PERFORMANSI VOIP
(VOICE OVER INTERNET PROTOCOL)
DENGAN SIP *(SESSION INITIATION PROTOCOL)*
MELALUI VPN *(VIRTUAL PRIVATE NETWORK)*

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

YUYUN TRIHANDINI
NIM. 0510632034

DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2009

LEMBAR PERSETUJUAN

PERFORMANSI VOIP *(VOICE OVER INTERNET PROTOCOL)*
DENGAN SIP *(SESSION INITIATION PROTOCOL)*
MELALUI VPN *(VIRTUAL PRIVATE NETWORK)*

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

YUYUN TRIHANDINI
NIM. 0510632034

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Rusmi Ambarwati, ST., MT.	**Dwi Fadila K., ST., MT.**
NIP. 19720204 200003 2 002	**NIP. 19720630 200003 1 002**

LEMBAR PENGESAHAN

PERFORMANSI VOIP *(VOICE OVER INTERNET PROTOCOL)*
DENGAN SIP *(SESSION INITIATION PROTOCOL)*
MELALUI VPN *(VIRTUAL PRIVATE NETWORK)*

SKRIPSI
KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

YUYUN TRIHANDINI
NIM. 0510632034

Skripsi ini telah diuji dan dinyatakan lulus
pada tanggal 3 Agustus 2009

MAJELIS PENGUJI

Ir. Wahyu Adi Priyono, M.Sc.	**Ir. Erfan Achmad Dahlan, MT.**
NIP. 19600518 198802 1 001	**NIP. 19530714 198203 1 003**

Ali Mustofa, ST., MT.
NIP. 19710601 200003 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro,

Ir. Heru Nurwarsito, M.Kom.
NIP. 19650402 199002 1 001

ABSTRAK

YUYUN TRIHANDINI, 0510632034
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, 2009
Performansi VoIP (Voice over Internet Protocol) dengan SIP (Session Initiation Protocol) melalui VPN (Virtual Private Network)
Dosen Pembimbing:
Rusmi Ambarwati, ST., MT. dan Dwi Fadila K, ST., MT.

VoIP menggunakan pensinyalan SIP telah diterapkan secara luas karena sifatnya yang lebih sederhana dibanding protokol pensinyalan lain. Salah satu keuntungan VoIP adalah komunikasi suara dapat dilakukan melalui jaringan data yang sudah ada, dapat bersifat *private* atau publik (internet). Namun jaringan internet memungkinkan pihak lain untuk melakukan penyadapan dan perekaman data VoIP. Karena itu dibentuk suatu sistem jaringan *private* yang berjalan di atas jaringan publik yang telah ada sebagai media penghubung antar jaringan-jaringan privat, yaitu VPN.

Tujuan dari kajian ini adalah untuk menganalisis performansi sistem VoIP dengan SIP melalui jaringan VPN. Parameter-parameter yang diamati meliputi *delay end to end, packet loss, throughput, jitter,* dan kualitas suara pada tiga konfigurasi jaringan yang berbeda.

Konfigurasi yang digunakan adalah konfigurasi Non-VPN, VPN *less secure,* dan VPN *most secure.* Ketiga konfigurasi tersebut dibedakan berdasarkan algoritma enkripsi yang digunakan. Data diperoleh dari pengujian sistem dan perhitungan secara teoritis. Pengambilan data pada pengujian menggunakan *Wireshark.*

Pada analisis pengujian didapatkan nilai *delay end to end, jitter,* dan *packet loss* pada konfigurasi VPN-*most* paling tinggi, yaitu 22,92 ms, 9,43 ms, dan 0,20%. Konfigurasi VPN-*most* memiliki nilai *throughput* terendah sebesar 79,98 kbps. Nilai MOS dari *polling* sebesar 3,90 untuk Non-VPN, 3,24 untuk VPN-*less,* dan 2,72 untuk VPN-*most.* Nilai MOS dari perhitungan E-Model sebesar 4,29. Berdasarkan besar *delay, jitter, packet loss, throughput,* dan kualitas suara dapat diketahui bahwa performansi sistem VoIP melalui VPN masih *reliable,* meskipun ada perbedaan tetapi tidak terlalu signifikan dengan sistem Non-VPN. Dengan acuan rekomendasi ITU-T G.1010, komunikasi VoIP dengan SIP melalui VPN menghasilkan kualitas performansi yang tergolong baik.

