB*, 2007, hal. 11.
- [4] "Teredo Overview", *Microsoft Technet Articles*, 15 Mei 2007, Section 4. Diakses 4 September 2007, dari Microsoft Technet  
<http://www.microsoft.com/technet/network/teredo.mspx>
- [5] Victor Olifer, "Different Flavours of VPN : Technology and Applications", *JNT Association Journal* : Maret 2007, hal. 3.
- [6] Stockebrand, Benedikt, *IPv6 in Practice : A Unixer's Guide to the Next Generation Internet*, (Berlin : Springer, 2007), hal. 183
- [7] Joseph Davies, "The Cable Guy : IPv6 Traffic over VPN Connection", *Technet Magazine*. Diakses 18 November 2007, dari Microsoft Technet.  
<http://www.microsoft.com/technet/technetmag/issues/2007/07/>
- [8] "How VPN Works," *Microsoft Technet Articles*, 28 Maret 2003. Diakses 19 November 2007, dari Microsoft Technet.  
<http://technet2.microsoft.com/windowsserver/en/library/7fd37ece-f0df-4c84-b284-c9cf9e1401981033.mspx>
- [9] Mark Gates, *et al.*, "Iperf User Docs", Maret 2003 . Diakses 30 November 2007, dari DAST.  
<http://dast.nlanr.net/projects/iperf>
- [10] Bouras, Christos., *et al.*, "QoS Issues in a Large-scale IPv6 Network", *Journal for Research Academic Institute University of Patras* : Mei 2005, hal. 2.

## DAFTAR PUSTAKA

- Bouras, Christos., *et al.*, *QoS Issues in a Large-scale IPv6 Network*, University of Patras, Mei 2005
- Davies, Joseph, *The Cable Guy : IPv6 Traffic over VPN Connection*, Microsoft Technet, Juli 2007
- Davies, Joseph, *Understanding IPv6*, (USA : Microsoft Press, 2003)
- Gates, Mark., *et al.*, “Iperf User Docs”, DAST, Maret 2003
- Feilner, Markus, *OpenVPN : Building and Integrating Virtual Private Networks*, (Birmingham : Packt Publishing, April 2006)
- Hagen, Silvia, *IPv6 Essentials : Second Edition*, (USA : O’Reilly, Mei 2006)
- Huang, S. M., Quincy Wu, Yi-Bing Lin, *Tunneling IPv6 through NAT with Teredo Mechanism*, IEEE Journal, 2005
- Huston, Geoff, *Anatomy : A Look Inside Network Address Translators*, The Internet Protocol Journal – Volume 7 Number 3, Agustus 2004. Diakses 5 Oktober 2007, dari Cisco.com.  
[http://www.cisco.com/web/about/ac123/ac147/archived\\_issues/ipj\\_7-3/anatomy.html](http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html)
- Malone, David, Niall Murphy, *IPv6 Network Administration*, (USA : O’Reilly, Maret 2005)
- Olifer, Victor, *Different Flavours of VPN : Technology and Applications*, JNT Association, Maret 2007
- Stockebarnd, Benedikt, *IPv6 in Practice : A Unixer’s Guide to the Next Generation Internet*, (Berlin : Springer, 2007)
- “Teredo Overview”, Microsoft Technet, 15 Mei 2007. Diakses 4 September 2007, dari Microsoft Technet.  
<http://www.microsoft.com/technet/network/teredo.mspx>
- “Teredo”, The IPv6 Portal, 2007. Diakses 4 Oktober 2007, dari The IPv6 Portal  
<http://www.ipv6tf.org/index.php?page=using/connectivity/teredo>

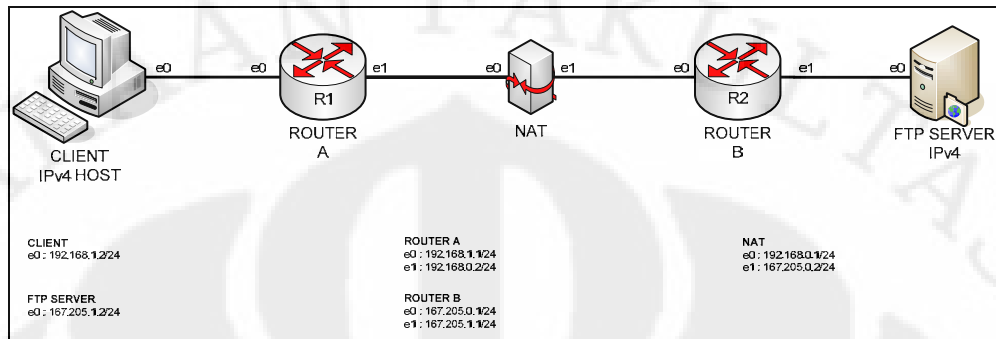
Wikipedia, "File Transfer Protocol", Maret 2006. Diakses 22 November 2007,  
dari Wikipedia  
[http://en.wikipedia.org/wiki/File\\_transfer\\_protocol](http://en.wikipedia.org/wiki/File_transfer_protocol)

