

**STUDI KUALITAS *VIDEO STREAMING* MENGGUNAKAN  
VLC DAN HELIX *STREAMING SERVER* PADA JARINGAN  
*TESTBED* IPV6 DAN *TUNNELING 6TO4* BERDASARKAN  
PARAMETER *BIT RATE* DAN *FRAME RATE***

**SKRIPSI**

**OLEH**

**IHTIAR NUR**

**04 04 03 049 Y**



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

**STUDI KUALITAS *VIDEO STREAMING* MENGGUNAKAN  
VLC DAN HELIX *STREAMING SERVER* PADA JARINGAN  
*TESTBED* IPV6 DAN *TUNNELING 6TO4* BERDASARKAN  
PARAMETER *BIT RATE* DAN *FRAME RATE***

**SKRIPSI**

**OLEH**

**IHTIAR NUR**

**04 04 03 049 Y**



**SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN  
PERSYARATAN MENJADI SARJANA TEKNIK**

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

## **PERNYATAAN KEASLIAN SKRIPSI**

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul :

**STUDI KUALITAS *VIDEO STREAMING* MENGGUNAKAN *VLC* DAN  
*HELIX STREAMING SERVER* PADA JARINGAN *TESTBED* *IPV6* DAN  
*TUNNELING 6TO4* BERDASARKAN PARAMETER *BIT RATE* DAN  
*FRAME RATE***

yang dibuat untuk melengkapi sebagian persyaratan sebagai 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 atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang merupakan sumber informasinya dicantumkan sebagaimana mestinya .

Depok, 14 Juli 2008

Ihtiar Nur

NPM 04 04 03 049 Y

## PERSETUJUAN

Skripsi dengan judul :

**STUDI KUALITAS *VIDEO STREAMING* MENGGUNAKAN VLC DAN  
HELIX *STREAMING SERVER* PADA JARINGAN *TESTBED* IPV6 DAN  
*TUNNELING* 6TO4 BERDASARKAN PARAMETER *BIT RATE* DAN  
*FRAME RATE***

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, 14 Juli 2008

Dosen Pembimbing

Ir. Endang Sriningsih, MT

NIP. 130 781 318

## UCAPAN TERIMA KASIH

Puji syukur kehadiran Tuhan Yang Maha Esa atas berkat dan rahmat-Nya skripsi ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada:

### **1. Ir. Endang Sriningsih, MT**

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

### **2. Ir. Adhi Yunirto L.Y., M.Kom**

Selaku Kepala Sub Direktorat PPSI Universitas Indonesia yang telah memberikan izin dan pengarahan dalam menggunakan fasilitas yang tersedia di PPSI.

### **3. Bpk. Awaluddin beserta tim**

Yang telah banyak memberikan bantuan dan pengarahan agar dalam pengerjaan skripsi ini.

## ABSTRAK

Ihtiar Nur  
NPM 04 04 03 049 Y

Dosen Pembimbing  
Ir. Endang Sriningsih, MT

Departemen Teknik Elektro

### **STUDI KUALITAS *VIDEO STREAMING* MENGGUNAKAN VLC DAN HELIX *STREAMING SERVER* PADA JARINGAN *TESTBED* IPV6 DAN *TUNNELING 6TO4* BERDASARKAN PARAMETER *BIT RATE* DAN *FRAME RATE***

#### **ABSTRAK**

Skripsi ini menguji coba penerapan video *streaming* dengan menggunakan aplikasi *VideoLAN Client* (VLC) dan *Helix Streaming Server* pada jaringan *testbed*. Konfigurasi topologi jaringan yang diterapkan adalah jaringan IPv6, jaringan IPv4, jaringan menggunakan metode *tunneling 6to4*.

Jaringan lokal yang digunakan menggunakan perangkat dua buah *mobile computer*, *router cisco 3800* dan *3700 series*, dan sebuah *layer 2 switch*. Parameter ukur yang diamati dibatasi pada variabel yang mempengaruhi kualitas *video streaming* seperti *bit rate* dan *frame rate*.

Uji coba VLC dilakukan dengan melakukan *streaming file* berformat *.mpg* dan *.mp4* yang masing-masing berdurasi 60 detik. *Streaming* dilakukan 10 kali untuk masing-masing *file* pada tiap konfigurasi jaringan. Dari hasil uji coba didapatkan bahwa terjadi penurunan *bit rate* pada sisi *client* jika dibandingkan dengan *bit rate* asli pada *server*. Penurunan *bit rate* di mana VLC berperan sebagai *server* adalah sebagai berikut: jaringan IPv4 (11.5%), jaringan IPv6 (9.98%), dan jaringan *tunneling 6to4* (11.43%). Sementara *frame rate file* asli dan hasil *streaming* tidak mengindikasikan adanya penurunan di mana aslinya memiliki *frame rate* 29.97 fps dan hasil *streaming* rata-rata memiliki *frame rate* 30 fps. Sementara *streaming* dengan menggunakan *Helix Streaming Server* dan *Real Player* pada *client* dengan *file-file* berformat *.mp4* dan *.rm* yang divariasikan besar *bit rate* dan *frame rate*-nya. Hasilnya adalah tetap terjadi penurunan *bit rate* sebesar untuk *file .rm* (0.29 %) untuk tiap topologi, namun *bit rate* tetap stabil untuk *file .mp4*. Sementara untuk *frame rate* terjadi penurunan pada hasil *streaming* sebagai berikut: jaringan IPv4 (3%), jaringan IPv6 (4%), dan jaringan *tunneling 6to4* (4%).

**Kata kunci : Video Streaming, IPv4, IPv6, Tunneling, 6to4, VLC, Helix**

## ABSTRACT

Ihtiar Nur  
NPM 04 04 03 049 Y

Counselor  
Ir. Endang Sriningsih, MT

Electrical Engineering Departement

### **STUDY OF VIDEO STREAMING QUALITY USING VLC AND HELIX STREAMING SERVER ON IPV6 NETWORK AND 6TO4 TUNNELING TESTBED BASED ON BIT RAE AND FRAME RATE AS THE PARAMETERS**

#### **ABSTRACT**

This thesis tests the implementation of video streaming with VideoLan Client (VLS) and Helix Streaming Server in a local network. The network topology that implemented is IPv6, IPv4 and 6to4 tunneling method.

The local network uses 2 mobile computers, cisco 3800 and 3700 routers, and a 2 layer switch. The measurement is focussed on the variables that influence the video streaming quality such as bit rate and frame rate.

The test is done by streaming the .mpg and .mp4 files with 60 seconds duration each. Streaming is done 10 times for each file and for each configuration. The test using VLC, the results describe there is bit-rate-reduction at client's side. The bit-rate-reduction at client side for IPv4 network (11.5%), IPv6 network (9.98%), and 6to4 tunneling method (11.43%). Meanwhile the frame rate of the origin file does not change much as the original file's frame rate is 29.97 and the streaming file's frame rate is 30. Mean while, the test using Helix Streaming Server and Real Player as the client software also produces 0.29 % of bit-rate-reduction at client side for all types of topology. The resulting frame rate also reduces 3 % on IPv4 network and 4 % on both IPv6 and 6to4 network.

**Keywords : Video Streaming, IPv4, IPv6, Tunneling, 6to4, VLC, Helix**

# DAFTAR ISI

	Halaman
PERNYATAAN KEASLIAN SKRIPSI	ii
PERSETUJUAN	iii
UCAPAN TERIMA KASIH	iv
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR GAMBAR	x
DAFTAR TABEL	xii
DAFTAR LAMPIRAN	xiii
DAFTAR SINGKATAN	xiv
DAFTAR ISTILAH	xv
BAB I PENDAHULUAN	1
1.1. LATAR BELAKANG	1
1.2. PERUMUSAN MASALAH	2
1.3. TUJUAN PENULISAN	2
1.4. BATASAN MASALAH	2
1.5. SISTEMATIKA PENULISAN	3
BAB II PROTOKOL <i>ROUTING</i> IPV6	4
DAN APLIKASI <i>VIDEO STREAMING</i>	4
2.1. SEJARAH DAN LATAR BELAKANG PERKEMBANGAN IPV6	4
2.2. MEKANISME IPV6	5
2.2.1. Perbandingan IPv6 dan IPv4	5
2.2.2. Tipe Alamat Pada IPv6	7
2.3. MEKANISME <i>TUNNELING</i>	7
2.3.1. <i>Tunneling</i> Manual	8
2.3.2. <i>Tunneling</i> Otomatis	9



2.3.2.1. <i>Tunneling 6to4</i>	9
2.4. APLIKASI <i>VIDEO STREAMING</i>	11
2.4.1. Protokol Pada <i>Video Streaming</i>	12
2.4.2. Parameter-Parameter Aplikasi <i>Video Streaming</i>	14
2.4.2.1. <i>Bit rate</i>	14
2.4.2.2. <i>Frame Rate</i>	15
BAB III TOPOLOGI KONFIGURASI JARINGAN DAN PROSES <i>STREAMING</i>	16
3.1 TOPOLOGI JARINGAN DAN SKENARIO PROSES <i>STREAMING</i>	16
3.2 KONFIGURASI JARINGAN	18
3.2.1. Konfigurasi Jaringan IPv6 Murni	18
3.2.2. Konfigurasi Jaringan IPv4 Murni	19
3.2.3. Konfigurasi <i>Tunneling 6to4</i>	19
3.3 KELENGKAPAN KONFIGURASI JARINGAN	20
3.4 METODE PENGAMBILAN DATA	21
BAB IV ANALISIS TOPOLOGI JARINGAN DAN HASIL <i>STREAMING</i>	24
4.1. PROSES PENGAMBILAN DATA	24
4.1.1. Pengambilan Data VLC	24
4.1.2. Pengambilan Data <i>Helix Streaming Server Dan Real Player</i>	25
4.2. DOKUMENTASI JARINGAN	27
4.2.1. Jaringan IPv6	27
4.2.2. Jaringan IPv4	28
4.2.3. Jaringan 6to4	28
4.3. DOKUMENTASI <i>SERVER</i> DAN <i>CLIENT</i>	30
4.3.1. <i>VLC Server dan Client</i>	30
4.3.2. <i>Helix Streaming Server dan Real Player</i>	33
4.4. ANALISIS <i>BIT RATE</i> VLC PADA TIAP TOPOLOGI JARINGAN	35
4.5. ANALISIS <i>FRAME RATE</i> VLC PADA TIAP TOPOLOGI JARINGAN	40
4.6. ANALISIS <i>BIT RATE</i> HELIX PADA TIAP TOPOLOGI JARINGAN	41
4.7. ANALISIS <i>FRAME RATE</i> HELIX PADA TIAP TOPOLOGI JARINGAN	43
4.8. PERBANDINGAN KINERJA KESELURUHAN <i>STREAMING</i> MENGUNAKAN VLC DAN HELIX <i>STREAMING SERVER</i>	45
BAB V KESIMPULAN	48

DAFTAR ACUAN

49

LAMPIRAN

50



## DAFTAR GAMBAR

	Halaman
Gambar 2.1. Enkapsulasi <i>Packet</i> IPv6 dalam IPv4	8
Gambar 2.2. Format Alamat 6to4	10
Gambar 2.3. Mekanisme dan Komponen 6to4	11
Gambar 2.4. Arsitektur <i>Video Streaming</i>	12
Gambar 2.5. Konektivitas Protokol pada <i>Media Streaming</i> .	13
Gambar 3.1. Topologi Umum Jaringan Test-bed Pengujian	16
Gambar 3.2. <i>Router</i> CISCO 3725 Series	17
Gambar 3.3. <i>Router</i> CISCO 3845 Series	17
Gambar 3.4. <i>Catalyst</i> CISCO 3550 Series	17
Gambar 3.5. Topologi Jaringan Test-bed IPv6	18
Gambar 3.6. Topologi Jaringan Test-bed IPv4	19
Gambar 3.7. Topologi Jaringan 6to4	20
Gambar 4.1. <i>Client</i> IPv6	27
Gambar 4.2. <i>Server</i> IPv6	27
Gambar 4.3. <i>Client</i> IPv4	28
Gambar 4.4. <i>Server</i> IPv4	28
Gambar 4.5. <i>Client</i> 6to4	29
Gambar 4.6. <i>Server</i> 6to4	29
Gambar 4.7. <i>Setting</i> VLC <i>Server</i> dan <i>Client</i> pada Topologi IPv4	30
Gambar 4.8. Hasil Packet Sniffing Menggunakan Wireshark pada VLC <i>Client</i> Topologi IPv4	31
Gambar 4.9. <i>Setting</i> VLC <i>Server</i> dan <i>Client</i> pada Topologi IPv6 dan 6to4	32

Gambar 4.10. Hasil Packet Sniffing Menggunakan Wireshark pada VLC Client Topologi IPv6	33
Gambar 4.11. Hasil Packet Sniffing Menggunakan Wireshark pada VLC Client Topologi 6to4	33
Gambar 4.12. <i>Setting</i> Real Player Untuk Melakukan <i>Streaming</i>	34
Gambar 4.13. Hasil Packet Sniffing Real Player Pada topologi IPv4	34
Gambar 4.14. Hasil Packet Sniffing Real Player Pada topologi Ipv6	35
Gambar 4.15. Hasil Packet Sniffing Real Player Pada topologi 6to4	35
Gambar 4.16. Grafik VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada IPv4	36
Gambar 4.17. Grafik VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada Ipv6	37
Gambar 4.18. Grafik VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada 6to4	38
Gambar 4.19. Grafik <i>Real Player</i> Perbandingan <i>Bit rate</i> Asli dengan <i>Bt Rate Streaming</i>	42

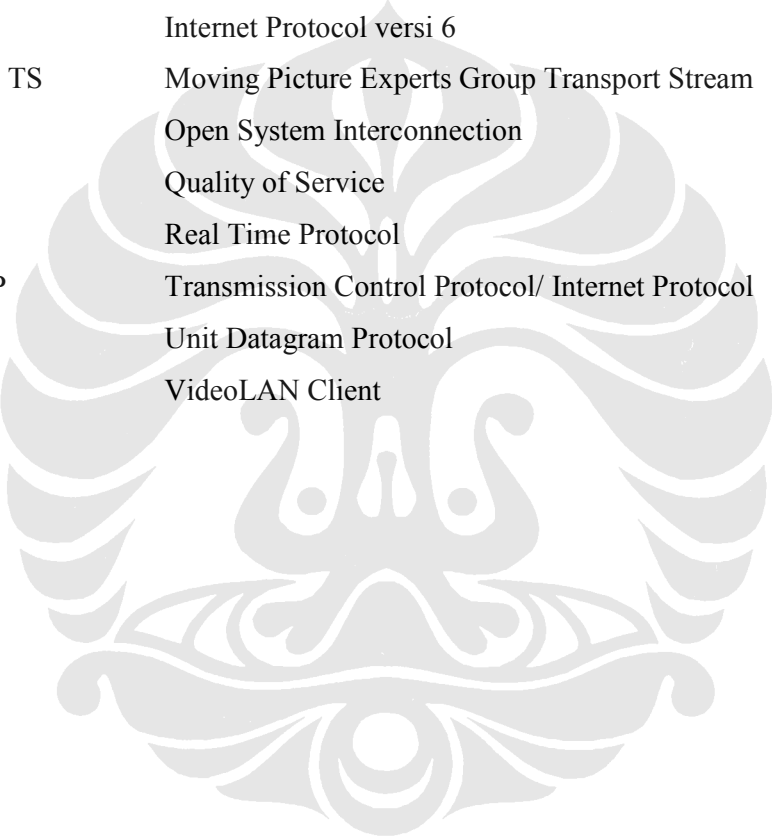
## DAFTAR TABEL

	Halaman
Tabel 2.1. Perbedaan IPv4 dengan IPv6	6
Tabel 2.2. Hal-Hal yang Ekuivalen pada IPv6 dan IPv4	6
Tabel 3.1. Alamat 6to4 Pada <i>Interface Router</i>	20
Tabel 4.1. Tabel VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada IPv4	36
Tabel 4.2. Tabel VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada IPv6	36
Tabel 4.3. Tabel VLC Perbandingan Send <i>Bit rate</i> dengan <i>Stream Rate</i> pada IPv6	37
Tabel 4.4. Tabel VLC Penurunan <i>Bit rate</i> (kb/s) dan Reduksi <i>Bit rate</i> (%)	38
Tabel 4.5. Tabel VLC Perbandingan Rata-Rata <i>Frame Rate</i> Seluruh Topologi	40
Tabel 4.6. Tabel Helix-Real Perbandingan <i>Bit rate</i> asli dengan <i>Bit rate Client</i>	41
Tabel 4.7. Tabel Helix-Real <i>Frame Rate</i> Pada Topologi IPv4	43
Tabel 4.8. Tabel Helix-Real <i>Frame Rate</i> Pada Topologi IPv6	43
Tabel 4.9. Tabel Helix-Real <i>Frame Rate</i> Pada Topologi 6to4	44

## DAFTAR LAMPIRAN

	Halaman
LAMPIRAN 1: KONFIGURASI JARINGAN IPV6 MURNI	50
LAMPIRAN 2: KONFIGURASI JARINGAN IPv4 MURNI	51
LAMPIRAN 3: KONFIGURASI JARINGAN 6to4	52
LAMPIRAN 4: RATA-RATA DATA <i>BIT RATE</i> PADA VLC <i>CLIENT</i>	54
LAMPIRAN 5: RATA-RATA DATA <i>FRAME RATE</i> PADA VLC <i>CLIENT</i>	54
LAMPIRAN 6: RATA-RATA DATA FRAME RATE PADA HELIX-REAL PLAYER	55
LAMPIRAN 7: CONTOH PENCATATAN SAMPEL DATA <i>BIT RATE</i> VLC TIAP 5 DETIK	56
LAMPIRAN 8: CONTOH PENCATATAN SAMPEL DATA <i>FRAME RATE</i> VLC TIAP 5 DETIK	57
LAMPIRAN 9: CONTOH PENCATATAN SAMPEL DATA <i>BIT RATE</i> HELIX- REAL PLAYER TIAP 5 DETIK	57

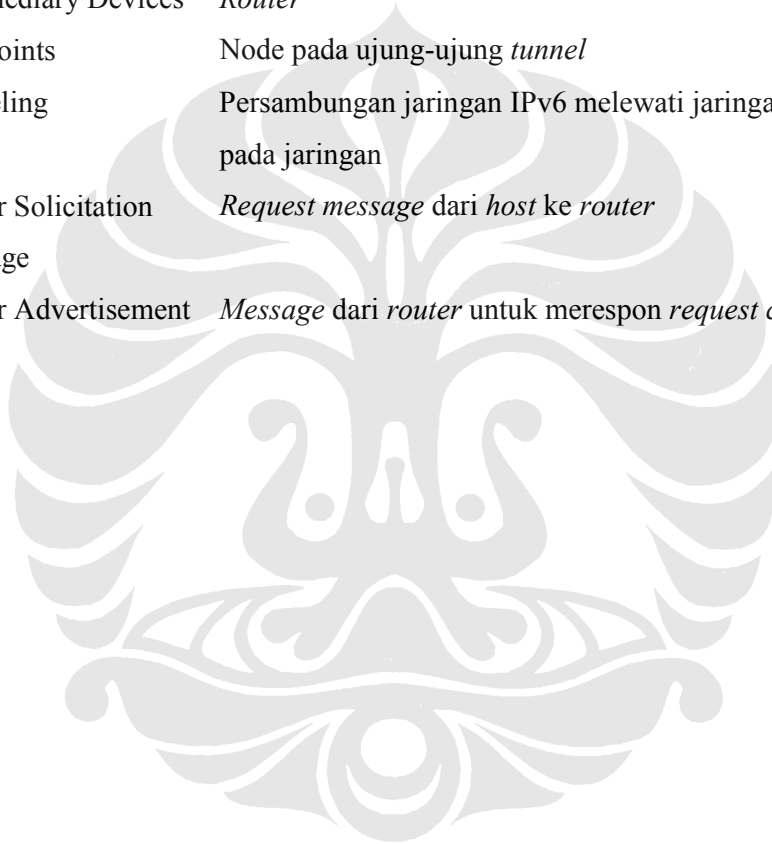
## DAFTAR SINGKATAN



ARP	Address Resolution Protocol
HTTP	Hiper Text Transfer Protocol
IETF	Internet Engineering Task Force
IPv4	Internet Protocol versi 4
IPv6	Internet Protocol versi 6
MPEG TS	Moving Picture Experts Group Transport Stream
OSI	Open System Interconnection
QoS	Quality of Service
RTP	Real Time Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
UDP	Unit Datagram Protocol
VLC	VideoLAN Client

## DAFTAR ISTILAH

<i>Video Streaming</i>	Proses pengiriman data <i>real time</i>
<i>Bit rate</i>	Banyaknya bit yang diproses suatu <i>interface</i> per satuan waktu
<i>Frame Rate</i>	Banyaknya <i>frame</i> yang ditampilkan per satuan waktu
Intermediary Devices	<i>Router</i>
End Points	Node pada ujung-ujung <i>tunnel</i>
Tunneling	Persambungan jaringan IPv6 melewati jaringan IPv4 pada jaringan
Router Solicitation Message	<i>Request message</i> dari <i>host</i> ke <i>router</i>
Router Advertisement	<i>Message</i> dari <i>router</i> untuk merespon <i>request client</i>





# BAB I

## PENDAHULUAN

### 1.1. LATAR BELAKANG

Jaringan internet, yang pada dasarnya dibangun dan dirancang merujuk kepada model referensi Open Systems Interconnection (OSI) dan model protokol *Transmission Control Protocol/Internet Protocol* (TCP/IP), pada perkembangannya telah mengalami kemajuan yang sangat pesat dalam hal penggunaannya. Namun tentunya hal ini juga mengarah kepada semakin terbatasnya kemampuan protokol yang telah digunakan, terutama dalam hal ini adalah *routing protocol* seperti *Internet Protocol version 4* (IPv4). IPv4 pada dasarnya menggunakan metode pengalamatan berdasarkan 32-bit mampu mengakomodasi pengalamatan *end device* atau *host* sejumlah  $2^{32}$  ( $\approx 4,2 \times 10^9$ ). Seiring berjalannya waktu dan pesatnya penggunaan internet jumlah pengalamatan tersebut menjadi sangat terbatas sehingga dibutuhkan suatu standar baru pada *routing protocol* yang mampu mengakomodasi jumlah pengalamatan yang lebih banyak. Berangkat dari masalah ini, *Internet Protocol version 6* (IPv6) kemudian dikembangkan dan dijadikan sebagai standar baru karena IPv6 mampu mengakomodasi pengalamatan *host* hingga sejumlah  $2^{128}$  ( $\approx 3,4 \times 10^{38}$ ).