Kata kunci: VoIP, SIP, VPN

KATA PENGANTAR

Syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT. karena rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi dengan judul "Performansi VoIP *(Voice over Internet Protocol)* dengan SIP *(Session Initiation Protocol)* melalui VPN *(Virtual Private Network)*".

Penulis menyadari bahwa kajian ini tidak akan mencapai titik akhir penyelesaian tanpa bantuan berbagai pihak, karenanya penulis mengucapkan terima kasih kepada:

1. Bapak Ir. Heru Nurwarsito, M.Kom. selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya Malang.
2. Bapak Rudy Yuwono, ST., M.Sc. selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya Malang.
3. Ibu Ir. Endah Budi P., MT. selaku KKDK Telekomunikasi yang selalu memberikan motovasi.
4. Ibu Rusmi Ambarwati, ST., MT. selaku dosen pembimbing 1 pada penyusunan skripsi ini.
5. Bapak Dwi Fadila K., ST., MT. selaku dosen pembimbing 2 pada penyusunan skripsi ini.
6. Bapak, Ibu, dan Kakak-kakakku beserta keponakanku yang lucu-lucu untuk seluruh do'a, dukungan, dan semua cinta yang telah diberikan hingga terselesaikannya skripsi ini.
7. Teman-teman TPT UB, terutama Galih Kusumo Budiono yang sangat berperan mendukung penyelesaian skripsi ini.
8. Seluruh teman-teman senasib seperjuangan SAP '05 yang selalu memberi spirit dan semangat dalam setiap langkahku.
9. Temen-temen kos Genk-G, Erna, Ani, Norma, dan Ayik yang selalu menemaniku dalam suka dan duka.
10. Serta semua pihak yang telah turut membantu baik secara langsung maupun tidak langsung dalam penyelesaian skripsi ini.

Tak ada gading yang tak retak. Tiada yang sempurna di dunia ini, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan. Oleh karena itu dengan segala kerendahan hati, penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak demi penyempurnaan skripsi ini.

Akhir kata, penulis berharap semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, Agustus 2009

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

3DES : *Triple Data Encryption Standard*
ACK : *Acknowledgement*
AES : *Advanced Encryption Standard*
CBC : *Cipher Block Chaining*
CLI : *Command Line Interface*
CSeq : *Command Sequence*
DES : *Data Encryption Standard*
ECB : *Electronic Code Book*
FIPS : *Federal Information Processing Standard*
FTP : *File Transfer Protocol*
HTTP : *Hypertext Transfer Protocol*
IETF : *Internet Engineering Task Force*
IP : *Internet Protocol*
ISO : *International Organization for Standartdization*
ITU : *International Telecommunication Union*
LTS : *Long Time Support*
LZO : *Lempel-Ziv-Oberhumer*
MD5 : *Message Digest 5*
MOS : *Mean Opinion Score*
NAT : *Network Address Translation*
NIC : *Network Interface Card*
NSA : *National Security Agency*
OSI : *Open System Interconnection*
PCM : *Pulse Code Modulation*
QoS : *Quality of Service*
RTCP : *Real-Time Transport Control Protocol*
RTP : *Realtime Transport Protocol*
RTP : *Real-Time Transport Protocol*
SHA-1 : *Secure Hash 1*
SIP : *Session Initiation Protocol)*