Wikipedia, "Transmission Control Protocol", Maret 2006. Diakses 22 November  
2007, dari Wikipedia  
[http://en.wikipedia.org/wiki/Transmission\\_control\\_protocol](http://en.wikipedia.org/wiki/Transmission_control_protocol)

Wikipedia, "User Datagram Protocol", Maret 2006. Diakses 22 November 2007,  
dari Wikipedia  
[http://en.wikipedia.org/wiki/User\\_datagram\\_protocol](http://en.wikipedia.org/wiki/User_datagram_protocol)

# LAMPIRAN

## LAMPIRAN 1 : KONFIGURASI IPv4 MURNI



### Client IPv4

```
C:\> netsh interface ip add address "Local Area Connection"
192.168.1.2 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=192.168.1.1 gwmetric=0
```

### Router A (Ubuntu Linux)

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
# ifconfig eth1 192.168.0.2 netmask 255.255.255.0
# route add default gw 192.168.0.1
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

### NAT (Ubuntu Linux)

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0
# ifconfig eth1 167.205.0.2 netmask 255.255.255.0
# route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.2
# route add default gw 167.205.0.1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -t nat -A PREROUTING -o eth0 -j DNAT -- to-destination
192.168.0.2
```

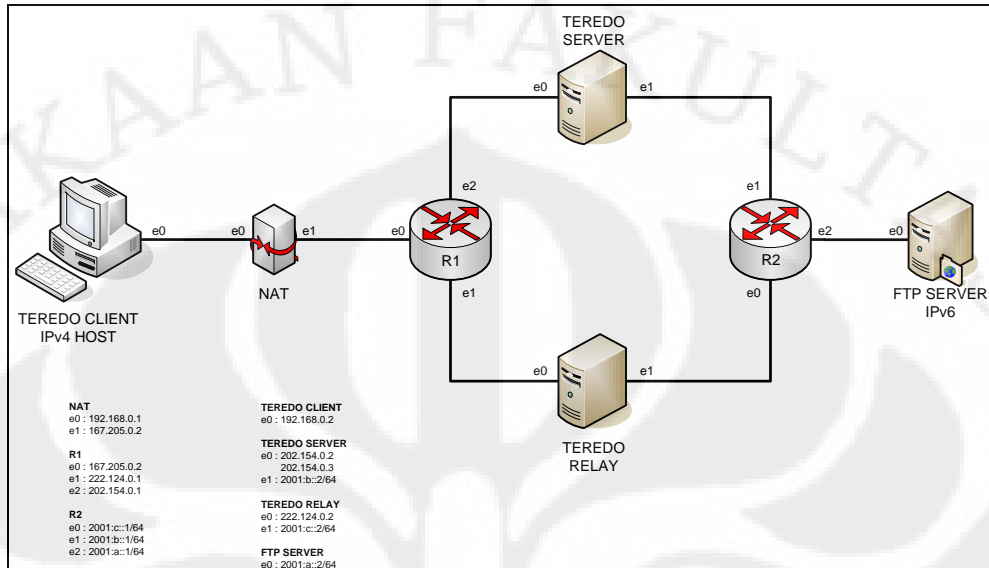
### Router B (Ubuntu Linux)

```
# ifconfig eth0 167.205.0.1 netmask 255.255.255.0
# ifconfig eth1 167.205.1.1 netmask 255.255.255.0
# route add default gw 167.205.0.2
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

### Server IPv6

```
C:\> netsh interface ip add address "Local Area Connection"
167.205.1.2 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=167.205.1.1 gwmetric=0
```