IPv6 dirancang sedemikian rupa agar memiliki kinerja secara keseluruhan yang lebih baik dari IPv4 seperti halnya dalam proses pengiriman *packet*, skalabilitas, kualitas *service* dan *security*. Untuk menunjang hal-hal tersebut IPv6 menawarkan metode pengalamatan 128-bit, kemampuan *authentication* dan *privacy*, serta penyediaan opsi dan tempat untuk tambahan fitur-fitur yang masih akan dikembangkan [1]. Hal-hal tersebut akan dibahas lebih lanjut pada bab 2.

IPv6, yang telah menjadi penerus dari IPv4, terus dikembangkan dan telah diterapkan pada sejumlah area. Saat ini penerapan IPv6 masih bekerja berdampingan dengan IPv4 dan akan menggantikan sepenuhnya di masa mendatang. Berbagai metode telah diteliti dan dapat diterapkan untuk menunjang mekanisme transisi tersebut, beberapa diantaranya adalah metode *Dual Stack*, *Tunneling*, dan *Translation*.

Salah satu aplikasi yang dapat diimplementasikan sekaligus menjadi uji coba dalam IPv6 adalah *video streaming*. Tidak hanya aplikasi ini telah menjadi salah satu alternatif dalam melakukan *multimedia connectivity*, namun juga telah menjadi sarana lain dalam melakukan *streaming communication*.

## 1.2. PERUMUSAN MASALAH

Penerapan IPv6 dalam jaringan yang mayoritas masih menerapkan IPv4 dapat dijumpai dengan menggunakan metode *tunneling*. Beberapa metode *tunneling* yang lazim digunakan adalah ISATAP, 6to4, 6over4, Teredo, GRE, dll. Karenanya penting untuk memahami cara kerja dan karakteristik OSI dan TCP/IP (terutama pada *layer Internet/Network*, *routing protocol* IPv4 dan IPv6 murni), konfigurasi jaringan, konsep beserta mekanisme metode *tunneling*, aplikasi yang akan digunakan, dan lain-lain.

## 1.3. TUJUAN PENULISAN

Skripsi ini bertujuan untuk melakukan studi terhadap kualitas *video streaming* pada jaringan IPv4 murni, IPv6 murni dan *tunneling* 6to4. Berdasarkan parameter *bit rate* dan *frame rate* sebagai parameter yang diamati.

## 1.4. BATASAN MASALAH

Pengujian menggunakan *test bed* akan terfokus melakukan uji coba aplikasi *video streaming* yang diterapkan pada konfigurasi jaringan IPv4 murni, IPv6 murni, dan metode *tunneling* 6to4.

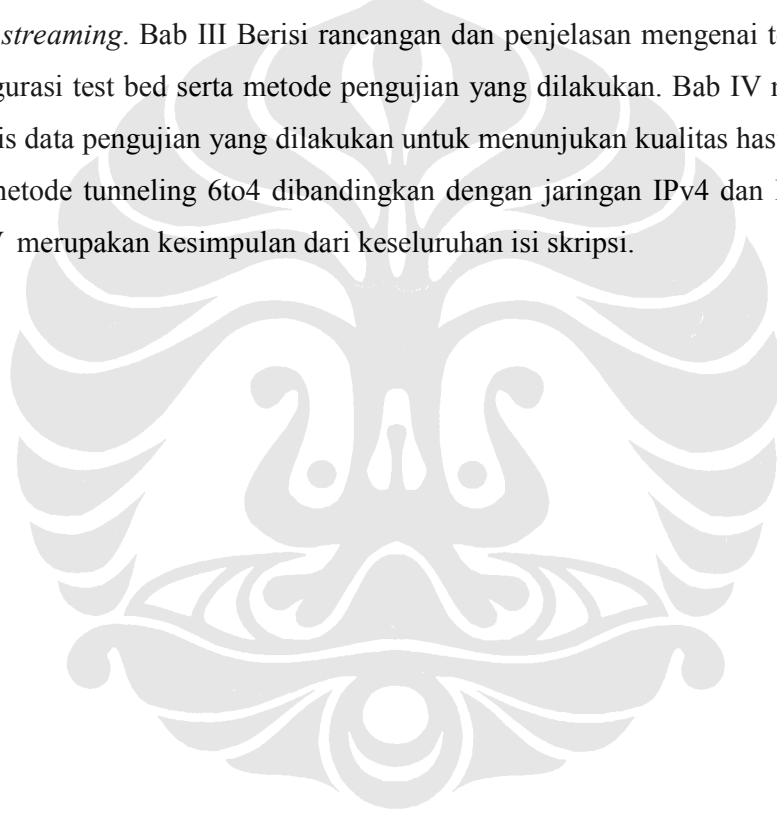
Rancangan *test bed* yang digunakan adalah jaringan lokal yang terdiri dari dua buah *Mobile Computer* sebagai satu *server* dan satu *client*, yang terhubung dengan *intermediary devices* berupa router CISCO 3725 dan 3845.

Aplikasi *video streaming* uji yang akan diimplementasikan pada *test bed* adalah aplikasi VideoLAN *Client* pada sisi *server* dan *client*, dan juga *Helix streaming server* sebagai *server* serta *Real Player* sebagai *client*.

Parameter-parameter yang akan diperhatikan dan dijadikan data untuk menganalisis kinerja jaringan adalah *bit rate* dan *frame rate*.

### **1.5. SISTEMATIKA PENULISAN**

Skripsi ini terdiri atas lima bab. Bab I merupakan pendahuluan yang berisi gambaran tentang skripsi yang terdiri dari latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, dan sistematika penulisan. Bab II berisi teori-teori dasar mengenai IPv4, IPv6, metode transisi dari IPv4 ke IPv6, dan aplikasi *video streaming*. Bab III Berisi rancangan dan penjelasan mengenai topologi dan konfigurasi test bed serta metode pengujian yang dilakukan. Bab IV menjelaskan analisis data pengujian yang dilakukan untuk menunjukkan kualitas hasil *streaming* dari metode tunneling 6to4 dibandingkan dengan jaringan IPv4 dan IPv6 murni. Bab V merupakan kesimpulan dari keseluruhan isi skripsi.



## BAB II

### PROTOKOL *ROUTING* IPV6

### DAN APLIKASI *VIDEO STREAMING*

#### 2.1. SEJARAH DAN LATAR BELAKANG PERKEMBANGAN IPV6

Sejak awal tahun 1990-an, organisasi *Internet Engineering Task Force* (IETF) mulai menyadari bahwa suatu saat *routing protocol* IPv4 akan mengalami keterbatasan dalam penyediaan alamat *Internet Protocol* (IP) dan mulai mencari suatu *routing protocol* pengganti yang dapat menyediakan jumlah alamat IP lebih banyak. Hal inilah yang kemudian mengawali proses pengembangan IPv6 (*IP next generation*) sebagai penerus IPv4. IPv6 ditetapkan menjadi salah satu standar IETF melalui RFC 2460.

Perubahan ke IPv6 juga mendorong berkembangnya protokol-protokol baru pada OSI dan TCP/IP sebagai penunjang protokol *routing* IPv6 itu sendiri, seperti misalnya protokol baru *Internet Control Message Protocol* (ICMPv6), *Neighbor Discovery*, dan *Multicast Listener Discovery* (MLD) [1], [2]. *Header* IPv6 yang lebih sederhana sehingga secara mendasar hal ini juga mempengaruhi infrastruktur *network* keseluruhan.

IPv6 telah dirancang dengan skalabilitas yang tinggi agar dapat digunakan dalam jangka waktu yang lama untuk terus memungkinkan pertumbuhan internet. Namun, penerapan IPv6 masih berjalan lambat dan masih terbatas dalam jaringan internet tertentu. Hal ini terjadi karena perangkat dan infrastruktur yang secara luas digunakan dalam keseluruhan jaringan internet masih merupakan perangkat dan infrastruktur IPv4, dan sepertinya masih akan terus berlangsung sampai beberapa waktu ke depan. Akan tetapi cepat atau lambat pada akhirnya IPv6 akan menggantikan dominasi IPv4 sebagai *routing protocol*.

Sebelum perubahan infrastruktur sepenuhnya ke IPv6, maka diperlukan suatu solusi di mana IPv6 harus dapat bekerja berdampingan dengan IPv4; keduanya harus dapat saling berkomunikasi dengan *compability* yang sesuai. Apabila selama masa transisi hal tersebut tidak dapat terpenuhi, maka dapat

terjadi kekacauan pada jaringan internet. Inilah yang membuat transisi dari IPv4 ke IPv6 dilaksanakan secara bertahap. Sebagai salah satu solusi dari permasalahan ini adalah dengan menggunakan metode *Tunneling*.

## 2.2. MEKANISME IPV6

Pada dasarnya IPv6 dikembangkan karena ada hal-hal yang tidak diantisipasi saat IPv4 dijadikan standar protokol *routing* pada 1981 (RFC 791). IPv4 telah terbukti tangguh, mudah diimplementasikan dan dioperasikan, dan telah melewati tes skalabilitas dalam jaringan internet yang digunakan secara global sampai saat ini. Namun desain IPv4 tidak mengantisipasi perkembangan pesat jaringan internet, beberapa di antaranya :

- a. Ketersediaan jumlah alamat IPv4 yang suatu saat akan tidak mencukupi,
- b. Kebutuhan akan konfigurasi yang lebih sederhana sekaligus keamanan pada level protokol internet,
- c. Layanan hanya bersifat *best effort* sehingga dibutuhkan *QoS* yang lebih baik.

Sedangkan IPv6 didesain untuk memperbaiki masalah di atas, seperti :

- a. Format header lebih sederhana,
- b. *Space* alamat lebih besar (128-bit),
- c. Pengalamatan dan infrastruktur *routing* yang lebih efisien dan berbentuk hirarki,
- d. Adanya opsi status konfigurasi pengalamatan,
- e. keamanan dan *QoS* lebih baik,
- f. dan lain-lain [3].

### 2.2.1. Perbandingan IPv6 dan IPv4

Selain beberapa poin perbedaan yang telah ditulis sebelumnya di atas, secara lebih spesifik masih ada lagi beberapa perbedaan lainnya. Secara garis besar, perbedaan lain antara IPv4 dengan IPv6 adalah seperti yang terlihat pada Tabel 2.1.

Tabel 2.1. Perbedaan IPv4 dengan IPv6

IPv4	IPv6
Pengalamatan 32-bit	Pengalamatan 128-bit
Dukungan terhadap IPsec bersifat pilihan	Secara <i>default</i> mendukung IPsec
<i>Header</i> tidak mengidentifikasi QoS oleh <i>router</i>	Terdapat <i>Flow Label field</i> pada <i>header</i> , sehingga <i>router</i> dapat mengidentifikasi QoS
Fragmentasi <i>packet</i> dilakukan oleh <i>sending host</i> dan <i>router</i>	Hanya <i>sending host</i> yang melakukan fragmentasi <i>packet</i> . <i>router</i> mengirim ICMPv6 ke <i>sending host</i> bila fragmen <i>packet</i> terlalu besar dan membuangnya.
<i>Header</i> mengandung <i>checksum</i> dan <i>options</i>	<i>Header</i> tidak mengandung <i>checksum</i> dan <i>options</i> dipindahkan ke <i>header</i> ekstensi
<i>Address Resolution Protocol</i> (ARP) digunakan untuk menerjemahkan alamat MAC dari alamat IP	<i>Multicast Neighbor Solicitation messages</i> digunakan sebagai pengganti ARP
ICMP <i>Router Discovery</i> digunakan untuk mencari <i>default gateway</i> terbaik	Menggunakan ICMPv6 <i>Router Solicitation and Router Advertisement messages</i>
Mendukung sampai 576-byte ukuran <i>packet</i>	Mendukung sampai 1280-byte ukuran <i>packet</i>

Sumber: Microsoft Corporation, 2008

Walaupun terdapat beberapa perbedaan antara pengalaman IPv4 dengan IPv6, ada hal-hal yang bisa dianggap ekuivalen di antara keduanya, atau bisa dikatakan konsep pengalaman keduanya tetap memiliki beberapa persamaan. Hal-hal yang ekuivalen antara IPv4 dengan IPv6 dapat terlihat pada Tabel 2.2 di bawah ini :

Tabel 2.2. Hal-Hal yang Ekuivalen pada IPv6 dan IPv4

IPv4 Address	IPv6 Address
<i>Class: A, B, C, D, E</i>	Konsep <i>Class</i> dihilangkan
IPv4 <i>Multicast addresses</i> (224.0.0.0/4)	IPv6 <i>multicast addresses</i> (FF00::/8)
<i>Broadcast addresses</i>	Tidak menggunakan <i>Broadcast</i>
<i>Unspecified address</i> (0.0.0.0)	<i>Unspecified address</i> (::)
<i>Loopback address</i> (127.0.0.1)	<i>Loopback address</i> (::1)
<i>Public IP addresses</i>	<i>Global unicast addresses</i>
Alamat IP <i>Private</i> 10.0.0.0/8, 172.16.0.0/12, dan 192.168.0.0/16)	Alamat <i>Site-local</i> (FEC0::/10)
Alamat <i>Autoconfigured</i> (169.254.0.0/16)	Alamat <i>Link-local</i> (FE80::/64)

Sumber: Microsoft Corporation, 2008

### 2.2.2. Tipe Alamat Pada IPv6

Berbeda dengan IPv4 yang menggunakan alamat *Broadcast*, fitur ini dihilangkan pada IPv6 karena dianggap memberikan beban lebih kepada jaringan secara keseluruhan, terutama pada suatu domain jaringan yang memiliki *host* dalam jumlah besar. Fungsi *Broadcast* ini digantikan dengan *Multicast*. Terdapat tiga jenis alamat pada IPv6, yaitu :

a. *Unicast*

Suatu alamat *unicast* digunakan untuk mengidentifikasi suatu *interface* tunggal. Sebuah *packet* yang ditujukan ke suatu alamat *unicast* akan dikirimkan ke *interface* yang memiliki alamat *unicast* tersebut [2], [3], [4].

b. *Multicast*

Suatu alamat *multicast* digunakan untuk mengidentifikasi suatu kumpulan dari beberapa *interface*, sehingga bila ada *packet* yang ditujukan ke suatu alamat *multicast*, akan dikirimkan ke semua *interface* yang diidentifikasi alamat *multicast* tersebut. Alamat *multicast* digunakan untuk komunikasi dari satu-ke-sekumpulan *interface* [2], [3], [4].

c. *Anycast*

Suatu alamat *anycast* juga digunakan untuk mengidentifikasi suatu kumpulan dari beberapa *interface*, namun bila ada *packet* yang ditujukan ke suatu alamat *anycast* maka akan dikirimkan ke salah satu *interface* (dari sekumpulan tersebut) yang memiliki jarak terdekat (*routing distance* terkecil) [2], [3], [4].

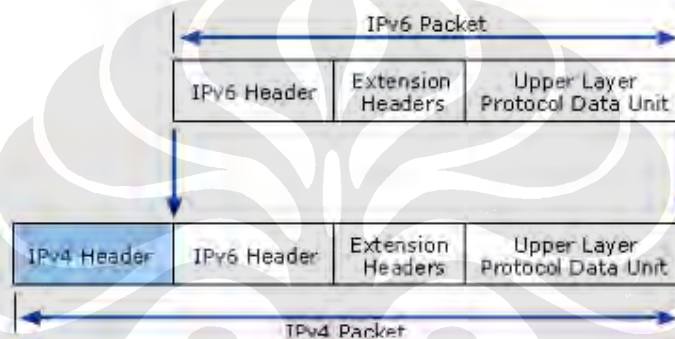
Berbeda dengan IPv4 di mana metode pengalamatan mengidentifikasikan langsung *node* yang terhubung ke jaringan, pengalamatan pada IPv6 mengidentifikasikan *interface*, bukan *node*. Sebuah *node* diidentifikasi oleh alamat *unicast* yang dimiliki *interface*-nya.

### 2.3. MEKANISME TUNNELING

*Tunneling* merupakan suatu metode transisi dari IPv4 ke IP generasi selanjutnya IPv6. Mekanisme transisi ini melibatkan baik IPv4 maupun IPv6. Contoh mekanisme *tunneling* paling sederhana adalah dua *node* IPv6 yang

terhubung melalui infrastruktur jaringan IPv4 atau sebaliknya. Salah satu alasan mengapa *tunneling* umum digunakan sebagai salah satu metode dalam transisi IPv4 ke IPv6 murni adalah karena metode ini tidak banyak menghabiskan biaya. Secara keseluruhan transisi ke IPv6 tidaklah murah dari segi teknologi dan infrastruktur jaringan disebabkan karena mayoritas infrakstruktur yang telah ada sebelumnya dirancang untuk IPv4 [5].

Secara sederhana prinsip dari *tunneling* adalah melewati *packet* IPv6 ke dalam jaringan IPv4 di mana *packet* IPv6 dijadikan sebagai *payload* dari *packet* IPv4. Gambar 2.1 di bawah ini merupakan ilustrasi sederhananya



Gambar 2.1. Enkapsulasi *Packet* IPv6 dalam IPv4  
(Sumber : Microsoft Technet, 2003)

Ada beberapa metode *tunneling* yang dapat dipilih untuk diaplikasikan terhadap suatu jaringan, yaitu: *tunneling* manual (*Configured tunneling*), *tunneling* otomatis (*automatic tunneling*). Tiap metode memiliki karakteristik dan macamnya masing-masing.

### 2.3.1. *Tunneling* Manual

*Tunneling* manual atau *Tunneling* terkonfigurasi merupakan cara konfigurasi paling sederhana dalam menerapkan IPv6 melalui jaringan IPv4. Tiap *endpoints* (*node-node* ujung dari *tunnel*) pada suatu *tunnel* manual memiliki pasangan alamat IPv4 dan IP6 dan dikonfigurasi secara manual. Syaratnya adalah pada *node endpoints* harus bersifat *dual-stack* (mampu bekerja pada IPv4 dan IPv6).



*Tunneling* manual mudah untuk diimplementasikan, tetapi secara *default* tidak menunjang fitur-fitur keamanan seperti *authentication* dan *monitoring*. Kekurangan terbesarnya adalah mengharuskan campur tangan administrator setiap ada perubahan topologi jaringan, misalnya bila ada *tunnel* baru yang harus diset atau diubah konfigurasinya.