SSL : *Security Socket Layer*
TCP : *Transfer Control Protocol*
TDEA : *Triple Data Encryption Algorithm*
ToS : *Type of Service*
UDP : *User Datagram Protocol*
URI : *Universal Request Identifier*
UTP : *Unshielded Twisted Pair*
VoIP : *Voice over Internet Protocol*
VPN : *Virtual Private Network*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi telekomunikasi, banyak layanan multimedia telah dikembangkan di internet. Salah satu dari layanan itu adalah VoIP *(Voice over Internet Protocol)*. Salah satu keuntungan VoIP adalah komunikasi suara dapat dilakukan melalui jaringan data yang sudah ada, dimana jaringan ini dapat bersifat *private* atau publik. Jaringan publik tersebut yang lebih dikenal dengan internet. Perusahaan-perusahaan telekomunikasi telah menerapkan VoIP, salah satunya dalam telekomunikasi suara internasional dan menjadikan tarif telekomunikasi internasional relatif lebih murah jika dibandingkan telekomunikasi melalui telepon konvensional (M. Iskandarsyah H, 2005: 1). Namun jaringan internet memungkinkan pihak lain untuk melakukan penyadapan dan perekaman data VoIP. Karena itu dibentuk suatu sistem jaringan *private* yang berjalan di atas jaringan publik yang telah ada sebagai media penghubung antar jaringan-jaringan *private*, yaitu VPN.

VPN *(Virtual Private Network)* memungkinkan terbentuknya sebuah jaringan data *private* pada jaringan publik. Istilah *private* berkaitan dengan virtual. Membangun *virtual network* di atas infrastruktur yang dapat diakses publik memberikan beberapa implikasi sekuriti dimana informasi sensitif tidak ingin dibaca oleh pengguna lain yang tidak berhak. Sehingga diciptakan mekanisme untuk menjaga informasi tetap bersifat terbatas. *Enkripsi* merupakan salah satu solusinya, agar hanya pihak-pihak tertentu yang dipercaya yang dapat mengakses informasi. Sistem ini menjadikan komunikasi VoIP yang aman dapat dilakukan.

Dengan menggunakan SIP *(Session Initiation Protocol)* sebagai protokol pensinyalan pada VoIP memberikan kelebihan jika dibandingkan dengan protokol pensinyalan pendahulunya, yaitu H.323. SIP memiliki kelebihan mudah dikembangkan bersama aplikasi-aplikasi internet. Protokol pada SIP mudah dibaca, sederhana, dan merupakan protokol yang sifatnya *request-response*. SIP secara struktur lebih cepat dari H.323. SIP menggunakan lebih sedikit waktu dalam *setting-up* panggilan. Protokol SIP dibuat berdasarkan teks yang

membuatnya lebih mudah dianalisis sedangkan H.323 berdasarkan kode *software* biner yang sulit dibaca dan dimengerti (Hendra Cahya, 2007: 1).

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka dapat dirumuskan permasalahan sebagai berikut :

1. Bagaimana pengaruh penerapan konfigurasi Non-VPN dan VPN (*less-secure* dan *most-secure)* terhadap kualitas VoIP (*delay*, *throughput*, *jitter*, dan *packet loss*).
2. Bagaimana pengaruh penerapan konfigurasi Non-VPN dan VPN (*less-secure* dan *most-secure)* terhadap kualitas suara.
3. Konfigurasi manakah yang memiliki performansi lebih baik untuk komunikasi dengan VoIP.

1.3 Batasan Masalah

Untuk menjaga agar tidak melebarnya masalah yang dibuat dalam skripsi ini, maka penulis membatasi permasalahan yang dibahas, yaitu :

1. Sistem diterapkan pada empat buah komputer yang terhubung melalui kabel UTP, dimana satu berfungsi sebagai *server*, satu berfungsi sebagai *router*, dan dua sebagai *client*.
2. Sistem VoIP yang digunakan adalah SIP *signalling*.
3. Sistem VPN menggunakan aplikasi *Open*VPN GUI versi 1.0.3.
4. *Server* menggunakan sistem operasi *Ubuntu* 8.04 beserta aplikasi utama *Asterisk* VoIP *Server* dan *Open*VPN *Server*.
5. *Router* dan *Client* menggunakan sistem operasi *Windows XP* beserta aplikasi utama *X-Lite* VoIP *Client* dan *Open*VPN *Client* .
6. Pengujian dan pengambilan data dilakukan pada skala laboratorium.
7. Pengukuran parameter dilakukan pada dua *client* yang melakukan komunikasi VoIP.
8. Pada pengujian menggunakan tiga konfigurasi yaitu, Non-VPN, VPN *less-secure*, dan VPN *most-secure*.
9. *Codec* yang digunakan adalah G.711 PCMU.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah untuk menganalisis performansi sistem VoIP *(Voice over Internet Protocol)* dengan SIP *(Session Initiation Protocol)* melalui jaringan VPN *(Virtual Private Network).*

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan laporan tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Membahas konsep dasar yang berkaitan langsung dengan perencanaan sistem VoIP melalui VPN.