## LAMPIRAN 2 : KONFIGURASI TEREDO



### Teredo Client

```

C:\> netsh interface ip add address "Local Area Connection"
192.168.1.2 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=192.168.1.1 gwmetric=0
C:\> netsh interface ipv6 set teredo client 202.154.0.2
    
```

### NAT

```

# ifconfig eth0 192.168.0.1 netmask 255.255.255.0
# ifconfig eth1 167.205.0.2 netmask 255.255.255.0
# route add default gw 167.205.0.1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -t nat -A PREROUTING -o eth0 -j DNAT -- to-destination
192.168.0.2
    
```

### Router A (R1)

```

# ifconfig eth0 167.205.0.2 netmask 255.255.255.0
# ifconfig eth1 222.124.0.1 netmask 255.255.255.0
# ifconfig eth2 202.154.0.2 netmask 255.255.255.0
# echo "1" > /proc/sys/net/ipv4/ip_forward
    
```

### Teredo Server

```

# ifconfig eth0 202.154.0.2 netmask 255.255.255.0
# ifconfig eth0:0 202.154.0.3 netmask 255.255.255.0
# ifconfig eth1 inet6 2001:b::2/64
# route add default gw 202.154.0.1
# route -A inet6 add default gw 2001:b::1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
    
```

/usr/local/etc/miredo.conf

```
RelayType relay
InterfaceName teredo
BindAddress 222.124.0.2
BindPort 3545
Prefix 2001:0::
InterfaceMTU 1280
```

### **Teredo Relay**

```
# ifconfig eth0 222.124.0.2 netmask 255.255.255.0
# ifconfig eth1 inet6 2001:c::2/64
# route add default gw 222.124.0.1
# route -A inet6 add default gw 2001:c::1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

/usr/local/etc/miredo-server.conf

```
Prefix 2001:0::
InterfaceMTU 1280
ServerBindAddress 202.154.0.2
ServerBindAddress2 202.154.0.3
```

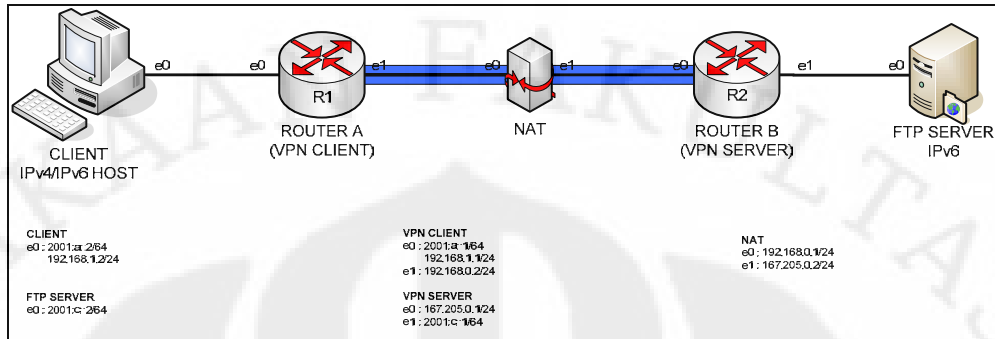
### **Router B (R2)**

```
# ifconfig eth0 inet6 2001:c::1/64
# ifconfig eth1 inet6 2001:b::1/64
# ifconfig eth2 inet6 2001:a::1/64
# route -A inet6 add 2001:0::/32 gw 2001:c::2
```

### **Server IPv6**

```
# ifconfig eth0 inet6 2001:a::2/64
# ifconfig -A inet6 add default gw 2001:a::1
```

### LAMPIRAN 3 : KONFIGURASI IPv6 VPN



#### Client IPv4/IPv6

```
C:\> netsh interface ip add address "Local Area Connection"
192.168.1.2 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=192.168.1.1 gwmetric=0
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:a::2
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:a::1
```

#### Router A / VPN Client (Ubuntu Linux)

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
# ifconfig eth0 inet6 2001:a::1/64
# ifconfig eth1 192.168.0.2 netmask 255.255.255.0
# route add default gw 192.168.0.1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

#### /etc/openvpn/openvpn\_client.conf

```
remote 167.205.0.1
dev tun
tun-ipv6
up "sh openvpn_client.up"
```

#### /etc/openvpn/openvpn\_client.up