### 2.3.2. *Tunneling* Otomatis

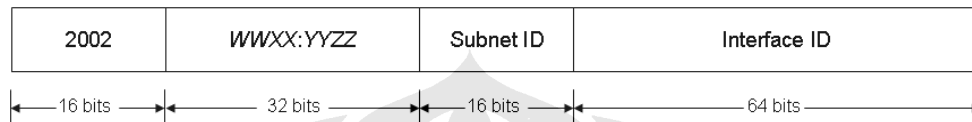
*Tunneling* otomatis menggunakan mekanisme metode pengalamatan di mana bisa disediakan sebuah alamat atau *prefix* IPv6 berdasarkan sebuah alamat IPv4, atau alamat IPv4 dari sebuah *node* bisa juga diintegrasikan ke alamat IPv6-nya. Koneksi *tunnel* terjadi saat dibutuhkan dan berakhir saat tidak dibutuhkan. Konfigurasi *endpoints* dari *tunnel* yang menentukan prioritas rute, *tunnel interfaces* secara logika, dan alamat tujuan IPv6. Contoh-contoh metode *tunneling* otomatis adalah 6to4, ISATAP, dan Teredo. Metode-metode tersebut dapat diimplementasikan dalam *Microsoft operating systems*. Setelah ini beberapa di antaranya akan dijelaskan lebih lanjut.

Penerapan metode *tunneling* otomatis memiliki kekuatan dan kelemahan. Salah satu kekuatan utamanya adalah mekanismenya yang dapat dikonfigurasi secara langsung serta otomatis dan dapat bertukar *packet* IPv6 secara langsung dari satu *host* ke *host* lain menggunakan jaringan IPv4. Namun kelemahannya adalah *host* yang terhubung pada jaringan dapat memiliki alamat IPv6 yang tidak tetap (dapat berubah), sehingga terkadang menyulitkan untuk dapat bertindak sebagai *server*. Contoh kelemahan lainnya adalah (misal pada 6to4) adanya protokol yang menggunakan *anycast* sehingga menjadi lebih sulit untuk melakukan pelacakan rute (*trace path*). *Packet* dapat melewati rute balik yang berbeda-beda dari rute kirimnya [6].

#### 2.3.2.1. *Tunneling* 6to4

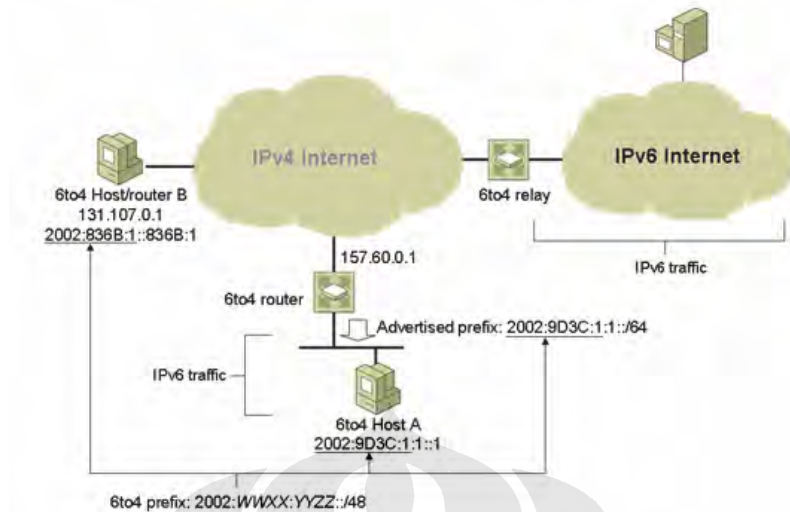
*Tunneling* 6to4 dijadikan suatu standar IETF melalui RFC 3056. Metode ini memungkinkan *Packet* IPv6 untuk ditransmisi melalui suatu jaringan IPv4 tanpa perlu melakukan konfigurasi *explicit tunnel*. 6to4 dapat menganggap keseluruhan jaringan IPv4 sebagai *single link* [7].

6to4 menggunakan format alamat global dengan prefiks 2002:wwwx:yyzz::/48. Di mana wwwx:yyzz merupakan bentuk heksadesimal dari alamat desimal IPv4 *host* (w.x.y.z). Alamat IPv4 yang terintegrasi dengan alamat IPv6 secara otomatis akan mengindikasikan bahwa metode transmisi (*traffic data*) yang akan digunakan adalah 6to4. Prefiksnya bisa juga disederhanakan dengan alamat global menjadi 2002::/16. Gambar 2.2 di bawah ini merupakan strukturnya :



Gambar 2.2. Format Alamat 6to4  
(Sumber : Microsoft Corp., 2008)

Mekanisme jaringan 6to4 ditunjang oleh komponen-komponen yang terdiri dari beberapa bagian utama, yaitu *router* 6to4 serta *relay* 6to4, dan tentunya *host*. Ada dua jenis *host*, yaitu *host* 6to4 dan *host/router* 6to4. Perbedaannya adalah *host* 6to4 merupakan *host* IPv6 yang tidak mendukung fitur *tunneling* IPv6 dalam jaringan IPv4, sebaliknya *host/router* 6to4 merupakan *host* IPv6/IPv4 yang secara default mampu mendukung fitur tersebut. *Host* 6to4 bergantung pada *router* 6to4 (*router* IPv6/IPv4) untuk mengatur terjadinya *tunneling*, dan *router* 6to4 juga bertugas untuk *men-forward traffic packet* yang ditujukan ke komponen lain pada jaringan 6to4. Sedangkan *relay* 6to4 sebenarnya merupakan *router* 6to4 yang digunakan sebagai semacam *border router* antara jaringan 6to4 dengan IPv6 murni. Router yang merupakan ujung-ujung dari *tunnel* 6to4 harus mampu mendukung protokol IPv4 dan IPv6 sekaligus. Gambar 2.3 di bawah ini merupakan yang mengilustrasikan penjelasan di atas langsung beserta contoh pengalamatannya :



Gambar 2.3. Mekanisme dan Komponen 6to4  
(Sumber : Microsoft Corp., 2008)

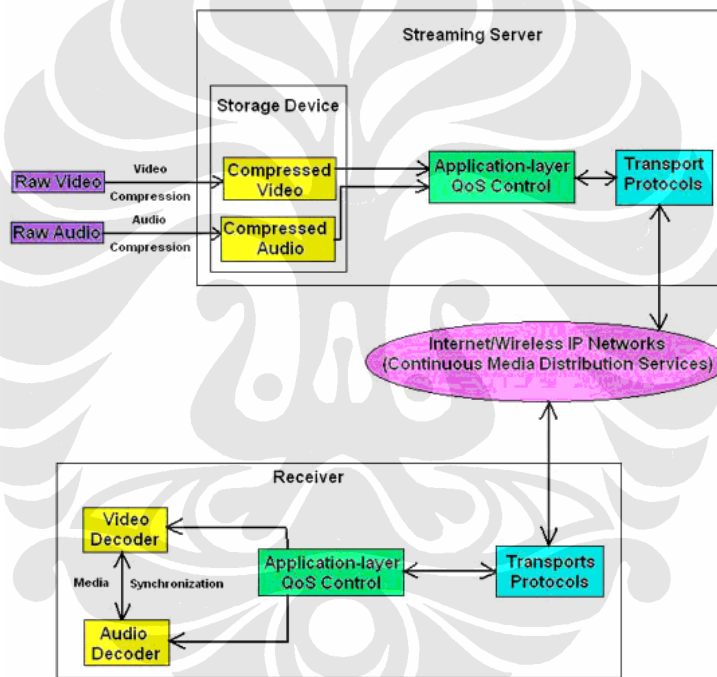
Pada contoh topologi di atas, *router* 6to4 dan *relay* 6to4 saling terhubung dengan default route karena masing-masing saling menganggap sebagai tetangga (*next-hop* dari *tunnel*). Keduanya melakukan *tunneling router-to-host* untuk mencapai *host/router* 6to4 (*host/router* B). Sedangkan *host* 6to4 (*host* A) menganggap *router* 6to4 sebagai *gateway* ke jaringan 6to4 [7].

#### 2.4. APLIKASI VIDEO STREAMING

Metode *Streaming* merupakan metode pengiriman *video* dan *audio* melalui internet dari *server* ke *client* untuk menjawab *request client* terhadap suatu *video* yang berada dalam suatu website. *Client* memainkan *file streaming* yang datang dalam kondisi *real-time* pada saat data diterima. Faktor yang berpengaruh terhadap *streaming* adalah *bandwith*. Faktor tersebut dapat menyebabkan proses *streaming* sering terganggu ketika *bandwith* yang ada tidak cukup sehingga mengakibatkan terjadinya *loss* dan *delay* dalam pengiriman.

Dalam proses *video streaming* ada beberapa hal yang harus diperhatikan yaitu kompresi, *continous media distribution services*, *quality of service (QoS)*, *streaming server*, mekanisme sinkronisasi, dan protokol untuk *media streaming*.

Gambar 2.4 menggambarkan bahwa data *raw video* dan *data raw audio* dikompresi terlebih dahulu dan disimpan di *storage device* dari *streaming server*. *Server* akan melakukan *streaming* ketika menerima *request* dari *client* (melalui *Internet*). *Streaming* dilakukan *server* dengan menggunakan modul *application-layer QoS*. *QoS control* lalu melakukan sinkronisasi *bit-stream data* ke dalam status jaringan dan persyaratan dari *QoS*. Tahap selanjutnya *packet* data tersebut akan dikirimkan oleh *transport protocol* ke dalam jaringan setelah mengalami proses sinkronisasi. Setiap *packet* yang sampai di *client* akan diproses terlebih dahulu oleh *transport layer* dan *application layer protocol* lalu akan di-*decode* oleh *decoder*.



Gambar 2.4. Arsitektur *Video Streaming*

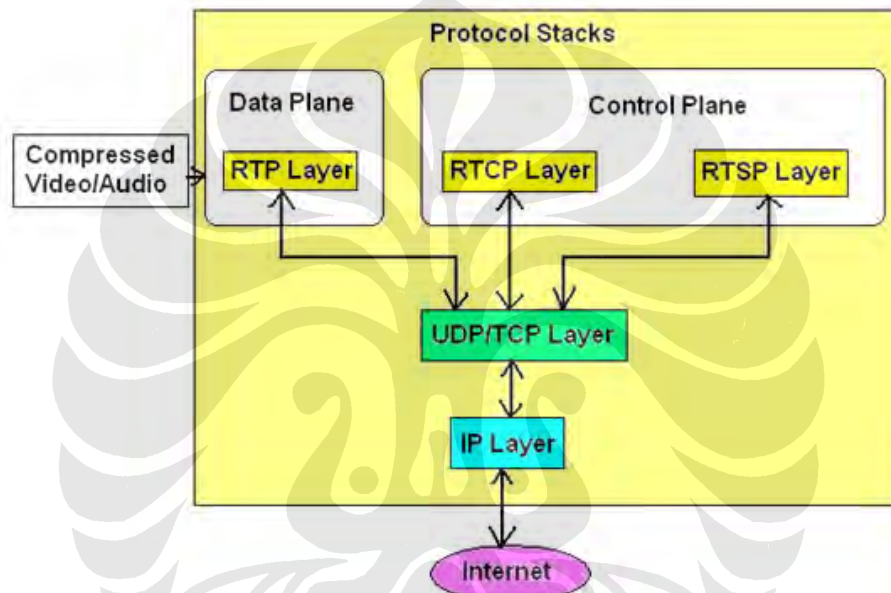
#### 2.4.1. Protokol Pada *Video Streaming*

Dua protokol yang mendukung berjalannya *video streaming* yaitu:

1. *Transport Protocol* yang menyediakan konektivitas secara *end-to-end* di jaringan untuk aplikasi *streaming*. *Transport protocol* terbagi menjadi *User Datagram Protocol* (UDP) dan *Transmission Control Protocol* (TCP).

2. *Session Control Protocol* yang mendefinisikan pesan dan prosedur untuk mengatur pengiriman data dari multimedia selama *session* terbentuk. Yang termasuk *Session control protocol* adalah *Real-Time Protocol* (RTP), *Real-Time Streaming Protocol* (RTSP), dan *Real-Time Control Protocol* (RTCP) [8].

Hubungan antara protokol-protokol di atas dapat dilihat pada Gambar 2.5 berikut ini :



Gambar 2.5. Konektivitas Protokol pada *Media Streaming*.

Pada *Layer Transport* protokol utama yang digunakan untuk bertukar data adalah TCP dan UDP. TCP menggunakan komunikasi dua arah di mana biasanya terdapat *acknowledgement* sebagai balasan indikasi bahwa suatu informasi telah sampai atau diterima, sehingga TCP lebih memberikan jaminan bahwa pengantaran *packet* akan lebih *reliable* jika dibandingkan dengan UDP yang tidak memiliki fitur *acknowledgement* tersebut dan lebih bersifat komunikasi satu arah. Salah satu penggunaan TCP adalah autentikasi *password* dan *user commands* seperti *pause* dan *fast-forward*. Kelemahan dari sifat TCP adalah memiliki respon yang kurang dalam kondisi jaringan yang sering berubah dan membuat *overhead* keseluruhan yang lebih besar. Namun pada beberapa kasus

tertentu seperti di mana jaringan menggunakan *firewall* yang memblok UDP, penggunaan TCP lebih menguntungkan. Sementara itu, UDP bersifat memiliki overhead keseluruhan lebih kecil sehingga *packet-packet* yang diantarkan bisa lebih cepat sampai. Karena data video dan audio mengkonsumsi *bandwidth* lebih besar maka *default* dari *media streaming* biasanya menggunakan UDP, terlebih jika *streaming* bersifat *live*.

Pada *Layer Application* protokol yang umum digunakan adalah RTSP, RTP ataupun RTCP. Beberapa *server streaming* juga ada yang menyediakan fitur protokol lain seperti HTTP dan MMS. *Real-Time Streaming Protocol* (RTSP) merupakan sebuah standar protokol dalam mendukung presentasi multimedia, terutama jika *broadcasting* bersifat global dan berskala besar. RTSP menggunakan TCP untuk *message kontrol* pada *player* dan UDP untuk pengiriman data video dan audio. RTP merupakan *Real-Time Transport Protocol* di mana biasanya bekerja berdampingan dengan protokol RTSP untuk mendukung fitur *real-time* pada multimedia. Sedangkan *Real-Time Control Protocol* (RTCP) lebih bersifat untuk monitoring dan kontrol terhadap RTP *sessions*.

Selain protokol-protokol di atas ada juga protokol lain yang digunakan bergantung pada *player* yang digunakan pada sisi *client*. Misal penggunaan protokol *RealNetworks Data Transport* (RDT) sebagai format *packet* saat *Helix Streaming Server* berkomunikasi secara RTSP dengan *Real Player*. *Microsoft Media Services* (MMS) yang digunakan untuk melayani presentasi multimedia dengan menggunakan *Windows Media Player* [9].

## 2.4.2. Parameter-Parameter Aplikasi Video Streaming

### 2.4.2.1. Bit rate

Dalam dunia multimedia, *bit rate* (kb/s) merepresentasikan jumlah informasi, atau detail yang disimpan per unit waktu dalam suatu rekaman *file*, terlebih pada video dan audio. *Bit rate* dapat tergantung pada frekuensi sampel, jenis *encoding*, dll. *Bit rate* juga dapat dikatakan sebagai *goodput* [10].

Cara manual menentukan suatu *bit rate* [11]:

$$V = \frac{S - (A \times L)}{L} \dots\dots\dots(2.1)$$

L = Durasi video dalam *second*

S = *Size* kB (700 MB x 1024° = 716 800 KB)

A = Audio *bitrate* in KB/s (224 kbit/s = 224 / 8° = 28 KB/s)

V = Video *bitrate* in KB/s, konversi ke kbit/s dikali dengan 8°

°8 *bit* = 1 *byte*.

°1024 = 1 *kilo*

#### 2.4.2.2. *Frame Rate*

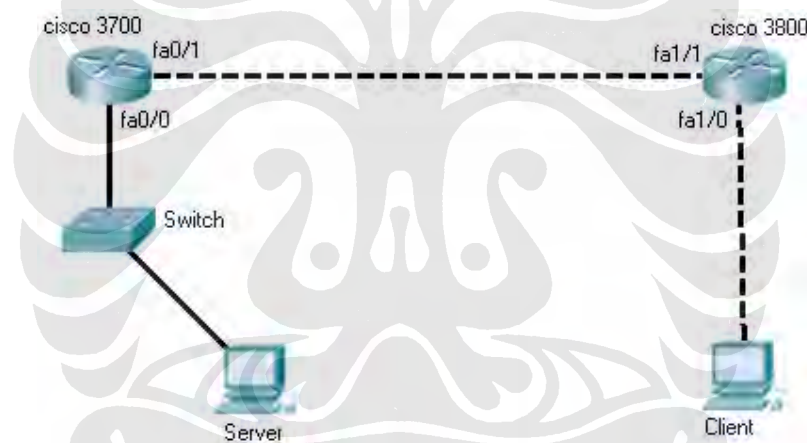
*Frame rate* merupakan banyaknya jumlah *frames* atau *images* yang bisa dihasilkan atau ditampilkan dalam satu satuan detik. *Frame rate* digunakan untuk mensinkronisasi audio dan gambar, terutama dalam video. *Frame rate* sangat menentukan *video playback rate*, yang bila semakin tinggi akan semakin menghasilkan video yang semakin halus [12].

## BAB III

### TOPOLOGI KONFIGURASI JARINGAN DAN PROSES *STREAMING*

#### 3.1 TOPOLOGI JARINGAN DAN SKENARIO PROSES *STREAMING*

Jaringan *test-bed* yang digunakan merupakan simulasi jaringan yang terdiri dari 2 buah komputer untuk jaringan IPv4 murni, jaringan IPv6 murni, dan jaringan 6to4 . Komputer yang digunakan menggunakan *platform Windows XP Service Pack 2* dan *Windows Server 2003*. Bentuk umum topologi jaringan *test-bed* yang digunakan untuk pengujian ditunjukkan oleh Gambar 3.1 di bawah ini :



Gambar 3.1. Topologi Umum Jaringan Test-bed Pengujian

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

1. *Server* merupakan laptop dengan sistem operasi *Windows Server 2003* dan memiliki spesifikasi: Intel Centrino *Mobile CPU 1.7GHz*, 2 *Gbytes RAM*, Ethernet 10/100 Mbps.
2. *Client* merupakan laptop dengan sistem operasi *Windows XP Service Pack 2* dan memiliki spesifikasi: Intel Pentium IV *Mobile CPU 2.26 GHz*, 512 *Mbytes RAM*, Ethernet 10/100 Mbps.



3. *Router1* merupakan *router* CISCO 3725 dengan spesifikasi: versi IOS 12.2(15)ZJ3, total RAM 262144 bytes, dan *Flash memory* 38862848 bytes.



Gambar 3.2. *Router* CISCO 3725 Series

4. *Router2* merupakan *router* CISCO 3845 dengan spesifikasi: versi IOS 12.3(11r)T2, total RAM 1048576 bytes, dan *Flash memory* 262144 bytes.



Gambar 3.3. *Router* CISCO 3845 Series

5. *Switch* CISCO Catalyst 3550.



Gambar 3.4. *Catalyst* CISCO 3550 Series

6. Kabel *Ethernet Cross* dan *Straight*.

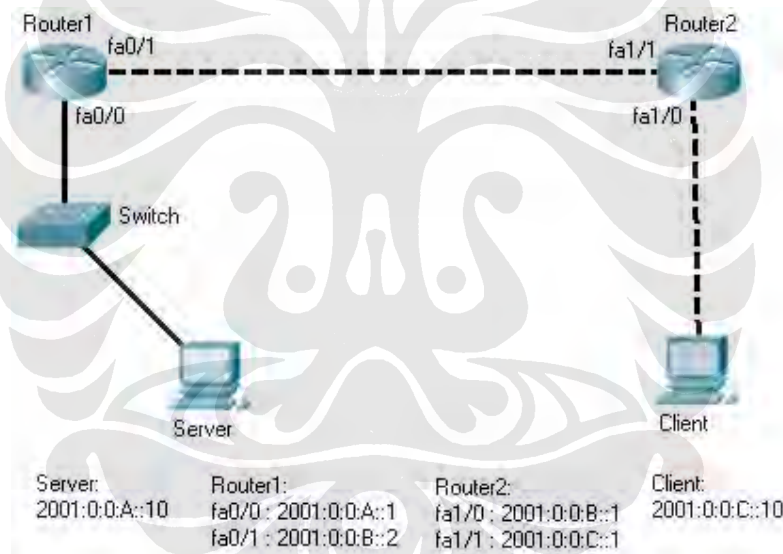
Skenario sederhana proses *streaming* yang akan dijalankan adalah *server* akan mengirimkan suatu file dengan metode *streaming* ke *client*. *Server* akan bertindak sebagai penampung file *streaming* tersebut dan kemudian akan

melakukan proses *streaming* ke *Client*. Semua perangkat keras tersebut menjalankan proses *streaming*, baik pengiriman dan penerimaan, dengan menggunakan Aplikasi VLC dan Helix *Streaming Server*.