BAB III Metodologi Penelitian

Membahas tentang metodologi yang dilakukan dalam rangka pencapaian tujuan penulisan skripsi ini.

BAB IV Implementasi Sistem

Menjelaskan tentang konfigurasi *client* dan *server* beserta instalasi, baik secara *hardware* maupun *software*.

BAB V Pengujian dan Pengambilan Data

Membahas pelaksanaan pengujian sistem dan langkah-langkah dalam pengambilan data.

BAB VI Analisis Data

Menjelaskan tentang analisis performansi sistem, meliputi: pengujian (*delay, throughput, jitter,* dan *packet loss*) serta perhitungan (*bandwidth* yang dibutuhkan, *delay end to end*, dan kualitas suara).

BAB VII Penutup

Merupakan bab penutup yang berisi kesimpulan dan saran.

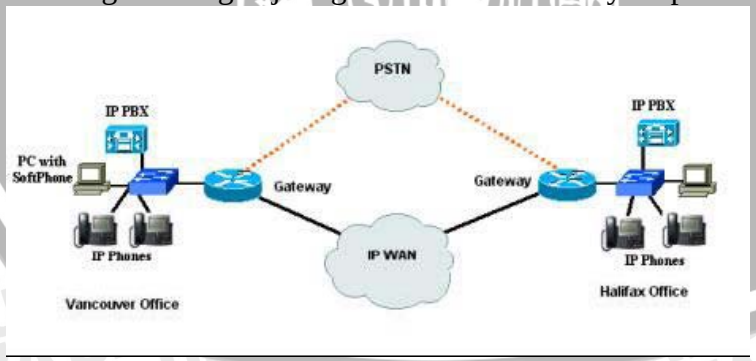
BAB II
TINJAUAN PUSTAKA

2.1 VoIP *(Voice over Internet Protocol)*

2.1.1 Konsep Dasar VoIP

VoIP adalah teknologi yang mampu melewatkan trafik suara yang berbentuk paket melalui jaringan IP. Jaringan IP sendiri adalah merupakan jaringan komunikasi data yang berbasis *packet switch*, jadi dalam bertelepon menggunakan jaringan IP atau internet. Dengan bertelepon menggunakan VoIP, banyak keuntungan yang dapat diambil, diantaranya adalah dari segi biaya jelas lebih murah daripada tarif telepon konvensional, karena jaringan IP bersifat global. Sehingga untuk hubungan internasional dapat ditekan hingga 70%. Selain itu, biaya *maintenance* dapat ditekan karena *voice* dan data *network* terpisah, sehingga IP *phone* dapat ditambah, dipindah, dan diubah. Hal ini karena VoIP dapat dipasang di sembarang *ethernet* dan IP *address*, tidak seperti telepon konvensional yang harus mempunyai port tersendiri di sentral atau PBX (M. Iskandarsyah H, 2005: 1).

Perkembangan teknologi internet yang sangat pesat mendorong ke arah konvergensi dengan teknologi komunikasi lainnya. Standarisasi protokol komunikasi pada teknologi VoIP seperti SIP atau H.323 telah memungkinkan komunikasi terintegrasi dengan jaringan komunikasi lainnya seperti PSTN.