```
localaddr="2001:b::2"
/sbin/ip -6 link set $dev up
/sbin/ip -6 addr add $localaddr dev $dev
/sbin/sysctl -w net.ipv6.conf.$dev.use_tempaddr=-1
/sbin/ip -6 route add default dev $dev
```

#### NAT (Ubuntu Linux)

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0
# ifconfig eth1 167.205.0.2 netmask 255.255.255.0
# route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.2
# route add default gw 167.205.0.1
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -t nat -A PREROUTING -o eth0 -j DNAT -- to-destination
192.168.0.2
```

#### Router B / VPN Server (Ubuntu Linux)

```
# ifconfig eth0 167.205.0.1 netmask 255.255.255.0
```



```
# ifconfig eth1 inet6 2001:c::1/64
# route add default gw 167.205.0.2
# echo "1" > /proc/sys/net/ipv4/ip_forward
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

```
/etc/openvpn/openvpn_server.conf
dev tun
tun-ipv6
up "sh openvpn_server.up"
```

```
/etc/openvpn/openvpn_server.up
localaddr="2001:b::2"
/sbin/ip -6 link set $dev up
/sbin/ip -6 addr add $localaddr dev $dev
/sbin/sysctl -w net.ipv6.conf.$dev.use_tempaddr=-1
/sbin/ip -6 route add default dev $dev
```

### Server IPv6