## 3.2 KONFIGURASI JARINGAN

### 3.2.1. Konfigurasi Jaringan IPv6 Murni

Topologi dikonfigurasi menjadi IPv6 murni. Komponen jaringan *test-bed* IPv6 murni adalah 2 komputer dan 2 *router* yang pengalamatannya IPv6. Secara *default router* CISCO tidak menjalankan fitur IPv6, sehingga fitur IPv6 harus diaktifkan terlebih dahulu secara manual. Topologi serta konfigurasi alamat untuk jaringan IPv6 murni dapat dilihat pada Gambar 3.5 dan Lampiran 1.

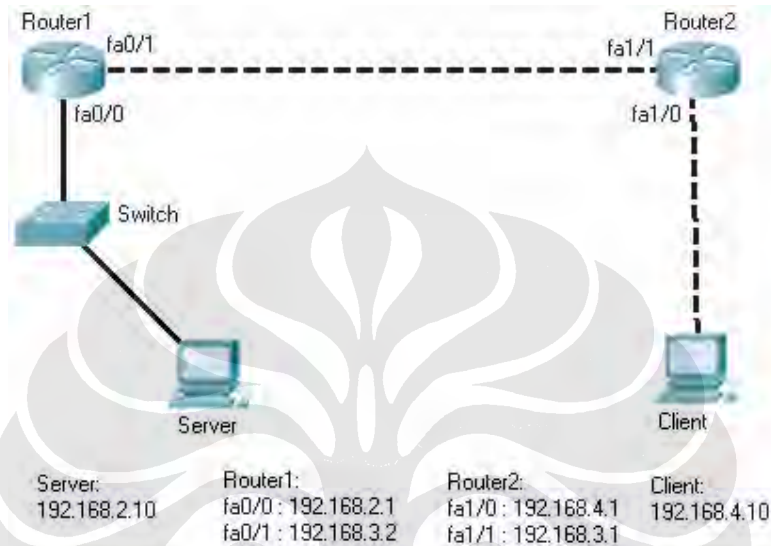


Gambar 3.5. Topologi Jaringan Test-bed IPv6

Topologi di atas merupakan topologi jaringan *test-bed* IPv6. Protokol *routing* yang digunakan adalah RIPng. Topologi jaringan IPv6 ini hanya bertujuan untuk mengecek apakah koneksi awal jaringan mampu bekerja dengan baik atau tidak pada jaringan IPv6, dan apakah *Streaming* dengan menggunakan aplikasi VLC dan Helix *Streaming Sever* dapat diimplementasikan atau tidak pada jaringan IPv6.

### 3.2.2. Konfigurasi Jaringan IPv4 Murni

Komponen jaringan *test-bed* IPv4 murni adalah 2 komputer dan 2 *router* yang pengalamatannya IPv4. Pemberian alamat IPv4 pada tiap *interface* dilakukan secara manual. Topologi serta konfigurasi alamat untuk jaringan IPv4 murni dapat dilihat pada Gambar 3.6 dan Lampiran 2.

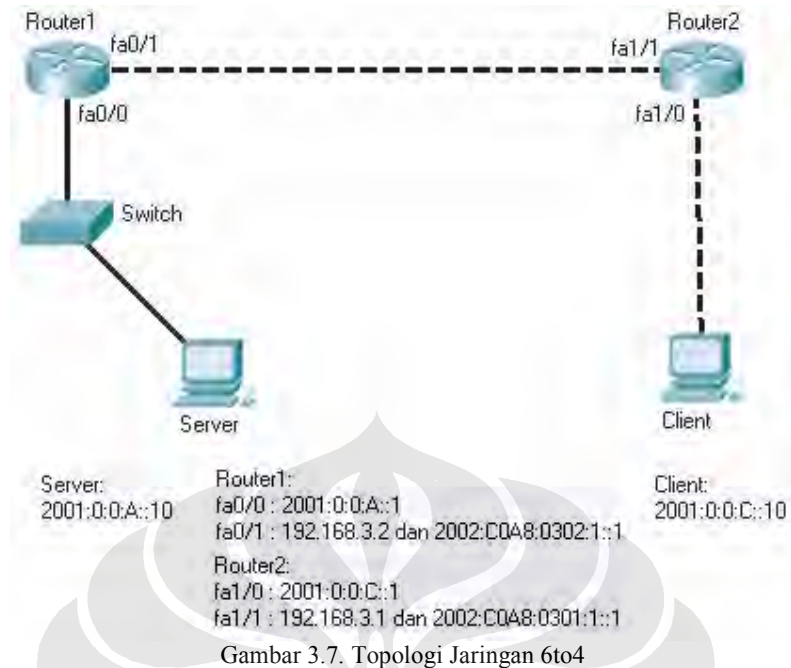


Gambar 3.6. Topologi Jaringan Test-bed IPv4

Topologi di atas merupakan topologi jaringan *test-bed* IPv4. Protokol *routing* yang digunakan adalah RIP versi 2, sehingga dapat memberikan fitur *classless routing*. Topologi jaringan IPv4 ini bertujuan untuk mengecek apakah koneksi awal jaringan mampu bekerja dengan baik atau tidak, dan apakah *streaming* dengan aplikasi VLC dan Helix *Streaming Server* yang digunakan mampu diimplementasikan atau tidak pada jaringan. Setelah ini jaringan IPv4 akan diberikan beberapa perubahan yang memungkinkan fitur *tunneling* dapat dilakukan.

### 3.2.3. Konfigurasi Tunneling 6to4

Secara garis besar jaringan *test-bed* yang digunakan tidak banyak diubah. Perubahan dilakukan pada *Client* dan *Server* yang semuanya dikonfigurasi menjadi jaringan IPv6 murni. Router1 dan Router2 yang dikonfigurasi menjadi *router 6to4*. Topologi serta konfigurasi alamat untuk jaringan 6to4 dapat dilihat pada Gambar 3.7 dan Lampiran 3.



Proses *tunneling* terjadi antar tiap *router* (sepanjang jaringan tengah). *Client* yang berada dalam jaringan IPv6 murni akan mengakses *Server* dan membangun komunikasi melalui jaringan IPv4 yang telah dikonfigurasi menggunakan alamat 6to4 namun tetap menggunakan protokol *routing* IPv4 RIP versi 2.

Pengalamatan IPv6 6to4 pada tiap *serial-interface-router* sesuai dengan bentuk heksadesimal dari alamat desimalnya. Tabel konversi dari alamat desimal ke heksadesimal dapat dilihat di bawah ini :

Tabel 3.1. Alamat 6to4 Pada *Interface Router*

Router	Interface	Alamat IPv4 (desimal)	Alamat IPv6 (heksadesimal)	Alamat 6to4
Router1	fa0/1	192.168.3.2	C0A8:0302	2002:C0A8:0302:1::1/48
Router2	fa1/1	192.168.3.1	C0A8:0301	2002:C0A8:0301:1::1/48

### 3.3 KELENGKAPAN KONFIGURASI JARINGAN

Untuk uji coba jaringan dengan menggunakan VLC sebagai *server streaming* dan *streaming client*, seluruh *host* pada konfigurasi jaringan menggunakan sistem operasi Windows XP SP 2. Sedangkan untuk uji coba jaringan dengan menggunakan Helix *Streaming Server* sebagai *server streaming*

dan VLC sebagai *client streaming* digunakan sistem operasi Windows XP SP 2 pada *client* dan Windows Server 2003 SP 2 pada *Server*.

Sedangkan untuk software, yang digunakan antara lain :

1. *Wireshark*, sebagai aplikasi untuk menangkap dan menganalisa trafik pada suatu interface.
2. VideoLAN *Client* (VLC), sebagai aplikasi yang digunakan untuk melakukan *streaming* file-file audio dan video dari berbagai jenis format. VLC dapat digunakan baik sebagai *server* maupun sebagai *client*. Dan hal ini berlaku bagi kedua IPv4 dan IPv6. VLC adalah aplikasi *freeware* sehingga bisa didapat dengan gratis.
3. *Helix Streaming Server*, sebagai aplikasi lain yang dapat digunakan untuk *server streaming*. Seperti halnya dengan VLC, *Helix Streaming Server* juga mampu mendukung berbagai format video dan audio. Aplikasi ini juga memerlukan *license* namun ada juga versi *freeware* dengan fitur terbatas.
4. *Helix Mobile Producer*, sebagai aplikasi penunjang *Helix Streaming Server* yang digunakan untuk melakukan konversi format *file* audio atau video ke format yang bisa didukung oleh *Helix Streaming Server* untuk melakukan proses *streaming*. Pada skripsi ini konversi *encoding* yang digunakan adalah *.rm* (format pada *Real Media*).
5. *Real Player*, sebagai aplikasi pada sisi *client* yang menerima proses *streaming* menggunakan *Helix Streaming Server*. Seperti halnya VLC, *Real Player* juga mampu mendukung berbagai format video dan audio.

### 3.4 METODE PENGAMBILAN DATA

Pengambilan data akan dilakukan dengan pengiriman file video *streaming* dengan menggunakan aplikasi VLC dan *Helix Streaming Server* yang dijalankan pada konfigurasi jaringan IPv4, jaringan IPv6 dan jaringan *tunneling* 6to4, Pengambilan data akan dilakukan bergantian pada tiap konfigurasi jaringan di mana proses *streaming* dengan konfigurasi VLC sebagai *server streaming* dan *client streaming* dan kemudian *Helix Streaming Server* sebagai *server streaming* serta menggunakan VLC dan *Real Player* sebagai *client streaming*.

Ada dua file yang dijadikan *file* untuk melakukan proses *streaming* VLC, yaitu :

1. TestSkripsi.mpg yang berdurasi 60 detik dan berukuran 2,114 KB
2. TestSkripsi.mp4 yang berdurasi 60 detik dan berukuran 4,216 KB

Kedua *file* tersebut dibuat untuk durasi yang sama dengan format yang berbeda (.mpeg dan .mp4) dimaksudkan untuk melihat bagaimana hubungan antara perbedaan format dengan pengaruhnya saat di-*encode* terhadap proses *streaming* pada tiap konfigurasi jaringan.

Pada proses dengan menggunakan Helix *Streaming Server* dan *Real Player* sebagai *client*, ada beberapa jenis *file* yang akan di-*streaming*, yaitu:

1. TestSkripsi80-10.mp4 , TestSkripsi80-20.mp4 , dan TestSkripsi80-30.mp4
2. TestSkripsi160-10.mp4 , TestSkripsi160-20.mp4, dan TestSkripsi160-30.mp4
3. TestSkripsi(mp4)80,10.rm , TestSkripsi(mp4)80,20.rm , dan TestSkripsi(mp4)80,30.rm

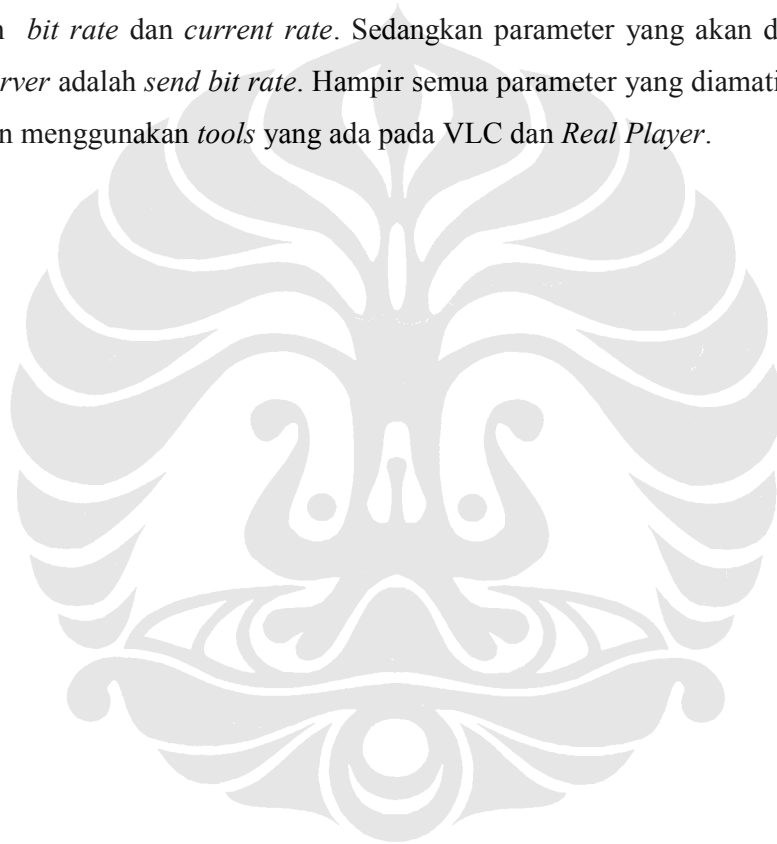
Sebenarnya hanya tiga jenis *file* yang akan di-*streaming*, namun file-file tersebut diberikan variasi pada *frame rate* dan *bit rate*. Sebagai contoh file dengan nama TestSkripsi80-10.mp4 mengandung arti bahwa *file* tersebut memiliki spesifikasi format .mp4 dengan *bit rate* 80 kb/s dan *frame rate* 10 fps, begitu juga yang lainnya. Format *file* .mp4 digunakan karena *size*-nya lebih besar dari pada format .mpg serta mampu didukung baik oleh VLC maupun *Real Player*.

Seluruh *file* tersebut berdurasi sama 60 detik. Ukuran *size* dari *file-file* berformat .mp4 dengan *bit rate* 80 kb/s adalah sekitar 1,1 Mb. Ukuran *size* dari *file-file* berformat .mp4 dengan *bit rate* 160 kb/s adalah sekitar 1,7 Mb. Ukuran *size* dari *file-file* berformat .rm dengan *bit rate* 80 kb/s adalah sekitar 1 Mb. File asli yang digunakan untuk konversi ke format .rm adalah *file* TestSkripsi.mp4. Yang perlu diperhatikan adalah hasil konversi ke *file* berformat .rm memiliki ukuran *size* yang jauh lebih kecil dari *file* aslinya TestSkripsi.mp4 yang berukuran 4,216 KB.

Hal-hal yang akan dibandingkan pada proses *streaming* dengan menggunakan Helix *Streaming Server* akan dijelaskan pada paragraf ini. Terdapat dua jenis variasi *bit rate* yang digunakan yaitu 80 dan 160 kb/s. Tujuannya adalah

untuk membandingkan bagaimana pengaruh besar *bit rate* terhadap hasil *streaming*. Variasi *frame rate* yang digunakan ada tiga, yaitu 10, 20, dan 30 fps, dan tujuannya adalah untuk membandingkan bagaimana pengaruh besar *frame rate* terhadap hasil *streaming*. Dan yang terakhir adalah akan dibandingkan bagaimana hasil *streaming* antara *file* asli berformat .mp4 dengan *file* hasil *encode* berformat .rm. Konversi ke *file* berformat .rm dilakukan dengan menggunakan *Helix Mobile Producer*.

Parameter yang akan diamati pada sisi *client* selama proses *streaming* adalah *bit rate* dan *current rate*. Sedangkan parameter yang akan diamati pada sisi *server* adalah *send bit rate*. Hampir semua parameter yang diamati didapatkan dengan menggunakan *tools* yang ada pada VLC dan *Real Player*.



# BAB IV

## ANALISIS TOPOLOGI JARINGAN

### DAN HASIL *STREAMING*

#### 4.1. PROSES PENGAMBILAN DATA

##### 4.1.1. Pengambilan Data VLC

Proses *streaming* video pada jaringan *test bed* dilakukan dengan menggunakan aplikasi VLC yang diuji coba melalui tiga macam konfigurasi, yaitu jaringan IPv4, Jaringan IPv6 dan Jaringan 6to4. Pemantauan data selama proses *streaming* dilakukan dengan menggunakan tambahan aplikasi *Wireshark*, selain dari penggunaan aplikasi VLC itu sendiri. Penggunaan fitur-fitur pada *Wireshark* dimaksudkan untuk melihat parameter-parameter yang bisa digunakan untuk pemantauan kinerja jaringan, sedangkan penggunaan fitur-fitur pada VLC dimaksudkan untuk mendapatkan perbandingan antara parameter *Send Bit rate (server)* dengan parameter *Stream Bit rate (client)*, serta parameter *Frame Rate (client)*.

Ada dua jenis *file* yang diuji coba pada tiap konfigurasi jaringan, yaitu:

1. TestSkripsi.mpg yang berdurasi 60 detik dan berukuran 2,114 KB
2. TestSkripsi.mp4 yang berdurasi 60 detik dan berukuran 4,216 KB

Kedua *file* tersebut dibuat untuk durasi yang sama dengan format yang berbeda (.mpeg dan .mp4) dimaksudkan untuk melihat bagaimana hubungan antara perbedaan format dan size *file* dengan pengaruhnya terhadap proses *streaming* pada tiap konfigurasi jaringan.

Semua *file* diuji coba *streaming* terhadap setiap konfigurasi jaringan dengan menggunakan VLC. Metode enkapsulasi data (encoding) yang digunakan untuk *streaming* pada konfigurasi VLC adalah MPEG TS. MPEG TS (Moving Picture Experts Group Transport *Stream*) merupakan suatu protokol komunikasi untuk audio, video dan data yang tujuan utamanya adalah untuk me-*multiplex*



variabel-variabel video dan audio agar outputnya sinkron. MPEG TS merupakan default *setting* dari VLC untuk enkapsulasi. Konfigurasi tambahannya berupa *setting*an codec video dan audio yang diatur dan divariasikan besarannya, dan inilah yang akan memberikan variasi yang akan digunakan untuk pemanggilan data. Pada *setting* video digunakan codec mp4v yang *Bitrate*-nya divariasikan besarannya mulai dari 1024, 512 dan 256 kb/s. Sedangkan pada *setting* audio digunakan codec mpga yang *bitrate*-nya juga divariasikan besarannya mulai dari 192, 96 dan 32 kb/s. Variasi tersebut di-set berpasangan mulai dari terkecil sampai terbesar. Ketentuan penulisan variasi *bit rate* dan jenis *file streaming* pada skripsi ini adalah sebagai berikut :

**Jenis\_file bit\_rate\_video,bit\_rate\_audio**

misalnya : mpeg 1204 , 192

Pengambilan parameter berupa *send bit rate* dan *stream bit rate* dilakukan selama proses *streaming* berlangsung. Khusus untuk *send bit rate* dan *stream bit rate* dicatat setiap lima detik sekali sehingga dihasilkan 11 data untuk setiap pengambilan datanya. Percobaan *streaming* dengan kedua *file* video dilakukan masing-masing 10 kali untuk tiap konfigurasi jaringan. Contoh pengambilan datanya bisa dilihat pada Lampiran 7 dan Lampiran 8.

#### **4.1.2. Pengambilan Data Helix Streaming Server Dan Real Player**

Proses *streaming* video pada jaringan *test bed* dilakukan dengan menggunakan aplikasi Helix *Streaming Server* yang diuji coba melalui tiga macam konfigurasi, yaitu jaringan IPv4, Jaringan IPv6 dan Jaringan 6to4. Pemantauan data selama proses *streaming* dilakukan dengan menggunakan tambahan aplikasi *Wireshark*. Penggunaan fitur-fitur pada *Wireshark* dimaksudkan untuk melihat parameter-parameter yang bisa digunakan untuk pemantauan kinerja jaringan, sedangkan penggunaan fitur-fitur pada *Real Player (client)* dimaksudkan untuk mendapatkan parameter *Bit rate* serta parameter *Frame Rate*.

Pada proses *streaming* menggunakan Helix *streaming server* dan *Real Player* sebagai *client* pendekatan saat pengambilan data yang dilakukan agak berbeda. Hal ini dikarenakan bahwa karakteristik Helix dan *Real Player* berbeda dari VLC. Setelah dilakukan beberapa percobaan, maka penulis berkesimpulan bahwa *variable bit rate* yang dikirim pada sisi *server* tidak dapat dipantau jika hanya dengan menggunakan *tools* dari Helix saja. Sehingga pada pengolahan data nantinya akan diasumsikan bahwa *bit rate* yang dikirim adalah sama dengan *bit rate* dari *file* aslinya yang di-*streaming*. Helix mampu melakukan *streaming file* tanpa enkapsulasi terlebih dahulu (*raw file*) sehingga selain variasi *bit rate*, variasi *frame rate* juga dapat dilakukan (sebelumnya pada VLC tidak tersedia fitur tersebut). Bahkan hal ini juga dapat dilakukan bila *file* asli ingin dienkapsulasi atau dikonversi ke format yang diinginkan menggunakan Helix *Mobile Produser*. *File-file* yang akan di-*streaming* akan diberi variasi *bit rate* dan *frame rate* yang berbeda-beda nantinya.