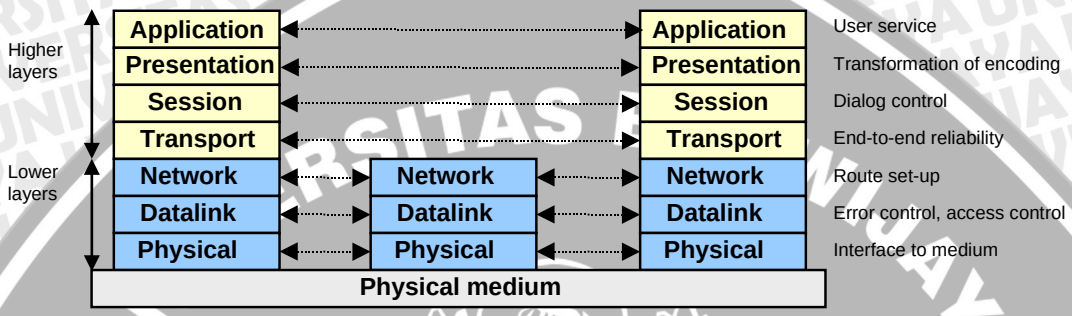


Gambar 2.1 Topologi Umum VoIP
Sumber: M. Iskandarsyah H., 2005 : 1

2.1.2 Model Referensi OSI *(Open System Interconnection)*

Jaringan komunikasi data memerlukan tingkat *compatibility* dan *interoperability* yang tinggi diantara elemen-elemen jaringan, khususnya *interface*

fisik dan logika serta pengendaliannya. Tahun 1977, ISO *(International Organization for Standartdization)* membentuk subkomite yang mengembangkan arsitektur standar bertujuan memperoleh sasaran jangka panjang berupa OSI. Istilah OSI mengacu pada standar pertukaran informasi diantara sistem-sistem ”terbuka”, yaitu sistem yang menerapkan standar OSI dan tidak berpengaruh terhadap implementasi, teknologi, atau interkoneksi sistem, melainkan berhubungan dengan kesesuaian sistem terhadap standar yang ditetapkan.



Gambar 2.2 Model OSI
Sumber : www.prasimax.com, 2002

OSI membagi fungsi komunikasi ke dalam 7 lapis *(layer)*. Setiap *layer* bertugas melaksanakan *subset* fungsi komunikasi tertentu yang diterapkan dalam suatu DTE yang berkomunikasi dengan DTE yang lain. Setiap *layer* mengandalkan *layer* di bawahnya untuk melaksanakan fungsi-fungsi yang lebih primitif sekaligus menyediakan layanan untuk mendukung *layer* di atasnya. *Layer-layer* ini dirancang agar setiap perubahan yang terjadi dalam satu *layer* tidak mempengaruhi *layer* lainnya.

Tujuan OSI adalah memecahkan keberagaman DTE dan DCE. Model referensi OSI juga memperlihatkan susunan protokol *(protocol stack)* yang diterapkan pada dua DTE. Kedua DTE dihubungkan oleh suatu sub-jaringan komunikasi *(communication subnetwork)* yang terletak di antara keduanya (Heywood, 1999 : 30).

a. *Layer 1 : Physical*

Physical layer berkomunikasi langsung dengan media komunikasi dan bertanggung jawab mengirim dan menerima *bit-bit*. Lapisan ini mengetahui spesifikasi mekanik, elektrik, dan prosedural untuk mengaktifkan, memelihara *(maintenance)*, dan memutuskan *(deactivate)* koneksi untuk mentransmisikan deretan *bit* melalui saluran fisik dari media transmisi serta *interface*-nya.

b. *Layer 2 : Data Link*

Dalam jaringan lokal, *data link layer* bertanggung jawab menyediakan komunikasi dari *node* ke *node*. Untuk menyediakan layanan ini, *data link layer* harus melakukan dua fungsi, yaitu harus menyediakan mekanisme alamat yang memungkinkan pesan-pesan dikirimkan ke *node* yang benar, dan *layer* ini menerjemahkan pesan dari *physical layer*. Tanggung jawab utama *data link layer* adalah sebagai berikut:

1. ***Framing,*** membagi *bit stream* yang diterima dari *network layer* menjadi unit-unit data yang disebut *frame.*
2. ***Physical addressing.*** Jika *frame-frame* didistribusikan ke sistem lain pada jaringan, maka *data link* akan menambahkan sebuah *header* di muka *frame* untuk mendefinisikan pengirim atau penerima.
3. ***Flow control.*** Jika *rate* atau laju *bit stream* berlebihan atau berkurang, maka *flow control* akan melakukan tindakan menstabilakan laju *bit.*
4. ***Error control.*** *Data link layer* menambah kehandalan lapisan fisik dengan penambahan mekanisme deteksi dan pentransmisian kembali *frame-frame* yang gagal dikirim.
5. ***Access control.*** Jika dua atau lebih *device* dikoneksi dalam *link* yang sama, *data link layer* perlu menentukan *device* yang mana yang harus dikendalikan pada saat tertentu.

c. *Layer 3 : Network*

Adapun tanggung jawab spesifik *Network Layer :*

1. ***Logical Addressing***

   Bila pada *data link layer* diimplementasikan *physical addressing* untuk menangani pengalamatan/ *addressing* secara local, maka pada *network layer* diimplementasikan *logical addressing* untuk menangani pengalamatan antar jaringan.