```
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:c::2
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:c::1
```

#### LAMPIRAN 4 : DATA HASIL UJI TOPOLOGI IPv4 MURNI

##### Lampiran Data Pengujian Koneksi TCP IPv4 Murni

| TCP Window Size | Throughput (Mbps) |       |       |       |       | $\Sigma$ | $\sigma$ |
|-----------------|-------------------|-------|-------|-------|-------|----------|----------|
|                 | 1                 | 2     | 3     | 4     | 5     |          |          |
| 8               | 82.50             | 87.40 | 83.20 | 87.50 | 81.60 | 84.44    | 2.81     |
| 16              | 92.20             | 92.30 | 92.20 | 92.40 | 92.10 | 92.24    | 0.11     |
| 32              | 94.20             | 94.20 | 94.20 | 94.40 | 94.00 | 94.20    | 0.14     |
| 64              | 90.30             | 90.10 | 90.50 | 90.80 | 90.60 | 90.46    | 0.27     |
| 128             | 90.10             | 90.30 | 90.40 | 90.00 | 90.80 | 90.32    | 0.31     |

##### Lampiran Data Pengujian Koneksi UDP IPv4 Murni

| UDP SIZE (bytes) | PARAMETER      | Data ke - n |       |       |       |       | $\Sigma$ | $\sigma$ |
|------------------|----------------|-------------|-------|-------|-------|-------|----------|----------|
|                  |                | 1           | 2     | 3     | 4     | 5     |          |          |
| 16               | Frame Loss (%) | 17.00       | 16.00 | 17.00 | 18.00 | 16.00 | 16.80    | 0.84     |
|                  | Jitter (ms)    | 4.00        | 4.00  | 3.49  | 3.51  | 3.37  | 3.67     | 0.30     |
| 80               | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 1.16        | 1.16  | 1.14  | 0.53  | 0.53  | 0.90     | 0.34     |
| 208              | Frame Loss (%) | 0.08        | 0.00  | 0.00  | 0.00  | 0.00  | 0.02     | 0.04     |
|                  | Jitter (ms)    | 1.51        | 2.07  | 1.45  | 1.60  | 2.18  | 1.76     | 0.34     |
| 464              | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 2.73        | 2.75  | 3.32  | 3.26  | 2.73  | 2.96     | 0.31     |
| 720              | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 4.62        | 4.57  | 4.60  | 4.56  | 4.58  | 4.58     | 0.02     |
| 976              | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 5.05        | 4.96  | 5.01  | 5.00  | 4.95  | 4.99     | 0.04     |
| 1232             | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 7.08        | 6.42  | 6.51  | 6.38  | 6.50  | 6.58     | 0.28     |
| 1452             | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 6.87        | 6.99  | 7.07  | 6.60  | 7.04  | 6.91     | 0.19     |

Lampiran Data Pengujian FTP IPv4 Murni

| FILE SIZE<br>(MB) | PARAMETER         | Data ke - <i>n</i> |          |          |          |          |          |          |          |          |          | $\Sigma$ | $\sigma$ |
|-------------------|-------------------|--------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
|                   |                   | 1                  | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       |          |          |
| 8                 | Throughput (KB/s) | 3,235.99           | 3,453.05 | 3,108.39 | 3,891.03 | 3,416.74 | 3,263.68 | 3,311.93 | 3,457.80 | 3,554.74 | 2,894.96 | 3,358.83 | 268.30   |
|                   | Transfer Time (s) | 2.39               | 2.25     | 2.44     | 2.30     | 2.39     | 2.74     | 2.60     | 2.07     | 2.52     | 2.53     | 2.42     | 0.19     |
|                   | Latency (ms)      | 0.27               | 0.25     | 0.28     | 0.22     | 0.25     | 0.27     | 0.26     | 0.25     | 0.25     | 0.30     | 0.26     | 0.02     |
| 16                | Throughput (KB/s) | 2,903.88           | 3,575.71 | 4,370.97 | 3,715.45 | 3,602.13 | 2,908.38 | 3,540.38 | 4,366.98 | 4,404.77 | 4,404.77 | 3,779.34 | 590.36   |
|                   | Transfer Time (s) | 3.46               | 3.51     | 3.23     | 4.41     | 3.36     | 3.71     | 3.38     | 3.69     | 3.42     | 3.42     | 3.56     | 0.33     |
|                   | Latency (ms)      | 0.30               | 0.24     | 0.20     | 0.24     | 0.24     | 0.30     | 0.24     | 0.20     | 0.20     | 0.20     | 0.24     | 0.04     |
| 32                | Throughput (KB/s) | 3,567.76           | 3,432.06 | 3,609.93 | 3,267.09 | 3,580.54 | 3,281.59 | 3,715.20 | 3,583.15 | 3,905.62 | 3,487.81 | 3,543.07 | 191.22   |