Ada dua jenis *file* yang diuji coba pada tiap konfigurasi jaringan, yaitu:

1. TestSkripsi80,10.mp4 , TestSkripsi80,20.mp4 , dan TestSkripsi80,30.mp4
2. TestSkripsi160,10.mp4 , TestSkripsi160,20.mp4, dan TestSkripsi160,30.mp4
3. TestSkripsi(mp4)80,10.rm , TestSkripsi(mp4)80,20.rm , dan TestSkripsi(mp4)80,30.rm

Ketentuan penulisan variasi *bit rate* dan *frame rate* jenis *file streaming* Helix pada skripsi ini adalah sebagai berikut :

**Nama *file bit\_rate* , *frame\_rate* . format *file***

misalnya : TestSkripsi160,10.mp4

*Streaming* menggunakan *file* berformat .mp4 dengan variasi *bit rate* 80 dan 160 kb/s dimaksudkan untuk melihat bagaimana kinerja Helix *Streaming Server* dalam melakukan *streaming* dengan format *file* asli .mp4 tanpa konversi, serta bagaimana pengaruh besar *bit rate* terhadap hasil *streaming* pada *client*. Sedangkan *Streaming* menggunakan *file* berformat .rm dimaksudkan untuk melihat apakah hasil *streaming* dengan menggunakan *encoding* ke format realmedia dapat memberi hasil yang memuaskan atau tidak jika dibandingkan

dengan hasil *streaming* tanpa konversi format. Perlu diperhatikan bahwa *default codec* dari *encoding* format realmedia (.rm) adalah RealAudio dan RealVideo10 dengan *default bit rate* adalah 80 kb/s.

Pengambilan parameter berupa *bit rate* dilakukan pada *Real Player (client)* selama proses *streaming* berlangsung. Khusus untuk *frame rate* dicatat setiap lima detik sekali sehingga dihasilkan 11 data untuk setiap pengambilan datanya karena pengiriman dengan menggunakan *Helix Streaming Server* ternyata lebih stabil pada parameter *bit rate*. Percobaan *streaming* tiap *file* video dilakukan masing-masing 5 kali untuk tiap konfigurasi jaringan. Contoh pengambilan datanya dapat dilihat pada lampiran 9.

## 4.2. DOKUMENTASI JARINGAN

### 4.2.1. Jaringan IPv6

Pada Jaringan IPv6 memiliki karakteristik tersendiri. Topologi IPv6 menggunakan protokol *routing* RIPng. Setelah konfigurasi terhadap keseluruhan komponen jaringan dilakukan, maka diambil beberapa dokumentasi untuk menganalisis cara kerja jaringan IPv6.

Hasil tes koneksi pada komputer *client* (2001:0:0:C::10) dan komputer *server* (2001:0:0:A::10) dapat dilihat pada Gambar 4.1 dan Gambar 4.2.

```
C:\Documents and Settings\tiar>tracert 2001:0:0:a::10
Tracing route to 2001:0:0:a::10 over a maximum of 30 hops
  0  <1 ms    <1 ms    <1 ms    2001:0:0:c::1
  1  <1 ms    <1 ms    <1 ms    2001:0:0:b::2
  2  <1 ms    <1 ms    <1 ms    2001:0:0:a::10
Trace complete.
```

Gambar 4.1. Client IPv6

```
D:\Documents and Settings\Administrator>tracert 2001:0:0:C::10
Tracing route to 2001:0:0:c::10 over a maximum of 30 hops
  0  <1 ms    <1 ms    <1 ms    2001:0:0:a::1
  1  <1 ms    <1 ms    <1 ms    2001:0:0:b::1
  2  <1 ms    <1 ms    <1 ms    2001:0:0:c::10
Trace complete.
```

Gambar 4.2. Server IPv6

Dari konfigurasi di atas terlihat bahwa dengan menggunakan *command tracert* dari *client* ke *server* dan dari *server* ke *client* berhasil dilakukan. Keduanya

terhubungkan oleh *Router1* (2001:0:0:B::2) dan *Router2* (2001:0:0:B::1). Untuk lebih jelas konfigurasi selengkapnya dapat dilihat pada Lampiran 1.

#### 4.2.2. Jaringan IPv4

Pada Jaringan IPv4 memiliki karakteristik tersendiri. Topologi IPv4 menggunakan protokol *routing* RIP versi 2. Setelah konfigurasi terhadap keseluruhan komponen jaringan dilakukan, maka diambil beberapa dokumentasi untuk menganalisis cara kerja jaringan IPv4.

Hasil tes koneksi pada komputer *client* (192.168.4.10) dan komputer *server* (192.168.2.10) dapat dilihat pada Gambar 4.3 dan Gambar 4.4.

```
C:\Documents and Settings\tiar>tracert 192.168.2.10
Tracing route to 192.168.2.10 over a maximum of 30 hops
  1  <1 ms    <1 ms    <1 ms    192.168.4.1
  2  <1 ms    <1 ms    <1 ms    192.168.3.2
  3  <1 ms    <1 ms    <1 ms    192.168.2.10
Trace complete.
```

Gambar 4.3. *Client* IPv4

```
D:\Documents and Settings\Administrator>tracert 192.168.4.10
Tracing route to PRESARIO-6F6832 [192.168.4.10]
over a maximum of 30 hops:
  1  <1 ms    <1 ms    <1 ms    192.168.2.1
  2  <1 ms    <1 ms    <1 ms    192.168.3.1
  3  <1 ms    <1 ms    <1 ms    PRESARIO-6F6832 [192.168.4.10]
Trace complete.
```

Gambar 4.4. *Server* IPv4

Dari konfigurasi di atas terlihat bahwa dengan menggunakan *command tracert* dari *client* ke *server* dan dari *server* ke *client* berhasil dilakukan. Keduanya terhubungkan oleh *Router1* (192.168.3.2) dan *Router2* (192.168.3.1). Untuk lebih jelas konfigurasi selengkapnya dapat dilihat pada Lampiran 2.

#### 4.2.3. Jaringan 6to4

Pada Jaringan 6to4 memiliki karakteristik tersendiri. Topologi 6to4 menggunakan protokol *routing* RIP versi 2 pada bagian IPv4-nya dan RIPng pada bagian IPv6-nya. Setelah konfigurasi terhadap keseluruhan komponen jaringan dilakukan, maka diambil beberapa dokumentasi untuk menganalisis cara kerja jaringan 6to4.

Hasil tes koneksi pada komputer *client* (2001:0:0:C::10) dan komputer *server* (2001:0:0:A::10) dapat dilihat pada Gambar 4.5 dan Gambar 4.6.

```
C:\Documents and Settings\tiar>tracert 2001:0:0:a::10
Tracing route to 2001:0:0:a::10 over a maximum of 30 hops
  1  <1 ms    <1 ms    <1 ms    2001:0:0:c::1
  2  1 ms     <1 ms    <1 ms    2002:c0a8:302:1::1
  3  1 ms     <1 ms    <1 ms    2001:0:0:a::10
Trace complete.
```

Gambar 4.5. *Client* 6to4

```
D:\Documents and Settings\Administrator>tracert 2001:0:0:C::10
Tracing route to 2001:0:0:c::10 over a maximum of 30 hops
  1  <1 ms    <1 ms    <1 ms    2001:0:0:a::1
  2  1 ms     <1 ms    <1 ms    2002:c0a8:301:1::1
  3  1 ms     <1 ms    <1 ms    2001:0:0:c::10
Trace complete.
```

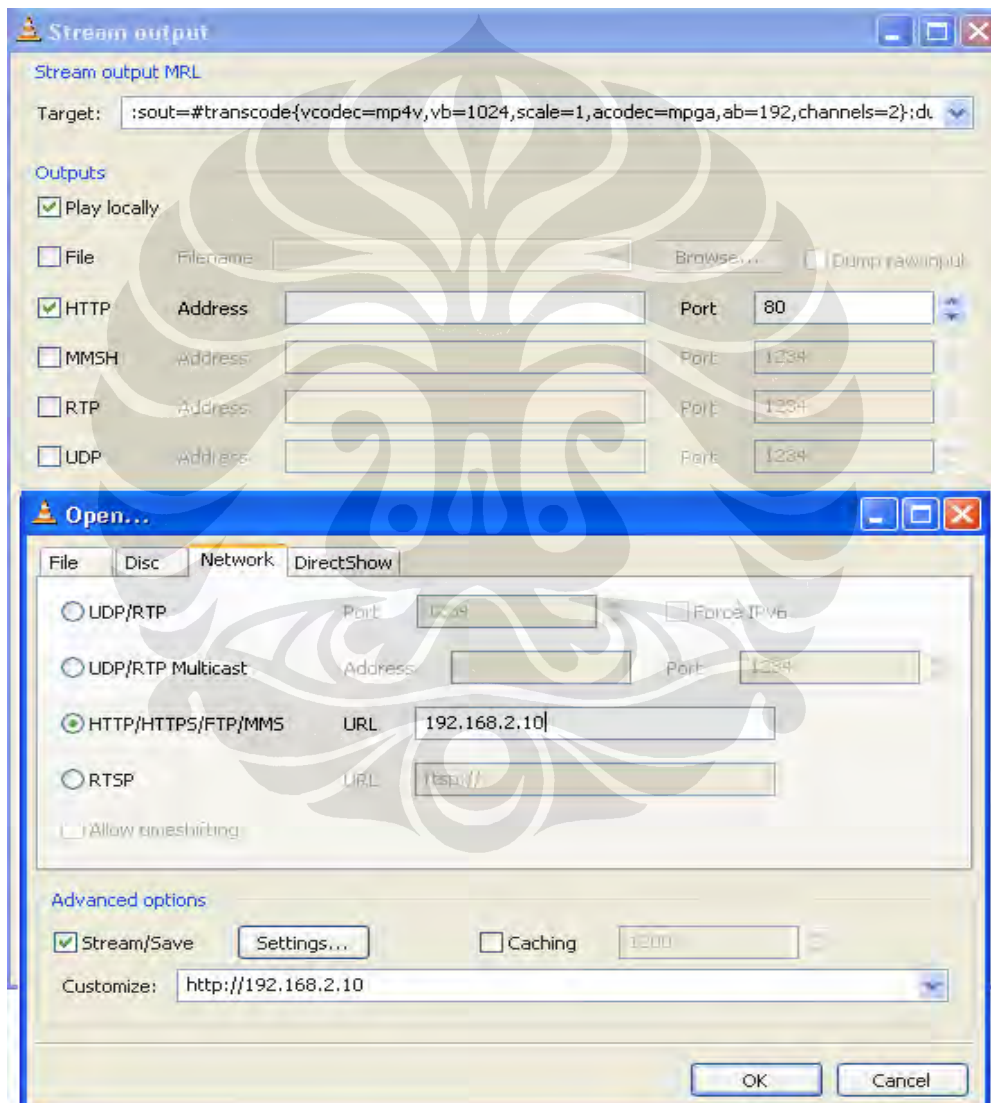
Gambar 4.6. *Server* 6to4

Dari konfigurasi di atas terlihat bahwa dengan menggunakan *command tracert* dari *client* ke *server* dan dari *server* ke *client* berhasil dilakukan. Keduanya terhubung oleh *Router1* (2002:c0:a8:302:1::1) dan *Router2* (2002:c0:a8:301:1::1). *Command tracert* memang tidak menampilkan alamat IPv4 dari kedua *router*, namun cara kerjanya adalah sebagai berikut: Alamat 6to4 dan IPv4 pada kedua *router* diberikan pada *interface fast ethernet* yang menyambungkan kedua *router*. Komputer *client* dan *server* yang terhubung dalam jaringan IPv6 secara otomatis hanya akan mencari alamat IPv6 pada *router*. Namun sebagai logikanya adalah kedua *router* tidak akan mungkin dapat tersambung dan saling bertukar data jika alamat dari kedua *router* tidak berada dalam satu *network* yang sama, sedangkan alamat dari kedua *interface fastethernet* pada kedua *router* adalah 2002:c0:a8:302:1::1/64 dan 2002:c0:a8:301:1::1/64. Porsi alamat *network*-nya adalah 64-bit pertama dari alamat IPv6, sehingga kedua *router* tersebut tidak berada dalam satu *network* IPv6. Artinya kedua *router* tersebut hanya dapat saling terhubung dengan melalui *interface fastethernet* yang beralamatkan IPv4 yang berada dalam satu *network* IPv4, yaitu 192.168.3.2 (*Router1*) dan 192.168.3.1 (*Router2*) yang memiliki subnet 255.255.255.0. Inilah proses *tunneling* 6to4 yang terjadi. Untuk lebih jelas konfigurasi selengkapnya dapat dilihat pada Lampiran 3.

### 4.3. DOKUMENTASI *SERVER* DAN *CLIENT*

#### 4.3.1. VLC *Server* dan *Client*

Pada percobaan topologi IPv4 proses *streaming* menggunakan protokol HTTP dengan port 80. Tujuan dari penggunaan HTTP adalah membuat agar sifat VLC *server* menjadi dapat diakses oleh siapa saja yang ingin melakukan *streaming* dari pihak *client*. Contoh *setting server* dan *client* pada VLC dapat dilihat pada Gambar 4.7.



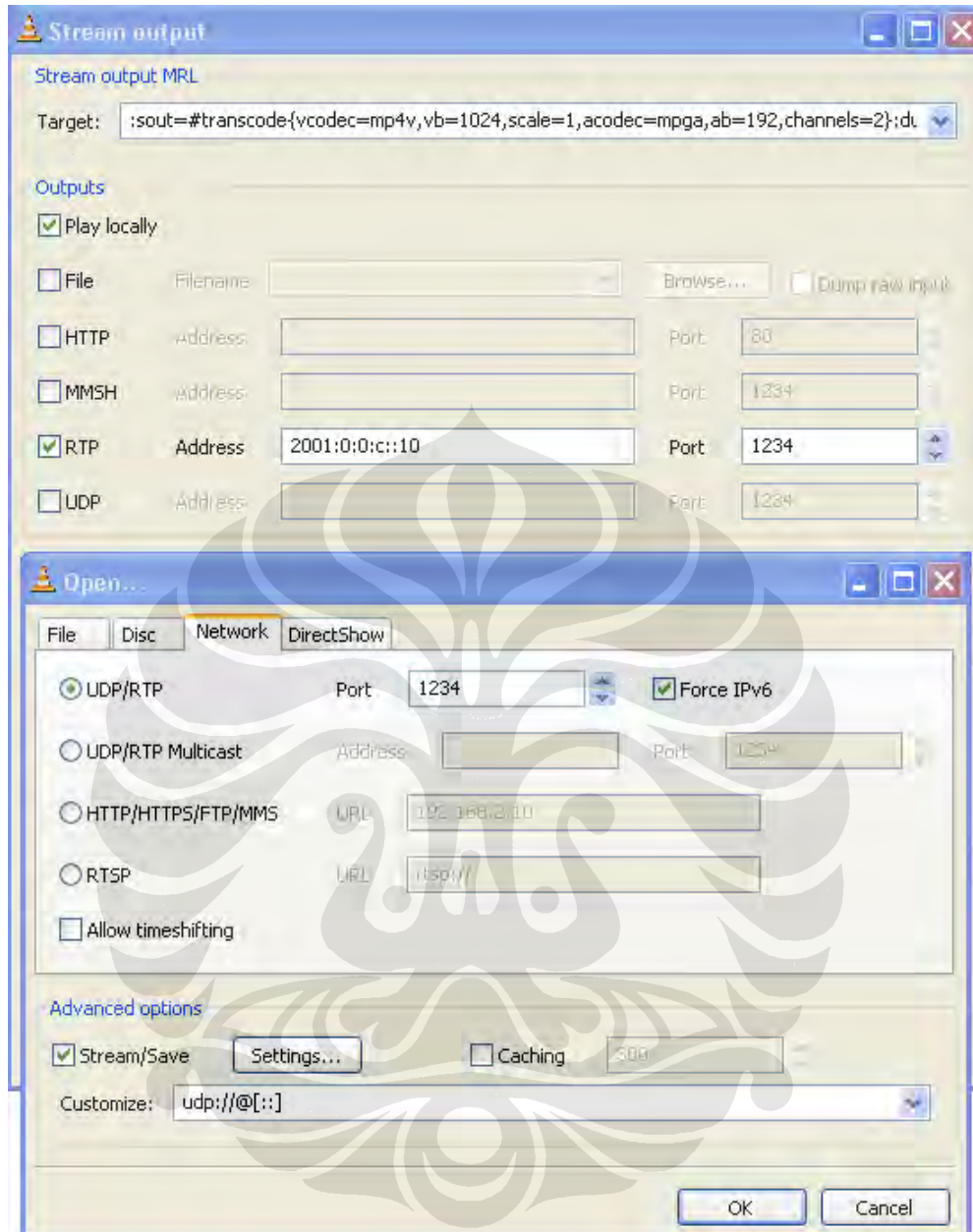
Gambar 4.7. *Setting VLC Server dan Client* pada Topologi IPv4

*Window Stream ouput* pada VLC merupakan *setting* dari *VLC server* dan *Window Open...* merupakan *setting* dari *VLC client*. Pada *VLC client* diatur untuk mengakses *server* pada *address* 192.168.2.10 melalui koneksi HTTP. Karena koneksi yang digunakan adalah HTTP maka protokol *streaming* pada *layer* bawah yang digunakan adalah TCP, seperti yang ditunjukkan oleh Gambar 4.8.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.10	192.168.4.10	HTTP	340	GET / HTTP/1.1
2	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
3	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
4	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
5	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
6	0.000000	192.168.2.10	192.168.4.10	HTTP	340	GET / HTTP/1.1
7	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
8	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
9	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
10	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
11	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
12	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
13	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
14	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192
15	0.000000	192.168.2.10	192.168.4.10	TCP	60	itto > mini-sql [594, 496] Seq=6140474100 Win=8192
16	0.000000	192.168.2.10	192.168.4.10	TCP	60	mini-sql >> itto [594, 496] Seq=6140474100 Win=8192

Gambar 4.8. Hasil Packet Sniffing Menggunakan Wireshark pada *VLC Client* Topologi IPv4  
Keterangan: Pada saat pengambilan data digunakan *address server* 192.168.2.20

Pada percobaan topologi IPv6 dan 6to4 proses *streaming* menggunakan protokol RTP dengan port 1234. HTTP tidak lagi digunakan karena fitur HTTP pada VLC ternyata belum mendukung untuk *address* IPv6. Karena protokol yang digunakan adalah RTP maka sifat *VLC server* berubah peran menjadi pengumpan *streaming* ke *client*. *Client* hanya menjadi tujuan dan penerima *streaming* yang dilakukan oleh *VLC server*. Contoh *setting server* dan *client* pada VLC dapat dilihat pada Gambar 4.9.



Gambar 4.9. Setting VLC Server dan Client pada Topologi IPv6 dan 6to4

*Window Stream output* pada VLC merupakan *setting* dari VLC server dan *Window Open...* merupakan *setting* dari VLC client. Pada VLC client diatur untuk mengakses server mengumpukan *streaming client* di address 2001:0:0:C::1 melalui koneksi RTP. Karena koneksi yang digunakan adalah RTP maka protokol *streaming* pada layer bawah yang digunakan adalah UDP, seperti yang ditunjukkan oleh Gambar 4.10 (IPv6) dan Gambar 4.11 (6to4).