2. ***Routing***

   Jaringan yang terhubung membentuk *internetwork* diperlukan metode *routing,* sehingga paket data ditransfer dari *device* pada jaringan tertentu menuju *device* pada jaringan lain.

d. *Layer 4 : Transport*

Transport layer melaksanakan pengendalian *end to end (station-to-station)* terhadap data yang ditransmisikan serta melakukan optimasi penggunaan sumber daya jaringan. *Transport layer* menyediakan layanan *layer-layer* atas berupa proses pembentukan, pemeliharaan *(maintenance)*, dan pemutusan komunikasi data pada jalur transmisi *full duplex*.

Kemampuan *protocol transport* yang diperlukan, ditentukan oleh kualitas layanan *layer-layer* di bawahnya. Jika kualitas layanan *layer* di bawahnya adalah layanan *virtual circuit* yang andal *(reliable)*, serta bebas kesalahan *(error free)*, maka hanya membutuhkan kemampuan *protocol transport* yang minimal.

e. *Layer 5 : Session*

Session layer bertanggung jawab mengendalikan "*dialog*" antar *node (dialog control)*. Suatu "*dialog*" adalah percakapan formal dimana dua *node* sepakat untuk bertukar data. Untuk melakukan suatu "*dialog*", maka dibutuhkan tiga fase, yaitu:

1. **Pembentukan hubungan,** untuk membentuk kontak, *node* menyepakati aturan-aturan komunikasi, *protocol-protocol* yang digunakan dan parameter komunikasinya.
2. **Pemindahan data,** *node-node* dipakai untuk *"dialog"* pertukaran data.
3. **Pemutusan hubungan,** terjadi ketika *node-node* tidak lagi perlu komunikasi.

f. *Layer 6 : Presentation*

Layer presentation bertugas mengurus format data yang dapat dipahami oleh berbagai macam media. *Layer presentation* juga dapat mengkonversi format data, sehingga *layer* berikutnya dapat memahami format yang diperlukan untuk komunikasi. Selain itu *layer presentation* juga berfungsi sebagai enkripsi data. Contoh format data yang didukung oleh *layer presentation* antara lain: *Text*, Data, *Graphic*, *Visual Image*, *Sound*, dan *Video*.

g. *Layer 7 : Application*

Application layer dianalogikan sebagai "jalan" penghubung proses-proses aplikasi yang menggunkan OSI untuk saling mempertukarkan informasi. Semua layanan yang disediakan dapat diakses langsung oleh proses aplikasi, meliputi:

1. Identifikasi *partner* komunikasi yang dituju.
2. Penentuan kesediaan *partner* yang dituju.
3. Pembentukan kewenangan komunikasi.
4. Persetujuan tanggung jawab atas pemulihan kesalahan *(error recovery).*
5. Persetujuan prosedur yang digunakan mempertahankan integritas data.

Dalam lingkungan komunikasi, aliran informasi berasal dari *application layer* pada salah satu ujung, ke *application layer* yang berada pada ujung lainnya.