|                   | Transfer Time (s) | 7.53               | 8.55     | 7.97     | 8.49     | 7.62     | 7.55     | 7.64     | 8.99     | 8.17     | 7.12     | 7.96     | 0.58     |
|                   | Latency (ms)      | 0.25               | 0.25     | 0.24     | 0.27     | 0.24     | 0.27     | 0.24     | 0.25     | 0.22     | 0.25     | 0.25     | 0.01     |
| 64                | Throughput (KB/s) | 3,409.16           | 3,418.07 | 3,613.01 | 3,775.24 | 3,798.47 | 5,014.64 | 4,535.29 | 4,696.74 | 4,273.07 | 5,439.69 | 4,197.34 | 706.16   |
|                   | Transfer Time (s) | 16.25              | 17.70    | 12.97    | 15.26    | 14.61    | 10.92    | 11.51    | 10.56    | 12.01    | 9.59     | 13.14    | 2.69     |
|                   | Latency (ms)      | 0.26               | 0.26     | 0.24     | 0.23     | 0.23     | 0.17     | 0.19     | 0.19     | 0.20     | 0.16     | 0.21     | 0.04     |
| 128               | Throughput (KB/s) | 4,679.82           | 4,737.09 | 4,961.94 | 5,065.68 | 4,467.40 | 4,601.64 | 5,810.75 | 4,752.24 | 4,932.63 | 4,468.60 | 4,847.78 | 393.51   |
|                   | Transfer Time (s) | 21.03              | 21.41    | 21.07    | 21.11    | 23.77    | 23.93    | 16.83    | 22.98    | 22.98    | 25.02    | 22.01    | 2.30     |
|                   | Latency (ms)      | 0.19               | 0.18     | 0.18     | 0.17     | 0.20     | 0.19     | 0.15     | 0.18     | 0.18     | 0.20     | 0.18     | 0.01     |

## LAMPIRAN 5 : DATA HASIL UJI TOPOLOGI TEREDO

### Lampiran Data Pengujian Koneksi TCP Teredo

| TCP Window Size | Throughput (Mbps) |       |       |       |       | $\Sigma$ | $\sigma$ |
|-----------------|-------------------|-------|-------|-------|-------|----------|----------|
|                 | 1                 | 2     | 3     | 4     | 5     |          |          |
| 8               | 15.90             | 16.00 | 16.00 | 16.00 | 15.70 | 15.92    | 0.13     |
| 16              | 16.10             | 16.10 | 16.10 | 16.00 | 16.10 | 16.08    | 0.04     |
| 32              | 15.90             | 15.90 | 15.80 | 15.80 | 15.90 | 15.86    | 0.05     |
| 64              | 16.00             | 16.00 | 15.90 | 16.00 | 15.90 | 15.96    | 0.05     |
| 128             | 15.90             | 16.00 | 16.00 | 16.00 | 16.10 | 16.00    | 0.07     |

### Lampiran Data Pengujian Koneksi UDP Teredo

| UDP SIZE (bytes) | PARAMETER      | Data ke - n |       |       |       |       | $\Sigma$ | $\sigma$ |
|------------------|----------------|-------------|-------|-------|-------|-------|----------|----------|
|                  |                | 1           | 2     | 3     | 4     | 5     |          |          |
| 16               | Frame Loss (%) | 6.30        | 5.30  | 9.50  | 5.00  | 9.10  | 7.04     | 2.12     |
|                  | Jitter (ms)    | 15.27       | 15.59 | 15.20 | 15.37 | 15.37 | 15.36    | 0.15     |
| 80               | Frame Loss (%) | 5.90        | 5.90  | 5.90  | 5.30  | 5.90  | 5.78     | 0.27     |
|                  | Jitter (ms)    | 15.25       | 15.03 | 15.00 | 15.37 | 14.57 | 15.04    | 0.31     |
| 208              | Frame Loss (%) | 5.60        | 5.00  | 5.90  | 5.90  | 5.30  | 5.54     | 0.39     |
|                  | Jitter (ms)    | 2.45        | 2.58  | 1.94  | 1.91  | 2.99  | 2.37     | 0.45     |
| 464              | Frame Loss (%) | 4.30        | 5.00  | 5.00  | 5.00  | 5.30  | 4.92     | 0.37     |
|                  | Jitter (ms)    | 18.11       | 17.28 | 18.44 | 18.38 | 17.28 | 17.90    | 0.58     |
| 720              | Frame Loss (%) | 5.60        | 4.30  | 4.30  | 4.20  | 4.20  | 4.52     | 0.61     |
|                  | Jitter (ms)    | 19.14       | 19.90 | 20.17 | 19.73 | 21.05 | 20.00    | 0.70     |
| 976              | Frame Loss (%) | 0.07        | 0.00  | 0.00  | 1.00  | 0.00  | 0.21     | 0.44     |
|                  | Jitter (ms)    | 11.47       | 11.12 | 11.50 | 10.91 | 10.96 | 11.19    | 0.28     |
| 1232             | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 8.31        | 8.29  | 8.23  | 8.29  | 9.39  | 8.50     | 0.50     |
| 1452             | Frame Loss (%) | 0.00        | 0.00  | 0.00  | 0.00  | 0.00  | 0.00     | 0.00     |
|                  | Jitter (ms)    | 9.77        | 7.79  | 9.77  | 8.50  | 9.15  | 9.00     | 0.86     |

Lampiran Data Pengujian FTP Tereedo