No.	Time	Source	Destination	Protocol	Length
1	0.000000	192.168.1.101	192.168.1.102	HTTP	48
2	0.040000	109.204.210.133	192.168.1.102	HTTP	48
3	0.080000	109.204.210.133	192.168.1.102	HTTP	48
4	0.120000	109.204.210.133	192.168.1.102	HTTP	48
5	0.160000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
6	0.200000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
7	0.240000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
8	0.280000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
9	0.320000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
10	0.360000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
11	0.400000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
12	0.440000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
13	0.480000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
14	0.520000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
15	0.560000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
16	0.600000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
17	0.640000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
18	0.680000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
19	0.720000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
20	0.760000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
21	0.800000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100

Gambar 4.10. Hasil Packet Sniffing Menggunakan Wireshark pada VLC Client Topologi IPv6

No.	Time	Source	Destination	Protocol	Length
1	0.000000	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
2	0.000008	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
3	0.000016	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
4	0.000024	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
5	0.000032	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
6	0.000040	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
7	0.000048	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
8	0.000056	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
9	0.000064	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
10	0.000072	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
11	0.000080	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
12	0.000088	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
13	0.000096	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
14	0.000104	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100
15	0.000112	2001:0:0:0:0:0:0:0	2001:0:0:0:0:0:0:0	UDP	100

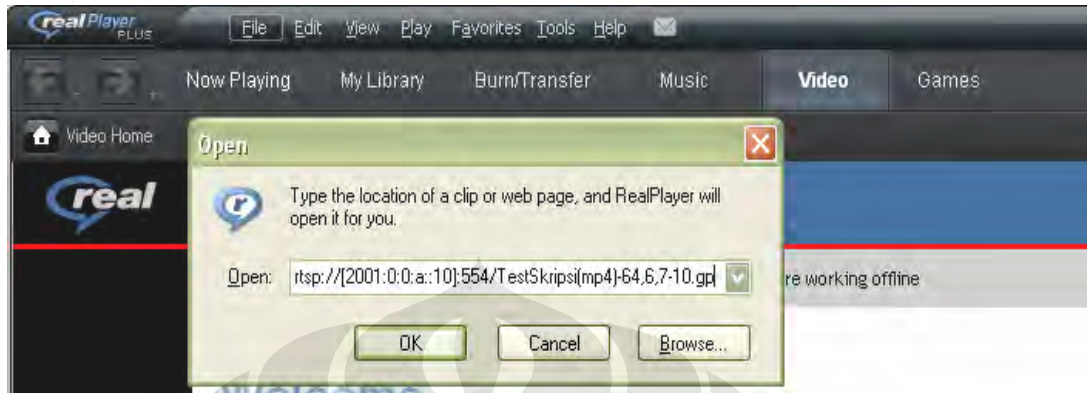
Gambar 4.11. Hasil Packet Sniffing Menggunakan Wireshark pada VLC Client Topologi 6to4

Keterangan: Pada saat pengambilan data digunakan *address server* 2001:0:0:A::20.

### 4.3.2. Helix Streaming Server dan Real Player

Proses *streaming* menggunakan Helix *streaming server* dan Real Player merupakan alternatif lain untuk melakukan *streaming*. Peran Helix sebagai *server* adalah sebagai mediator untuk melakukan *streaming* kepada *client* yang ingin mengakses dan *request streaming file-file* yang diinginkan. Sehingga *streaming*

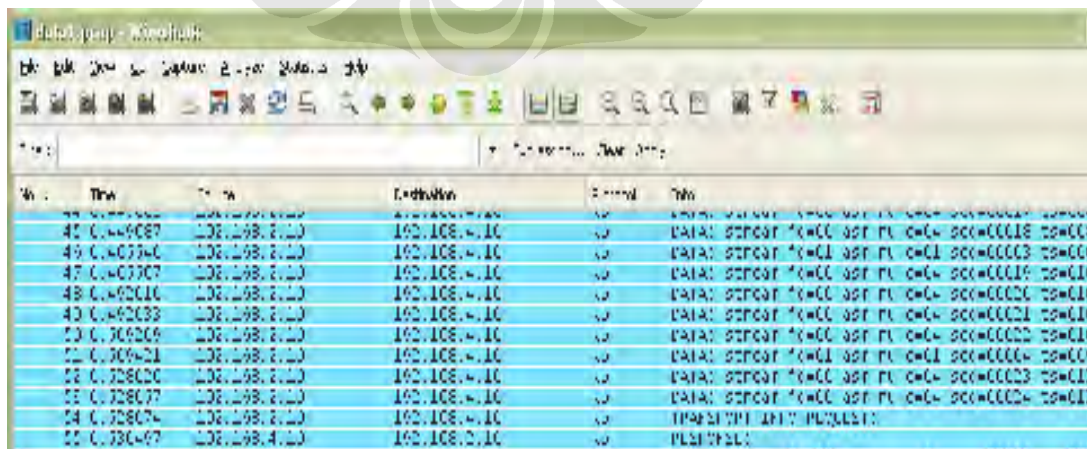
dapat dilakukan kapan pun saat dibutuhkan. *Real Player* pada *client* menggunakan protokol RTSP untuk mengakses *server* seperti yang ditunjukkan oleh Gambar 4.12.



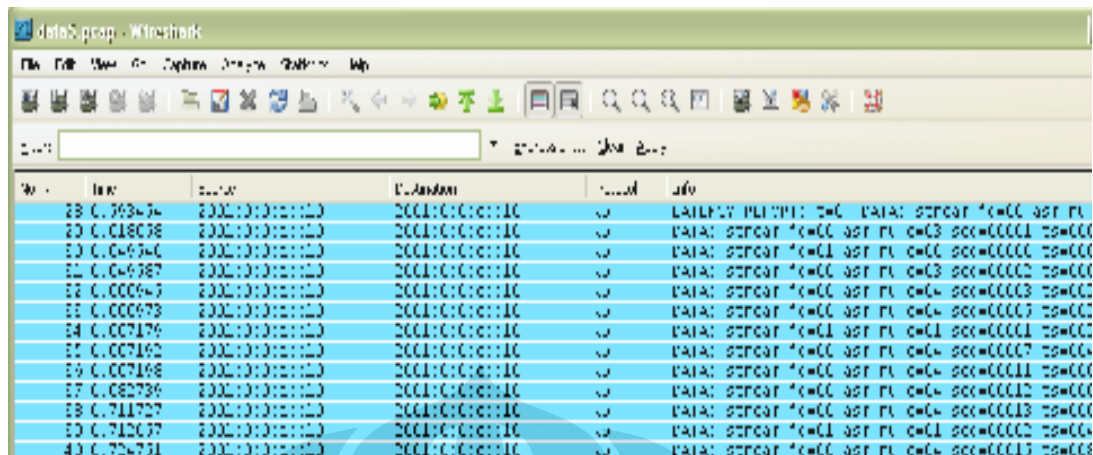
Gambar 4.12. Setting Real Player Untuk Melakukan Streaming

Pada Gambar 4.12 di atas adalah contoh *setting* pada *Real Player* yang akan mengakses *Helix streaming server* yang memiliki *address* 2001:0:0:A::10 untuk melakukan *streaming*. Protokol yang digunakan adalah RTSP dengan *port* 554. *Setting-an port* dapat diubah-ubah pada *Helix server* dan *Real Player* tinggal mencocokkan saja.

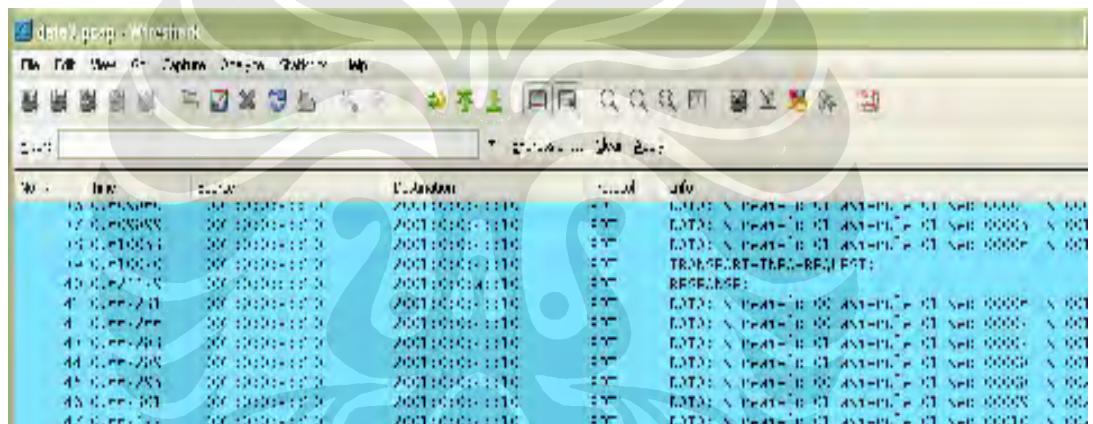
Tiap sesi *streaming* yang dibangun oleh *Real Player* menggunakan protokol RDT. Protokol RDT digunakan bila sesi media player yang digunakan adalah *Real Player*, apa pun jenis format *file* yang akan di-*streaming*. Hal ini berlaku baik pada topologi IPv4, topologi IPv6 dan topologi 6to4 seperti yang ditunjukkan oleh Gambar 4.13, Gambar 4.14 dan Gambar 4.15.



Gambar 4.13. Hasil Packet Sniffing Real Player Pada topologi IPv4



Gambar 4.14. Hasil Packet Sniffing Real Player Pada topologi IPv6



Gambar 4.15. Hasil Packet Sniffing Real Player Pada topologi 6to4

#### 4.4. ANALISIS BIT RATE VLC PADA TIAP TOPOLOGI JARINGAN

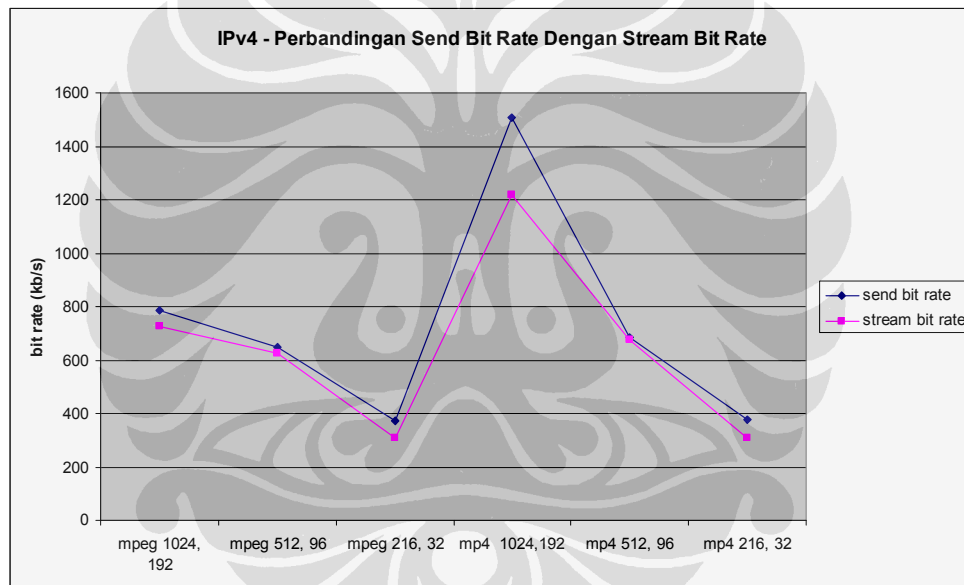
Pengambilan parameter *send bit rate* dan *stream bit rate* dilakukan menggunakan kedua *file* dengan variasi yang telah dijelaskan sebelumnya. Parameter *send bit rate* diukur pada sisi *server* sedangkan parameter *stream bit rate* diukur pada sisi *client*. Hal ini dapat dilakukan dengan menggunakan tools *Stream and Media Info* pada VLC. *Send bit rate* merupakan jumlah bit per satuan waktu yang menjadi output VLC pada sisi *server*. *Stream bit rate* merupakan jumlah bit per satuan waktu yang diterima dan menjadi input VLC pada sisi *client* selama proses *streaming* berlangsung. Satuan kedua parameter tersebut adalah kb/s.

Pada topologi jaringan IPv4, proses terjadinya *streaming* dilakukan dengan menggunakan protokol HTTP dengan port 80 sehingga menggunakan TCP pada transmisinya. Hasil pengolahan data pada jaringan IPv4 dapat dilihat pada Tabel 4.1.

Tabel 4.1. Tabel VLC Perbandingan Send *Bit rate* dengan *Stream Rate* pada IPv4

	<b>Mpeg 1024,192</b>	<b>mpeg 512, 96</b>	<b>mpeg 216, 32</b>	<b>mp4 1024,192</b>	<b>mp4 512, 96</b>	<b>mp4 216, 32</b>
<b>send bit rate (kb/s)</b>	788	646.873	370.827	1506.936	685.891	378.336
<b>stream bit rate (kb/s)</b>	725.555	627.091	307.836	1216.782	676.455	306.918

Tabel 4.1 di atas merupakan hasil rata-rata dari data-data yang telah diambil. Hubungan antara *send bit rate* dan *stream bit rate* untuk masing-masing *file* pada topologi jaringan IPv4 ditunjukkan lebih jelas pada Gambar 4.16.



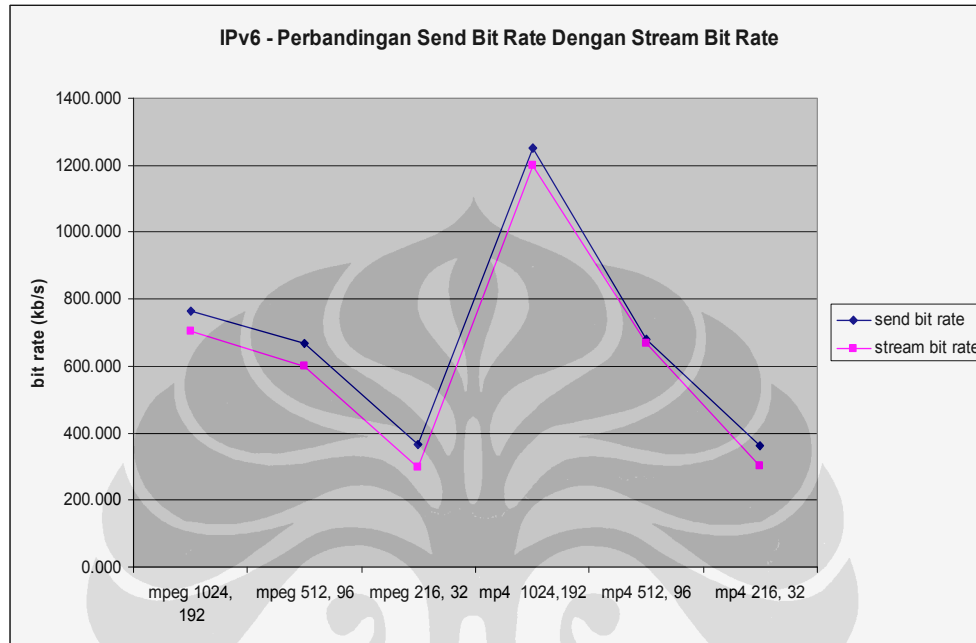
Gambar 4.16. Grafik VLC Perbandingan Send *Bit rate* dengan *Stream Rate* pada IPv4

Pada topologi jaringan IPv6 proses *streaming* dilakukan dengan menggunakan RTP (*Real Time Protocol*) karena ternyata fitur HTTP pada VLC belum mampu mendukung *streaming* pada address IPv6. Hasil pengolahan data pada topologi jaringan IPv6 dapat dilihat pada Tabel 4.2.

Tabel 4.2. Tabel VLC Perbandingan Send *Bit rate* dengan *Stream Rate* pada IPv6

	<b>mpeg 1024,192</b>	<b>mpeg 512, 96</b>	<b>mpeg 216, 32</b>	<b>mp4 1024,192</b>	<b>mp4 512, 96</b>	<b>mp4 216, 32</b>
<b>send bit rate (kb/s)</b>	764.255	666.973	367.827	1251.545	680.318	363.509
<b>stream bit rate (kb/s)</b>	705.118	600.345	297.018	1200.755	669.355	300.955

Tabel 4.2 di atas merupakan hasil rata-rata dari data-data yang telah diambil. Hubungan antara *send bit rate* dan *stream bit rate* untuk masing-masing *file* tersebut ditunjukkan lebih jelas pada Gambar 4.17.



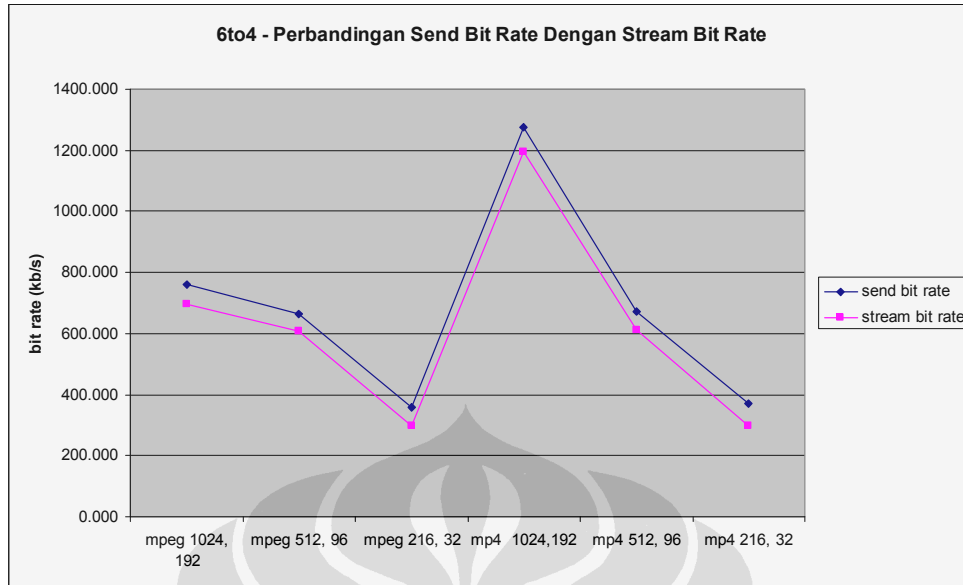
Gambar 4.17. Grafik VLC Perbandingan *Send Bit rate* dengan *Stream Rate* pada Ipv6

Pada topologi jaringan 6to4 proses *streaming* dilakukan tetap dengan menggunakan RTP (*Real Time Protocol*) karena sever dan *client* diletakkan pada subnet IPv6. Nantinya akan dibandingkan kinerja tunneling dengan metode 6to4 dengan topologi jaringan IPv4 dan IPv6. Hasil pengolahan data pada topologi jaringan 6to4 dapat dilihat pada Tabel 4.3.

Tabel 4.3. Tabel VLC Perbandingan *Send Bit rate* dengan *Stream Rate* pada IPv6

	mpeg 1024,192	mpeg 512, 96	mpeg 216, 32	mp4 1024,192	mp4 512, 96	mp4 216, 32
<b>send bit rate (kb/s)</b>	758.409	663.327	358.545	1276.382	672.773	370.645
<b>stream bit rate (kb/s)</b>	696.600	605.755	297.918	1194.955	613.482	297.727

Tabel 4.3 di atas merupakan hasil rata-rata dari data-data yang telah diambil. Hubungan antara *send bit rate* dan *stream bit rate* untuk masing-masing *file* tersebut ditunjukkan lebih jelas pada Gambar 4.18.



Gambar 4.18. Grafik VLC Perbandingan Send *Bit rate* dengan *Stream Rate* pada 6to4

Dengan melihat perbandingan data yang telah diperoleh dapat diketahui bahwa telah terjadi penurunan *bit rate* pada sisi *client*. Hal ini diindikasikan dengan garis merah yang merepresentasikan *stream bit rate* pada sisi *client* berada di bawah baris biru yang merepresentasikan *send bit rate* pada sisi *server*. Hal ini menunjukkan bahwa terjadi reduksi *bit rate* untuk dalam setiap *streaming file* baik itu melewati topologi IPv4, IPv6 atau pun 6to4. Untuk lebih memudahkan analisis dibuat Tabel 4.4.

Tabel 4.4. Tabel VLC Penurunan *Bit rate* (kb/s) dan Reduksi *Bit rate* (%)

		mpeg 1024, 92	mpeg 512, 96	mpeg 216, 32	mp4 1024,192	mp4 512, 96	mp4 216, 32
IPv4	reduksi <i>bit rate</i> (%)	7.92%	3.06%	16.99%	19.25%	1.38%	18.88%
	penurunan <i>bit rate</i> (kb/s)	62.445	19.782	62.991	290.155	9.436	71.418
IPv6	reduksi <i>bit rate</i> (%)	7.74%	9.99%	19.25%	4.06%	1.61%	17.21%
	penurunan <i>bit rate</i> (kb/s)	59.136	66.627	70.809	50.791	10.964	62.555
6to4	reduksi <i>bit rate</i> (%)	8.15%	8.68%	16.91%	6.38%	8.81%	19.67%
	penurunan <i>bit rate</i> (kb/s)	61.809	57.573	60.627	81.427	59.291	72.918

Dari Tabel 4.4 terlihat seberapa besar penurunan *bit rate* yang terjadi pada proses *streaming* di sisi *client*. Penurunan *bit rate* merupakan selisih antara parameter *send bit rate* dengan *stream bit rate*. Penerapan *streaming*

menggunakan VLC ini menunjukkan bahwa topologi jaringan IPv6 memiliki kinerja terbaik dengan rata-rata reduksi *bit rate* = 9.98 % dengan  $\sigma = 0.5$  %, diikuti dengan topologi jaringan IPv4 dengan rata-rata reduksi *bit rate* = 11.25 % dengan  $\sigma = 0.54$  %. Penerapan tunneling 6to4 juga tidak terlalu buruk kinerjanya dengan rata-rata reduksi *bit rate* = 11.43 % dengan  $\sigma = 0.36$  %.