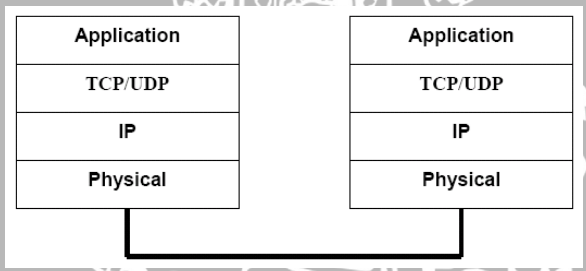
Tabel 2.1 Hubungan Model Referensi OSI dengan Protokol Internet

Model OSI		**TCP/ IP**	***Protocol* TCP/ IP**	
No	**Lapisan**		**Nama *Protocol***	**Kegunaan**
7	Application	Aplikasi	DHCP *(Dynamic Host Configuration Protocol)*	*Protocol* untuk distribusi IP pada jaringan dengan jumlah IP yang terbatas
			DNS *(Domain Name Server)*	*Database* nama domain mesin dan nomer IP
			FTP *(File Transfer Protocol)*	*Protocol* untuk transfer *file*
			HTTP *(Hyper Text Transfer Protocol)*	*Protocol* untuk transfer *file*HTML dan Web
			MIME *(Multipurpose Internet Mail Extention)*	*Protocol* untuk mengirim *file*binary dalam bentuk teks
			NNTP *(Network News Transfer Protocol)*	*Protocol* untuk menerima dan mengirim *newsgroup*
			POP *(Post Office Protocol)*	*Protocol* untuk mengambil mail dari *server*
			SMB *(Server Message Box)*	*Protocol* untuk transfer berbagai *server file*DOS dan *Windows*
6	*Presentation*		SMTP *(Simple Mail Transfer Protocol)*	*Protocol* untuk pertukaran mail
			SNMP *(Simple Network Management Protocol)*	*Protocol* untuk manajemen jaringan
			Telnet	*Protocol* untuk akses jarak jauh
			TFTP *(Trivial* FTP*)*	*Protocol* untuk transfer *file*
5	*Session*		NETBIOS *(Network Basic Input Output System)*	BIOS jaringan standar
			RPC *(Remote Procedure Call)*	Prosedur pemanggilan jarak jauh
			SOCKET	*Input output* untuk *network* jenis BSD-UNIX
4	*Transport*	Transport	TCP *(Transmission Control Protocol)*	*Protocol* pertukaran data berorientasi *(Connection oriented)*
			UDP *(User Datagram Protocol)*	*Protocol* pertukaran data non-orientasi *(Connectionless)*
3	*Network*	Internet	IP *(Internet Protocol)*	*Protocol* untuk menetapkan *routing*
			RIP *(Routing InformationProtocol)*	*Protocol* untuk memilih *routing*

				ARP *(Address Resolution Protocol)*	*Protocol* untuk mendapatkan informasi *hardware*dari nomer IP
				RARP *(Reverse ARP)*	*Protocol* untuk mendapatkan informasi nomer IP dari *hardware*
2	*Data Link*	LLC	*Network interface*	PPP *(Point to Point Protocol)*	*Protocol* untuk *point* ke *point*
		MAC		SLIP *(Serial Line Internet Protocol)*	*Protocol* dengan menggunakan sambungan serial
1	Fisik			*Ethernet*, FDDI, ISDN, ATM	

2.1.3 Protokol-protokol Penunjang Jaringan VoIP

TCP/IP *(Transfer Control Protocol/ Internet Protocol)* merupakan sebuah protokol yang digunakan pada jaringan internet. Protokol ini terdiri dari dua bagian besar, yaitu TCP dan IP. Ilustrasi pemrosesan data untuk dikirimkan dengan menggunakan protokol TCP/IP diberikan pada gambar 2.3.



Gambar 2.3 Mekanisme Protokol TCP/ IP
Sumber: M. Iskandarsyah H., 2005: 3

- **Application Layer**

Fungsi utama lapisan ini adalah pemindahan data dari sebuah sistem ke sistem lainnya yang berbeda dan memerlukan protokol yang sama untuk mengatasi inkompatibilitas. Selain itu lapisan ini berhubungan langsung dengan user. Salah satu contoh aplikasi yang telah dikenal, misalnya HTTP *(Hypertext Transfer Protocol)* untuk web/ situs, FTP *(File Transfer Protocol)* untuk transfer file, dan TELNET untuk koneksi terminal *remote*.

- **TCP (Transmission Control Protocol)**

Dalam transmisi data pada OSI layer *Transport* ada dua protokol yang berperan, yaitu TCP dan UDP. TCP merupakan protokol yang *connection-oriented*, yang artinya menjaga reliabilitas dan konektivitas hubungan komunikasi *end-to-end*.