| FILE SIZE<br>(MB) | PARAMETER         | Data ke - <i>n</i> |          |          |          |          |          |          |          |          |          | $\Sigma$ | $\sigma$ |
|-------------------|-------------------|--------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
|                   |                   | 1                  | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       |          |          |
| 8                 | Throughput (KB/s) | 622.20             | 777.32   | 843.78   | 948.88   | 927.55   | 1,113.15 | 926.12   | 1,108.09 | 992.37   | 1,042.91 | 930.24   | 151.93   |
|                   | Transfer Time (s) | 15.36              | 12.31    | 11.33    | 10.07    | 10.30    | 8.56     | 10.32    | 8.60     | 9.61     | 9.15     | 10.56    | 2.05     |
|                   | Latency (ms)      | 1.39               | 1.10     | 1.02     | 0.91     | 0.93     | 0.78     | 0.93     | 0.78     | 0.87     | 0.83     | 0.95     | 0.18     |
| 16                | Throughput (KB/s) | 1,230.17           | 1,084.71 | 878.37   | 1,029.01 | 980.39   | 982.69   | 1,058.84 | 1,103.66 | 899.97   | 1,036.17 | 1,028.40 | 102.16   |
|                   | Transfer Time (s) | 15.11              | 17.16    | 21.20    | 18.09    | 18.99    | 18.98    | 17.59    | 16.86    | 20.71    | 17.97    | 18.26    | 1.80     |
|                   | Latency (ms)      | 0.71               | 0.80     | 0.99     | 0.84     | 0.88     | 0.88     | 0.82     | 0.78     | 0.96     | 0.84     | 0.85     | 0.08     |
| 32                | Throughput (KB/s) | 1,041.34           | 1,057.19 | 1,032.27 | 997.71   | 987.36   | 1,043.64 | 1,062.99 | 1,117.31 | 1,061.89 | 1,045.89 | 1,044.76 | 36.09    |
|                   | Transfer Time (s) | 36.82              | 36.25    | 37.13    | 38.45    | 38.85    | 36.76    | 36.06    | 34.30    | 36.10    | 36.67    | 36.74    | 1.27     |
|                   | Latency (ms)      | 0.83               | 0.82     | 0.84     | 0.87     | 0.87     | 0.83     | 0.81     | 0.78     | 0.81     | 0.83     | 0.83     | 0.03     |
| 64                | Throughput (KB/s) | 1,109.41           | 1,061.27 | 1,055.08 | 1,119.33 | 1,115.72 | 1,022.50 | 1,118.01 | 1,047.55 | 1,022.81 | 1,058.32 | 1,073.00 | 39.04    |
|                   | Transfer Time (s) | 68.01              | 71.15    | 71.59    | 67.44    | 67.64    | 73.93    | 67.50    | 72.12    | 73.92    | 71.40    | 70.47    | 2.61     |
|                   | Latency (ms)      | 0.78               | 0.82     | 0.82     | 0.77     | 0.78     | 0.84     | 0.77     | 0.82     | 0.85     | 0.82     | 0.81     | 0.03     |
| 128               | Throughput (KB/s) | 1,097.00           | 1,109.71 | 1,098.86 | 1,082.65 | 1,081.38 | 1,115.14 | 1,080.17 | 1,090.45 | 1,052.07 | 1,122.54 | 1,093.00 | 20.50    |
|                   | Transfer Time (s) | 139.36             | 137.72   | 139.18   | 141.30   | 141.40   | 137.09   | 141.58   | 140.26   | 145.48   | 136.20   | 139.96   | 2.70     |
|                   | Latency (ms)      | 0.79               | 0.78     | 0.79     | 0.80     | 0.80     | 0.78     | 0.80     | 0.79     | 0.82     | 0.77     | 0.79     | 0.01     |

**LAMPIRAN 6 : DATA HASIL UJI TOPOLOGI IPv6 VPN**

Lampiran Data Pengujian Koneksi TCP IPv6 VPN

| TCP Window Size | Throughput (Mbps) |       |       |       |       | $\Sigma$ | $\sigma$ |
|-----------------|-------------------|-------|-------|-------|-------|----------|----------|
|                 | 1                 | 2     | 3     | 4     | 5     |          |          |
| 8               | 60.70             | 60.70 | 60.10 | 64.10 | 61.80 | 61.48    | 1.59     |
| 16              | 85.00             | 85.10 | 85.20 | 85.10 | 85.20 | 85.12    | 0.08     |
| 32              | 88.60             | 88.70 | 88.50 | 88.60 | 88.60 | 88.60    | 0.07     |
| 64              | 85.70             | 85.80 | 85.60 | 85.60 | 85.70 | 85.68    | 0.08     |
| 128             | 85.50             | 85.50 | 85.40 | 85.40 | 85.60 | 85.48    | 0.08     |

Lampiran Data Pengujian Koneksi UDP IPv6 VPN

| UDP SIZE (bytes) | PARAMETER      | Data ke - n |      |      |      |      | $\Sigma$ | $\sigma$ |
|------------------|----------------|-------------|------|------|------|------|----------|----------|
|                  |                | 1           | 2    | 3    | 4    | 5    |          |          |