Yang perlu diperhatikan di sini adalah dengan mengecilnya *bit rate* maka hasil *streaming* akan semakin menurun kualitasnya, baik itu video maupun audionya. *Bit rate* video yang terlalu rendah akan menghasilkan kualitas gambar yang buram dan kotak-kotak, sedangkan *bit rate* audio yang terlalu rendah akan menghasilkan kualitas suara yang bising dan tidak jernih. Dari hasil data yang diperoleh, maka *setting* terbaik untuk melakukan *streaming* dengan enkapsulasi MPEG TS mengacu pada parameter *bit rate* adalah sebagai berikut:

- ◆ *file .mp4* → *setting bit rate*: video mp4v (512 kb/s) dan audio mpga (96kb/s) dengan rata-rata reduksi bit terkecil sebesar 3.93 %.
- ◆ *file .mpg* → *setting bit rate*: video mp4v (512 kb/s) dan audio mpga (96kb/s) dengan rata-rata reduksi bit terkecil sebesar 7.24 %.

Artinya bisa disimpulkan bahwa pengaturan *bit rate* pada sisi *server* dengan *setting* 512 , 96 sudah cukup bagus untuk melakukan *streaming* pada semua topologi jaringan, mengingat buruknya kualitas *bit rate streaming* dengan *setting* 216 , 32 (reduksi *bit rate* terbesar). Namun bila ingin meningkatkan kualitas *streaming* dengan menaikkan *setting bit rate*, maka itu juga tidak akan menjadi masalah. Sebagai contoh bila kita lihat hasil *bit rate* dengan *setting* 1024 , 96 (selain dari pada ukuran *file*-nya juga yang relatif lebih besar, karena semakin besar *file* akan mempengaruhi hasil encoding pada output *streaming*) untuk format .mp4 maka menghasilkan hasil *bit rate* yang sudah cukup tinggi untuk semakin menghasilkan kualitas *streaming* yang bagus untuk diterapkan pada keseluruhan konfigurasi topologi jaringan.

#### 4.5. ANALISIS *FRAME RATE* VLC PADA TIAP TOPOLOGI JARINGAN

*Frame Rate* pada dasarnya menentukan apakah suatu *file* video akan memiliki kualitas baik atau tidak. Bila *frame rate* semakin meningkat, kualitas kehalusan gambar atau *frame* video akan semakin baik pula. Kedua *file* yang di-*streaming* memiliki *frame rate* 29.27 fps akan di-*stream* oleh *server* VLC dan akan dibandingkan dengan *frame rate* hasil tangkapan *streaming* pada VLC *client*. Metode enkapsulasi beserta variasi-nya yang digunakan sama dengan sebelumnya.

Satuan *frame rate* adalah *frame* per seconds, artinya *frame rate* dapat dicari dengan membagi total *displayed frames* pada tampilan *tools Stream and Media Info* pada VLC dengan durasi *file*, dalam hal ini artinya 60 detik. Pencatatan data dilakukan sesaat setelah tiap proses *streaming* berhenti dan kedua *file* di-*streaming* sebanyak 10 kali masing-masing pada topologi jaringan IPv4, jaringan IPv6 dan *tunneling* 6to4.

Hasil pengolahan data untuk setiap konfigurasi topologi jaringan yang digunakan adalah seperti yang terlihat pada Tabel 4.5.

Tabel 4.5. Tabel VLC Perbandingan Rata-Rata *Frame Rate* Seluruh Topologi

	<i>Frame Rate</i> (fps)		
	ipv4	ipv6	6to4
<b>mpeg 1024, 192</b>	30.41	30.18	30.74
<b>mpeg 512, 96</b>	29.66	30.70	30.44
<b>mpeg 216, 32</b>	30.28	30.69	30.07
<b>mp4 1024,192</b>	30.10	30.43	30.74
<b>mp4 512, 96</b>	29.66	30.53	30.43
<b>mp4 216, 32</b>	31.13	30.38	30.95
<b>rata-rata <i>frame rate</i> (fps)</b>	30.21	30.48	30.56

Hasil pengolahan data di atas menunjukkan bahwa *frame rate* antara ketiga topologi tidak terlalu banyak perubahan. Hal ini menunjukkan bahwa proses *streaming* dengan menggunakan metode enkapsulasi MPEG TS dengan hanya mengatur variasi *bit rate* tidak terlalu memberikan pengaruh terhadap hasil



*frame rate* pada *client*, bahkan dapat dikatakan tidak ada penurunan *frame rate* sama sekali. Sehingga dapat ditarik analisis bahwa variasi yang dilakukan terhadap besaran *bit rate* pada *setting* encoding yang dilakukan di sisi *streaming server* tidak terlalu berpengaruh mengubah *frame rate* di sisi *client*, melainkan untuk mengubah-ubah *bit rate* yang akan di-*streaming*, di mana hal ini telah ditunjukkan pada analisis *bit rate*. Kemungkinan rata-rata *frame rate* melebihi 29.97 fps karena ada keterlambatan sesaat sebelum memberhentikan VLC pada sisi *client* saat proses *streaming* pada saat pengambilan data. Sementara jumlah *displayed frames* pada VLC terus-menerus bertambah bila tidak di-stop walaupun layar telah menjadi hitam dan proses *streaming* telah berhenti pada VLC *client* (rata-rata pertambahannya antara 70 sampai 80 fps tiap detiknya). Sehingga kemungkinan ini mempengaruhi pada proses pencatatan data. Namun walaupun begitu, data rata-rata *frame rate* yang didapatkan masih memenuhi kriteria wajar, karena umumnya *file-file* video .mpg dan .mpv memiliki kurang lebih *frame rate* sebesar 30 fps.

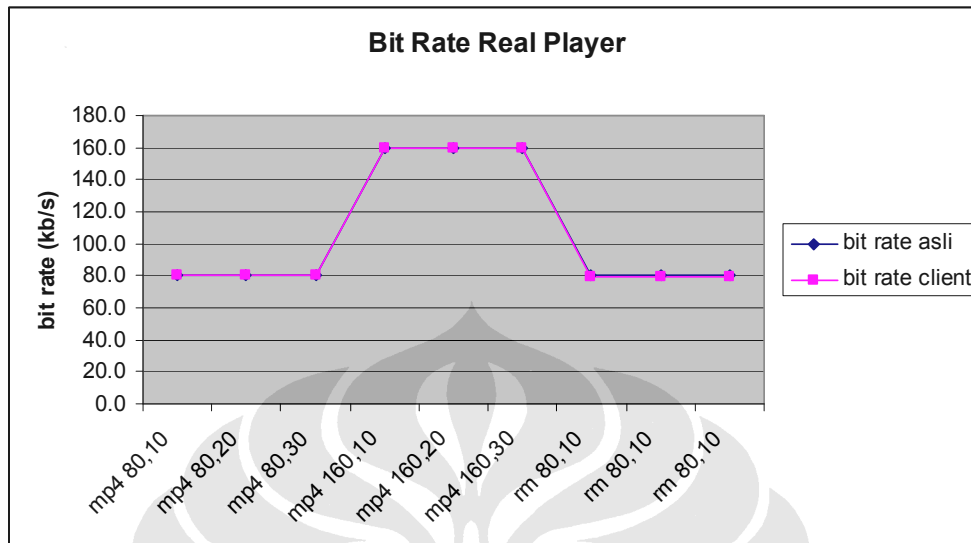
#### 4.6. ANALISIS BIT RATE HELIX PADA TIAP TOPOLOGI JARINGAN

Setelah dilakukan percobaan *streaming* dengan Helix *streaming server* dan Real Player sebagai *client* menggunakan *file-file* yang ingin diuji, maka dari data-data yang didapatkan terlihat bahwa terdapat kesamaan pada semua topologi jaringan bila menggunakan parameter *bit rate*. Kesamaannya adalah semua data *bit rate* yang terbaca oleh *tools* pada Real Player semuanya memiliki nilai keluaran *bit rate* yang sama untuk semua topologi jaringan yang ada. Hasil pencatatan data *bit rate* dapat dilihat pada Tabel 4.6.

Tabel 4.6. Tabel Helix-Real Perbandingan *Bit rate* asli dengan *Bit rate Client*

	mp4 80,10	mp4 80,20	mp4 80,30	mp4 160,10	mp4 160,20	mp4 160,30
<i>bit rate</i> asli (kb/s)	80.0	80.0	80.0	160	160	160
<i>bit rate client</i> (kb/s)	80.0	80.0	80.0	160	160	160
	rm 80,10	rm 80,20	rm 80,30			
<i>bit rate</i> asli (kb/s)	80.0	80.0	80.0			
<i>bit rate client</i> (kb/s)	79.3	79.3	79.3			

Hubungan antara *bit rate file* asli dengan *bit rate* hasil *streaming* dapat dibuat grafiknya untuk memudahkan analisis seperti Gambar 4.19.



Gambar 4.19. Grafik *Real Player* Perbandingan *Bit rate* Asli dengan *Bt Rate Streaming*

Seperti yang terlihat dari tabel dan grafik di atas, *bit rate* hasil *streaming* hampir sama persis dengan *bit rate* dari *file* asli. Bahkan yang berbeda dan terjadi penurunan *bit rate* hanya pada *streaming file-file* berformat *.rm* saja. Itupun tidak banyak dengan rata-rata penurunan *bit rate* (*.rm*) sebesar 0.29 %. Hal ini mengindikasikan bahwa penggunaan *Helix streaming server* dan *Real Player* sebagai *client* sudah sangat stabil, terbukti dengan hasil *bit rate streaming* yang didapatkan.

Perbedaan variasi besar *bit rate* pada *file* berformat *.mp4* tidak terlihat pengaruhnya, namun yang terlihat adalah pada perbedaan jenis format *file* yang di-*streaming* di mana terjadi sedikit penurunan *bit rate* pada *file* berformat *.rm* dan tidak terjadi pada *file* berformat *.mp4*.

Faktor penting yang harus dijadikan pertimbangan di sini adalah bahwa penggunaan enkapsulasi ke format *Realmedia* (*.rm*) sangat terasa kegunaannya. Disamping hanya terjadinya penurunan *bit rate* yang relatif sangat kecil, namun kualitas video dan audio hasil *streaming* mampu melebihi *file* berformat *.mp4* yang bervariasi *bit rate* lebih tinggi. *File-file* berformat *.mp4* tersebut memiliki ukuran *size* yang berkali lipat lebih besar dari *file Realmedia*. Tentunya ini dapat dijadikan pertimbangan untuk sebaiknya menggunakan enkapsulasi ke format

*Realmedia* karena dengan begitu komputer *server* akan bisa menghemat penggunaan *space* pada *hardisk*.

#### 4.7. ANALISIS *FRAME RATE* HELIX PADA TIAP TOPOLOGI JARINGAN

Proses *streaming* dengan *Helix streaming server* dan *Real Player* sebagai medianya menggunakan *file-file* berformat *.mp4* dan *.rm* yang divariasikan besar *frame rate*-nya, mulai dari 10, 20, dan 30 fps. Parameter *current frame* pada *tools Playback Statistic* di *Real Player* yang akan diperhatikan sebagai acuan untuk mendapatkan data *frame rate*. Penurunan atau reduksi *frame rate* merupakan selisih antara *frame rate file* asli (*server*) dengan *frame rate* hasil *streaming* (*client*).

Pada topologi IPv4 hasil pengolahan data rata-rata *frame rate* hasil *streaming* yang didapat untuk masing-masing file adalah seperti yang terlihat pada Tabel 4.7.

Tabel 4.7. Tabel Helix-Real *Frame Rate* Pada Topologi IPv4

<i>frame rate ipv4</i>			
	<i>frame rate file asli (fps)</i>	<i>frame rate streaming (fps)</i>	penurunan <i>frame rate (%)</i>
mp4 80,10	10	10	0%
mp4 80,20	20	19.24	4%
mp4 80,30	30	28.73	4%
mp4 160,10	10	9.53	5%
mp4 160,20	20	19.41	3%
mp4 160,30	30	28.67	4%
rm 80,10	10	10	0%
rm 80,20	20	18.99	5%
rm 80,30	30	29.11	3%

Tabel 4.7 di atas menunjukkan bagaimana perbandingan antara *frame rate* hasil *streaming* dibandingkan dengan *frame rate file* asli pada jaringan IPv4. Selanjutnya akan dibandingkan dengan *frame rate* hasil *streaming* pada topologi lain. Hasil pengolahan data rata-rata *frame rate* hasil *streaming* yang didapat untuk masing-masing file untuk topologi IPv6 adalah seperti yang terlihat pada Tabel 4.8.

Tabel 4.8. Tabel Helix-Real *Frame Rate* Pada Topologi IPv6

<i>frame rate ipv6</i>			
	<i>frame rate file asli (fps)</i>	<i>frame rate streaming (fps)</i>	penurunan <i>frame rate (%)</i>
mp4 80,10	10	9.52	5%
mp4 80,20	20	18.94	5%
mp4 80,30	30	28.65	5%
mp4 160,10	10	9.41	6%
mp4 160,20	20	19.26	4%
mp4 160,30	30	28.75	4%
rm 80,10	10	10	0%
rm 80,20	20	19.03	5%
rm 80,30	30	29.35	2%

Tabel 4.8 di atas menunjukkan bagaimana perbandingan antara *frame rate* hasil *streaming* dibandingkan dengan *frame rate file* asli pada jaringan IPv6. Selanjutnya akan dibandingkan dengan *frame rate hasil streaming* pada topologi 6to4. Hasil pengolahan data rata-rata *frame rate* hasil *streaming* yang didapat untuk masing-masing file untuk topologi 6to4 adalah seperti yang terlihat pada Tabel 4.9.

Tabel 4.9. Tabel Helix-Real *Frame Rate* Pada Topologi 6to4

<i>frame rate 6to4</i>			
	<i>frame rate file asli (fps)</i>	<i>frame rate streaming (fps)</i>	penurunan <i>frame rate (%)</i>
mp4 80,10	10	9.55	5%
mp4 80,20	20	19.01	5%
mp4 80,30	30	29.14	3%
mp4 160,10	10	9.19	8%
mp4 160,20	20	19.11	4%
mp4 160,30	30	28.86	4%
rm 80,10	10	10	0%
rm 80,20	20	19.00	5%
rm 80,30	30	29.18	3%

Dari tabel-tabel sebelumnya dapat diketahui seberapa besar terjadinya penurunan *frame rate* pada hasil *streaming* untuk masing-masing topologi. Bila kita jadikan persen rata-rata penurunan *frame rate* sebagai suatu parameter ukur maka topologi dengan kinerja *streaming* yang terbaik adalah topologi IPv4 dengan rata-rata penurunan *frame rate* sebesar 3 %, diikuti oleh topologi IPv6 dan 6to4 dengan rata-rata penurunan *frame rate* sebesar 4 %.

Penurunan *frame rate* ini tidak terjadi selama interval durasi penuh *streaming file*, namun hanya pada bagian-bagian tertentu saja ada yang mengalami penurunan *frame rate*. Sehingga penurunan *frame rate* ini tidak sepenuhnya mempengaruhi kualitas video dari hasil *streaming* hanya parsial saja.

Jika diperhatikan maka *frame rate* yang paling kualitasnya paling menurun adalah *streaming* dengan menggunakan format misalkan .mp4 yang memiliki *bit rate* 160 kb/s dan *frame rate* 10 fps. File ini adalah file yang berukuran cukup besar *size*-nya tapi memiliki *frame rate* lebih kecil jika dibandingkan dengan misalkan *file* berformat .rm yang memiliki *frame rate* lebih besar 30 fps sementara *size*-nya lebih kecil. Hal ini mengindikasikan bahwa semakin besar *bit rate* (yang berarti semakin besar juga ukuran *size file*) tanpa diikuti oleh besar *frame rate* yang memadai akan semakin mengalami penurunan *frame rate*. Namun hal ini dapat diatasi dengan menggunakan metode enkapsulasi atau konversi *file* ke format lain yang lebih cocok.

#### **4.8. PERBANDINGAN KINERJA KESELURUHAN *STREAMING* MENGGUNAKAN VLC DAN HELIX *STREAMING SERVER***

Dari hasil pengolahan data yang didapatkan, jika melakukan *streaming* dengan menggunakan VLC didapatkan parameter ukur penurunan atau reduksi *bit rate* maka jaringan IPv6 memiliki rata-rata reduksi *bit rate* = 9.98 % dengan  $\sigma = 0.5$  %, topologi jaringan IPv4 memiliki rata-rata reduksi *bit rate* = 11.25 % dengan  $\sigma = 0.54$  %. Penerapan tunneling 6to4 tidak terlalu buruk kinerjanya dengan rata-rata reduksi *bit rate* = 11.43 % dengan  $\sigma = 0.36$  %. Sementara jika dibandingkan dengan hasil *streaming* menggunakan Helix *streaming server* didapatkan rata-rata penurunan atau reduksi *bit rate* yang sangat kecil untuk keseluruhan topologi jaringan, rata-rata total reduksi *bit rate* sebesar 0.29 %. Ini mengindikasikan performa Helix di atas VLC.

Bila parameter lain yaitu *frame rate* yang dijadikan acuan kinerja *streaming* maka VLC dan Helix bisa dikatakan tidak banyak berbeda performanya. Keduanya sama-sama bekerja dengan baik dalam proses *streaming*.

Faktor-faktor lain yang perlu dipertimbangkan adalah pada tingkat melakukan konfigurasi untuk *streaming* tidak terlalu sulit VLC dan lebih mudah

untuk diimplementasikan. Namun fitur yang disediakan juga lebih sedikit jika dibandingkan dengan Helix yang memiliki fitur autentikasi, monitoring, firewall, dan lain-lain dalam proses *streaming*. Fitur koneksi seperti misalnya protokol HTTP pada VLC juga masih belum bisa diimplementasikan pada jaringan IPv6, hal ini menjadikan suatu keterbatasan juga dalam melakukan *streaming*.

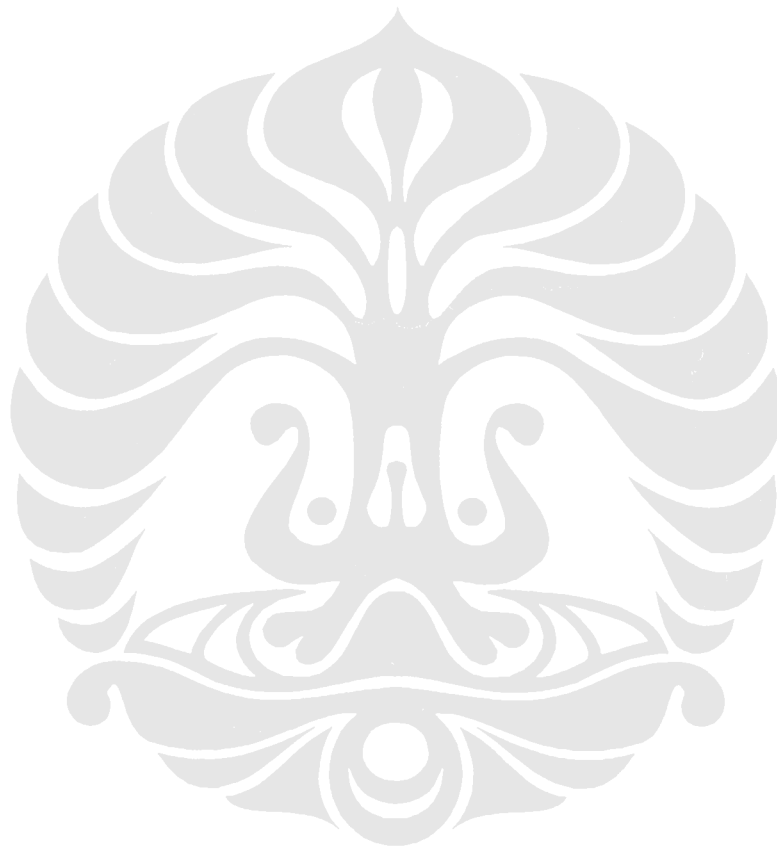
Sisi ekonomisnya adalah penggunaan VLC tidak dikenakan biaya karena bersifat *freeware* sedangkan Helix memerlukan license dalam jika ingin diterapkan dalam skala besar. Salah satu faktor lemah Helix adalah kompatibilitas yang masih terbatas pada platform Windows Server 2003 yang umumnya rentan terhadap masalah dalam jaringan, namun untungnya kini telah dirilis versi Linux. Sementara VLC sudah tersedia di berbagai platform lain seperti Mac dan Linux selain di Windows.