| 16               | Frame Loss (%) | 0.10        | 0.00 | 0.00 | 0.00 | 0.00 | 0.02     | 0.04     |
|                  | Jitter (ms)    | 0.59        | 0.33 | 0.68 | 0.95 | 0.96 | 0.70     | 0.26     |
| 80               | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 1.16        | 0.94 | 0.94 | 1.16 | 1.12 | 1.06     | 0.11     |
| 208              | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 1.59        | 1.58 | 1.83 | 1.79 | 1.59 | 1.68     | 0.12     |
| 464              | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 3.02        | 3.00 | 3.02 | 2.86 | 3.03 | 2.99     | 0.07     |
| 720              | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 4.32        | 4.25 | 4.23 | 3.76 | 4.35 | 4.18     | 0.24     |
| 976              | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 5.35        | 5.16 | 5.17 | 5.14 | 5.48 | 5.26     | 0.15     |
| 1232             | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 6.70        | 7.75 | 6.65 | 6.65 | 6.63 | 6.88     | 0.49     |
| 1452             | Frame Loss (%) | 0.00        | 0.00 | 0.00 | 0.00 | 0.00 | 0.00     | 0.00     |
|                  | Jitter (ms)    | 6.86        | 7.28 | 7.25 | 7.54 | 7.35 | 7.26     | 0.25     |

Lampiran Data Pengujian FTP IPv6 VPN

| FILE SIZE<br>(MB) | PARAMETER         | Data ke - $n$ |          |          |          |          |          |          |          |          |          | $\Sigma$ | $\sigma$ |
|-------------------|-------------------|---------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
|                   |                   | 1             | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       |          |          |
| 8                 | Throughput (KB/s) | 2,864.65      | 4,256.14 | 3,867.34 | 3,417.68 | 3,646.67 | 2,365.97 | 2,704.88 | 3,406.49 | 2,998.45 | 3,713.67 | 3,324.19 | 582.50   |
|                   | Transfer Time (s) | 3.14          | 2.11     | 2.32     | 2.63     | 2.46     | 3.80     | 3.33     | 2.63     | 3.00     | 2.42     | 2.78     | 0.52     |
|                   | Latency (ms)      | 0.40          | 0.27     | 0.30     | 0.33     | 0.32     | 0.48     | 0.43     | 0.34     | 0.38     | 0.31     | 0.36     | 0.07     |
| 16                | Throughput (KB/s) | 2,671.42      | 2,702.69 | 3,023.89 | 4,145.51 | 4,378.28 | 3,510.83 | 4,426.51 | 4,358.10 | 4,704.17 | 4,846.95 | 3,876.83 | 828.39   |
|                   | Transfer Time (s) | 6.56          | 6.49     | 5.80     | 4.23     | 4.00     | 4.99     | 3.82     | 3.88     | 3.62     | 3.58     | 4.70     | 1.18     |
|                   | Latency (ms)      | 0.43          | 0.43     | 0.38     | 0.28     | 0.26     | 0.33     | 0.26     | 0.26     | 0.24     | 0.24     | 0.31     | 0.08     |
| 32                | Throughput (KB/s) | 4,597.88      | 3,981.66 | 4,172.66 | 4,350.88 | 5,013.59 | 3,823.99 | 4,893.06 | 4,271.33 | 4,801.15 | 4,807.63 | 4,471.38 | 409.92   |
|                   | Transfer Time (s) | 7.15          | 8.93     | 8.33     | 7.80     | 6.94     | 8.92     | 7.27     | 8.45     | 7.49     | 7.05     | 7.83     | 0.77     |
|                   | Latency (ms)      | 0.25          | 0.29     | 0.27     | 0.26     | 0.23     | 0.30     | 0.23     | 0.27     | 0.24     | 0.24     | 0.26     | 0.02     |
| 64                | Throughput (KB/s) | 5,526.58      | 4,680.89 | 5,007.06 | 4,709.54 | 4,711.27 | 4,801.59 | 4,753.86 | 4,816.44 | 5,104.29 | 4,668.48 | 4,878.00 | 269.25   |
|                   | Transfer Time (s) | 12.52         | 14.81    | 13.81    | 14.79    | 14.50    | 14.48    | 14.75    | 14.47    | 13.58    | 14.31    | 14.20    | 0.72     |
|                   | Latency (ms)      | 0.21          | 0.25     | 0.23     | 0.24     | 0.24     | 0.24     | 0.24     | 0.24     | 0.22     | 0.24     | 0.24     | 0.01     |
| 128               | Throughput (KB/s) | 5,867.55      | 5,107.33 | 5,198.42 | 5,677.78 | 5,094.59 | 4,649.47 | 6,383.45 | 5,165.03 | 6,061.98 | 4,483.04 | 5,368.86 | 611.95   |
|                   | Transfer Time (s) | 22.07         | 25.38    | 25.99    | 23.69    | 26.70    | 30.06    | 21.17    | 26.35    | 23.07    | 31.23    | 25.57    | 3.26     |
|                   | Latency (ms)      | 0.19          | 0.22     | 0.22     | 0.20     | 0.22     | 0.25     | 0.18     | 0.22     | 0.19     | 0.26     | 0.22     | 0.02     |