Untuk penerapannya dalam jaringan, cara kerja *streaming* dengan menggunakan VLC adalah *server* bertindak sebagai pengumpan *file streaming*. Kelemahannya adalah bila *server* tidak mengumpan *file* tersebut untuk di-*streaming*, maka VLC *client* tidak dapat menerima atau pun mengakses suatu *file streaming* yang dibutuhkan saat itu juga. *Client* harus menunggu samapai *administrator server* mengumpan *file* tersebut. Lain halnya pada Helix, *server* tidak bertindak sebagai pengumpan, melainkan merespon terhadap suatu *request streaming* dari *client*. Sehingga proses *streaming* dapat terjadi kapan pun *client* ingin langsung mengakses *file streaming* yang dibutuhkan.

Yang penting untuk diperhatikan adalah penerapan *streaming* dengan VLC dan Helix ini masih dilakukan dalam jaringan skala kecil. Bila nantinya akan diterapkan pada jaringan yang berskala besar, di mana terdapat banyak sekali *client* yang akan mengakses *server* serta terdapat fitur-fitur *security* dan *redundancy* yang lebih rumit pada jaringan, maka parameter-parameter seperti penurunan atau reduksi *bit rate* dan *frame rate* harus menjadi bahan pertimbangan. Ke depannya diharapkan *deployment streaming menggunakan VLC dan Helix* akan bisa diterapkan pada level jaringan yang lebih *advance*.

Penerapan VLC cocok pada jaringan yang berskala kecil dan tidak terlalu menerapkan autentikasi dan *firewall* sebagai fitur keamanan jaringan. Namun bila

jaringan yang digunakan adalah jaringan berskala besar seperti misalnya Kampus Universitas Indonesia, maka *Helix streaming server* lebih cocok untuk diterapkan.



## BAB V

### KESIMPULAN

1. Aplikasi *video streaming* menggunakan VLC dan *Helix Streaming Server* berhasil diterapkan pada konfigurasi topologi jaringan IPv6 dan *tunneling 6to4*.
2. Hasil *streaming* menggunakan VLC mengindikasikan bahwa topologi jaringan IPv6 memiliki kinerja terbaik dengan rata-rata reduksi *bit rate* = 9.98 % dengan  $\sigma = 0.5$  %, diikuti dengan topologi jaringan IPv4 dengan rata-rata reduksi *bit rate* = 11.25 % dengan  $\sigma = 0.54$  %. Penerapan *tunneling 6to4* juga tidak terlalu buruk kinerjanya dengan rata-rata reduksi *bit rate* = 11.43 % dengan  $\sigma = 0.36$  %.
3. Hasil *streaming* menggunakan VLC tidak mengindikasikan adanya penurunan *frame rate*.
4. Hasil *streaming* menggunakan *Helix streaming server* didapatkan rata-rata penurunan atau reduksi *bit rate* sebesar 0.29 % untuk keseluruhan topologi jaringan yang digunakan.
5. Persen rata-rata penurunan *frame rate* suatu parameter ukur dengan *streaming* menggunakan *Helix* sebagai untuk topologi IPv4 rata-rata penurunan *frame rate* sebesar 3 %, diikuti oleh topologi IPv6 dan 6to4 dengan rata-rata penurunan *frame rate* sebesar 4 %.
6. *Video Streaming* menggunakan *Helix* lebih stabil dibandingkan menggunakan VLC jika mengacu pada hasil parameter *bit rate* dan *frame rate* yang didapatkan saat uji coba.

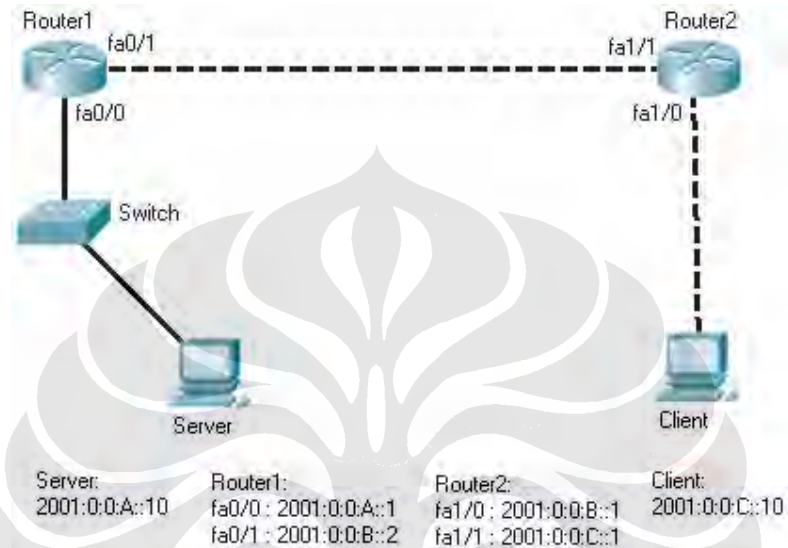


## DAFTAR ACUAN

- [1]. CISCO Networking Academy, “**Network Fundamentals**”, CCNA Exploration 4.0 (CISCO SYSTEMS, Inc., 2007), bab 6.3.6
- [2]. “**How IPv6 Works**” (Microsoft Technet, March 2003)
- [3]. “**Introduction to IP Version 6**” (Microsoft Corporation, updated January 2008)
- [4]. S. Deering, R. Hinden, “**Internet Protocol Version 6 (IPv6) Addressing Architecture**” (IETF, April 2003), RFC 3513
- [5]. Lorenzo Colitti, *et al.*, “**IPv6-in-IPv4 tunnel discovery: methods and experimental results**” (IEEE Transactions on Network and Service Management, vol. 1, no. 1, April 2004)
- [6]. Jean-Francois Tremblay, Mikael Lind, “**IPv6 Tunneling Techniques**”, (Hexago Inc., September 2006)
- [7]. “**IPv6 Transition Technologies**” (Microsoft Corporation, February 2008)
- [8]. David Austerberry, “**The Technology of Video & Audio Streaming**”, Chapter 2, (Focal Press, 1998)
- [9]. Helix Streaming Server manual guidance (help)
- [10]. [http://en.wikipedia.org/wiki/Bit\\_rate](http://en.wikipedia.org/wiki/Bit_rate)
- [11]. <http://www.videohelp.com/calc.htm>
- [12]. [http://searchnetworking.techtarget.com/sDefinition/0,,sid7\\_gci213531,00.html](http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci213531,00.html)

# LAMPIRAN

## LAMPIRAN 1: KONFIGURASI JARINGAN IPV6 MURNI



### Server

```
C:\> ipv6 install
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:0:0:A::10
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:0:0:A::1
```

### Client

```
C:\> ipv6 install
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:0:0:C::10
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:0:0:C::1
```

### Router1

```
Router1>enable
Router1#configure terminal
Router1(config)#ipv6 unicast routing
Router1(config)#interface fastethernet 0/0
Router1(config-if)#ipv6 enable
Router1(config-if)#ipv6 address 2001:0:0:A::1/64
Router1(config-if)#ipv6 rip process1 enable
Router1(config-if)#no shutdown
Router1(config)#interface fastethernet 0/1
Router1(config-if)#ipv6 enable
Router1(config-if)#ipv6 address 2001:0:0:B::2/64
Router1(config-if)#ipv6 rip process1 enable
Router1(config-if)#no shutdown
```

### Router2

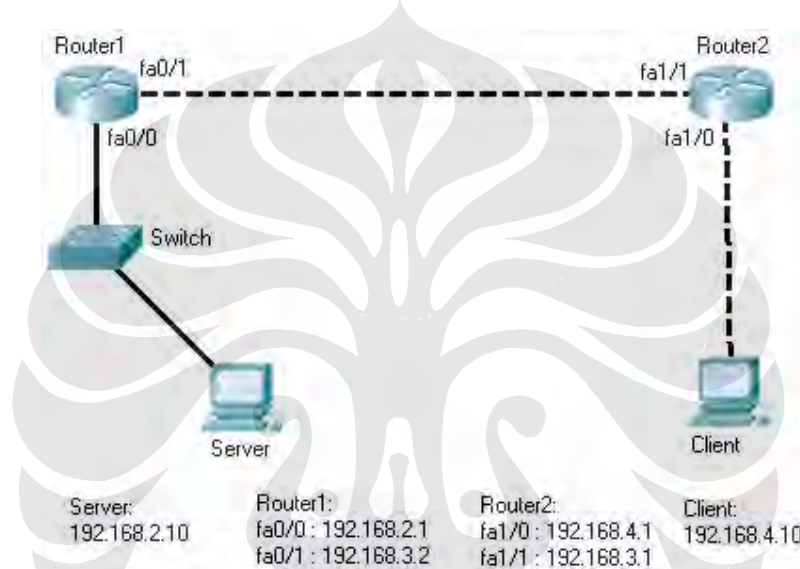
```
Router2>enable
Router2#configure terminal
```

```

Router2(config)#ipv6 unicast routing
Router2(config)#interface fastethernet 1/0
Router2(config-if)#ipv6 enable
Router2(config-if)#ipv6 address 2001:0:0:C::1/64
Router2(config-if)#ipv6 rip process1 enable
Router2(config-if)#no shutdown
Router2(config)#interface fastethernet 1/1
Router2(config-if)#ipv6 enable
Router2(config-if)#ipv6 address 2001:0:0:B::2/64
Router2(config-if)#ipv6 rip process1 enable
Router2(config-if)#no shutdown

```

## LAMPIRAN 2: KONFIGURASI JARINGAN IPv4 MURNI



### Server

```

C:\> netsh interface ip add address "Local Area Connection"
192.168.2.10 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=192.168.4.1 gwmetric=0

```

### Client

```

C:\> netsh interface ip add address "Local Area Connection"
192.168.4.10 255.255.255.0
C:\> netsh interface ip add address "Local Area Connection"
gateway=192.168.4.1 gwmetric=0

```

### Router1

```

Router1>enable
Router1#configure terminal
Router1(config)#interface fastethernet 0/0
Router1(config-if)#ip address 192.168.2.1 255.255.255.0
Router1(config-if)#no shutdown
Router1(config)#interface fastethernet 0/1
Router1(config-if)#ip address 192.168.3.2 255.255.255.0
Router1(config-if)#no shutdown
Router1(config)#router rip
Router1(config-router)#version 2
Router1(config-router)#network 192.168.2.0
Router1(config-router)#network 192.168.3.0
Router1(config-router)#passive-interface fastethernet 0/0

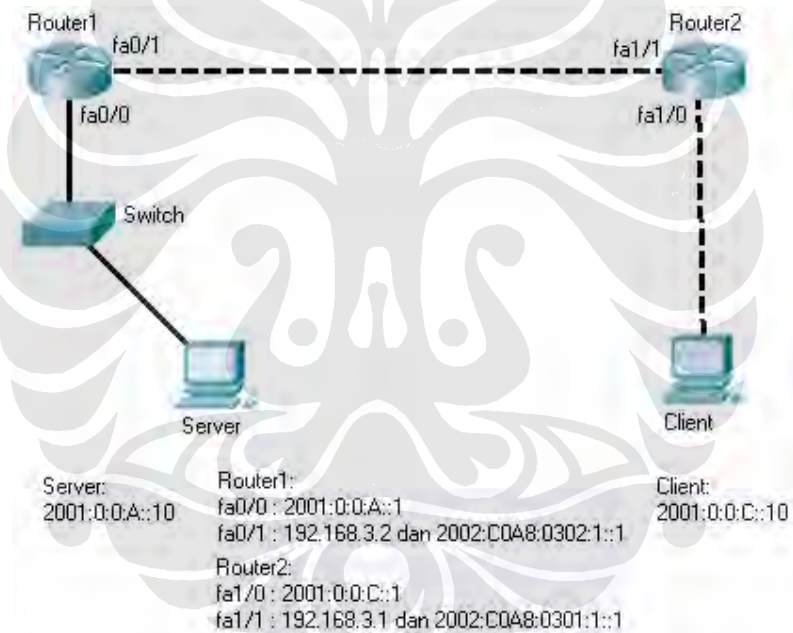
```

```
Router1(config-router)#no auto-summary
```

### Router2

```
Router2>enable
Router2#configure terminal
Router2(config)#interface fastethernet 1/0
Router2(config-if)#ip address 192.168.4.1 255.255.255.0
Router2(config-if)#no shutdown
Router2(config)#interface fastethernet 1/1
Router2(config-if)#ip address 192.168.3.1 255.255.255.0
Router2(config-if)#no shutdown
Router2(config)#router rip
Router2(config-router)#version 2
Router2(config-router)#network 192.168.3.0
Router2(config-router)#network 192.168.4.0
Router2(config-router)#passive-interface fastethernet 1/0
Router2(config-router)#no auto-summary
```

## LAMPIRAN 3: KONFIGURASI JARINGAN 6to4



### Server

```
C:\> ipv6 install
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:0:0:A::10
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:0:0:A::1
```

### Client

```
C:\> ipv6 install
C:\> netsh interface ipv6 add address "Local Area Connection"
2001:0:0:C::10
C:\> netsh interface ipv6 add route ::/0 "Local Area Connection"
2001:0:0:C::1
```

### Router1

```
Router1>enable
Router1#configure terminal
```

```
Router1(config)#ipv6 unicast routing
Router1(config)#interface fastethernet 0/0
Router1(config-if)#ipv6 enable
Router1(config-if)#ipv6 address 2001:0:0:A::1/64
Router1(config-if)#ipv6 rip process1 enable
Router1(config-if)#no shutdown
Router1(config)#interface fastethernet 0/1
Router1(config-if)#ipv6 enable
Router1(config-if)#ip address 192.168.3.2 255.255.255.0
Router1(config-if)#ipv6 address 2002:C0A8:0302:1::1/64
Router1(config)#interface tunnel 0
Router1(config-if)#ipv6 enable
Router1(config-if)#no ip address
Router1(config-if)#ipv6 unnumbered fastethernet 0/1
Router1(config-if)#tunnel source fastethernet 0/1
Router1(config-if)#tunnel mode ipv6ip 6to4
Router1(config)#router rip
Router1(config-router)#version 2
Router1(config-router)#network 192.168.3.0
Router1(config)#ipv6 route 2002::/16 tunnel 0
Router1(config)#ipv6 route ::/0 2002:C0A8:0301:1::1
```

## **Router2**

```
Router2>enable
Router2#configure terminal
Router2(config)#ipv6 unicast routing
Router2(config)#interface fastethernet 1/0
Router2(config-if)#ipv6 enable
Router2(config-if)#ipv6 address 2001:0:0:C::1/64
Router2(config-if)#ipv6 rip process1 enable
Router2(config-if)#no shutdown
Router2(config)#interface fastethernet 1/1
Router2(config-if)#ipv6 enable
Router2(config-if)#ip address 192.168.3.1 255.255.255.0
Router2(config-if)#ipv6 address 2002:C0A8:0301:1::1/64
Router2(config)#interface tunnel 0
Router2(config-if)#ipv6 enable
Router2(config-if)#no ip address
Router2(config-if)#ipv6 unnumbered fastethernet 1/1
Router2(config-if)#tunnel source fastethernet 1/1
Router2(config-if)#tunnel mode ipv6ip 6to4
Router2(config)#router rip
Router2(config-router)#version 2
Router2(config-router)#network 192.168.3.0
Router2(config)#ipv6 route 2002::/16 tunnel 0
Router1(config)#ipv6 route ::/0 2002:C0A8:0302:1::1
```

**LAMPIRAN 4: RATA-RATA DATA *BIT RATE* PADA *VLC CLIENT***

		mpeg 1024, 192	mpeg 512, 96	mpeg 216, 32	mp4 1024,192	mp4 512, 96	mp4 216, 32
ipv6	send <i>bit rate</i>	764.255	666.973	367.827	1251.545	680.318	363.509
	stream <i>bit rate</i>	705.118	600.345	297.018	1200.755	669.355	300.955
	reduksi <i>bit rate</i> (%)	7.74%	9.99%	19.25%	4.06%	1.61%	17.21%
	penurunan <i>bit rate</i>	59.136	66.627	70.809	50.791	10.964	62.555
ipv4	send <i>bit rate</i>	788	646.873	370.827	1506.936	685.891	378.336
	penurunan <i>bit rate</i>	725.555	627.091	307.836	1216.782	676.455	306.918
	reduksi <i>bit rate</i> (%)	7.92%	3.06%	16.99%	19.25%	1.38%	18.88%
	penurunan <i>bit rate</i>	62.445	19.782	62.991	290.155	9.436	71.418
6to4	send <i>bit rate</i>	758.409	663.327	358.545	1276.382	672.773	370.645
	<i>bit rate loss</i>	696.600	605.755	297.918	1194.955	613.482	297.727
	reduksi <i>bit rate</i> (%)	8.15%	8.68%	16.91%	6.38%	8.81%	19.67%
	penurunan <i>bit rate</i>	61.809	57.573	60.627	81.427	59.291	72.918

**LAMPIRAN 5: RATA-RATA DATA *FRAME RATE* PADA *VLC CLIENT***

	<i>frame rate</i> (fps)		
	ipv4	ipv6	6to4
mpeg 1024, 192	30.41	30.18	30.74
mpeg 512, 96	29.66	30.70	30.44
mpeg 216, 32	30.28	30.69	30.07
mp4 1024,192	30.10	30.43	30.74
mp4 512, 96	29.66	30.53	30.43
mp4 216, 32	31.13	30.38	30.95
rata-rata <i>frame rate</i>	30.21	30.48	30.56

**LAMPIRAN 6: RATA-RATA DATA FRAME RATE PADA HELIX-REAL  
PLAYER**

		ipv4		ipv6		6to4	
	<i>frame rate asli (fps)</i>	<i>frame rate streaming (fps)</i>	<i>penurunan frame rate (%)</i>	<i>frame rate streaming (fps)</i>	<i>penurunan frame rate (%)</i>	<i>frame rate streaming (fps)</i>	<i>penurunan frame rate (%)</i>
mp4 80,10	10	10	0%	9.52	5%	9.55	5%
mp4 80,20	20	19.24	4%	18.94	5%	19.01	5%
mp4 80,30	30	28.73	4%	28.65	5%	29.14	3%
mp4 160,10	10	9.53	5%	9.41	6%	9.19	8%
mp4 160,20	20	19.41	3%	19.26	4%	19.11	4%
mp4 160,30	30	28.67	4%	28.75	4%	28.86	4%
rm 80,10	10	10	0%	10.00	0%	10.00	0%
rm 80,20	20	18.99	5%	19.03	5%	19.00	5%
rm 80,30	30	29.11	3%	29.35	2%	29.18	3%
		<i>rata-rata penurunan frame rate</i>	3%	<i>rata-rata penurunan frame rate</i>	4%	<i>rata-rata penurunan frame rate</i>	4%





**LAMPIRAN 8: CONTOH PENCATATAN SAMPEL DATA *FRAME RATE*  
VLC TIAP 5 DETIK**

	<b>6to4 mp4 512,96</b>			
percobaan/data	<b>server</b>		<b>client</b>	
	displayed frame	frame rate	displayed frame	frame rate
1	1799	29.97	1817	30.283333
2	1799	29.97	1853	30.883333
3	1799	29.97	1851	30.85
4	1799	29.97	1795	29.916667
5	1799	29.97	1825	30.416667
6	1799	29.97	1833	30.55
7	1799	29.97	1894	31.566667
8	1799	29.97	1790	29.833333
9	1799	29.97	1798	29.966667
10	1799	29.97	1804	30.066667
		29.97		30.433333

Catatan: frame rate = displayed frame / durasi file (60 detik)

**LAMPIRAN 9: CONTOH PENCATATAN SAMPEL DATA *BIT RATE*  
HELIX-REAL PLAYER TIAP 5 DETIK**

<b>6to4</b>	<b>Current frame (real player-client)</b>				
	<b>ipv6 - rm 80,20</b>				
Detik\n	1	2	3	4	5
5	19.2	19.6	19.6	19	19.6
10	19.3	18.8	19	19	18.9
15	19	19	18.9	19	18.4
20	19	18.7	18.7	18.5	18.7
25	19	19.2	18.9	19	19
30	18.9	19	19.2	19	19
35	18.9	18.8	18.9	19	19
40	19.2	19.2	19	19	19
45	19	19	19	18.8	19
50	19.2	18.8	18.8	19.2	18.8
55	18.9	19.2	19.2	19	19.2
total	209.6	209.3	209.2	208.5	208.6
rata-rata	19.05455	19.02727	19.01818	18.95455	18.96364
	rata-rata frame rate				19.00364
	frame rate asli				20