Konsep dasar kerja TCP adalah mengirim dan menerima segmen-segmen informasi dengan panjang data bervariasi pada suatu datagram internet. TCP menjamin reliabilitas hubungan komunikasi karena melakukan perbaikan terhadap data yang rusak/ hilang. Hal ini dilakukan dengan memberikan nomor urut pada setiap oktet yang dikirimkan dan membutuhkan sinyal jawaban positif dari penerima berupa sinyal ACK *(Acknowledgement)*. Jika sinyal ACK ini tidak diterima pada interval waktu tertentu, maka data akan dikirimkan kembali.

Pada posisi penerima, nomor urut tadi berguna untuk mencegah kesalahan urutan data dan duplikasi data. TCP juga memiliki mekanisme *flow control* dengan cara mencantumkan informasi dalam sinyal ACK mengenai batas jumlah oktet data yang masih boleh ditransmisikan pada setiap segmen yang diterima dengan sukses.

Dalam hubungan VoIP, TCP digunakan pada saat *signalling*. TCP digunakan untuk menjamin setup suatu *call* pada sesi *signalling*. TCP tidak digunakan dalam pengiriman data suara pada VoIP karena pada suatu komunikasi data VoIP penanganan data yang mengalami keterlambatan lebih penting daripada penanganan paket yang hilang.

- **UDP (User Datagram Protocol)**

UDP merupakan salah satu protokol utama di atas IP dan lebih sederhana daripada TCP. UDP digunakan untuk situasi yang tidak mementingkan mekanisme reliabilitas. *Header* UDP hanya berisi empat *field*, yaitu *source port, destination port, length,* dan UDP *checksum*, dimana fungsinya hampir sama dengan TCP, namun fasilitas *checksum* pada UDP bersifat opsional.

UDP pada VoIP digunakan untuk mengirimkan *audio stream* yang dikirimkan secara terus-menerus. UDP digunakan pada VoIP karena pada pengiriman *audio streaming* yang berlangsung terus-menerus lebih mementingkan kecepatan pengiriman data agar tiba ditujuan tanpa memperhatikan adanya paket yang hilang walaupun mencapai 50% dari jumlah paket yang dikirimkan.

Karena UDP mampu mengirimkan data *streaming* dengan cepat, maka dalam teknologi VoIP UDP merupakan salah satu protokol penting yang digunakan sebagai *header* pada pengiriman data selain RTP *(Realtime Transport Protocol)* dan IP. Untuk mengurangi jumlah paket yang hilang saat pengiriman

data (karena tidak ada mekanisme pengiriman ulang), maka pada teknologi VoIP pengiriman data banyak dilakukan pada jaringan *private*.

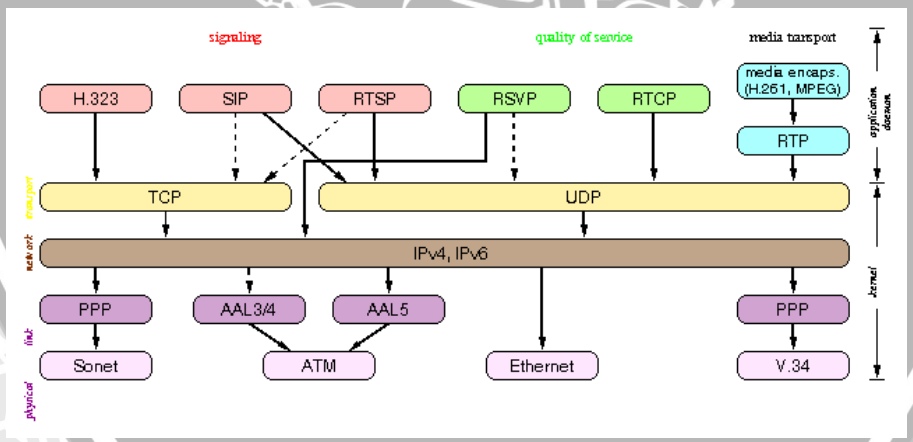
- **IP (Internet Protocol)**

IP didesain untuk interkoneksi sistem komunikasi komputer pada jaringan *packet-switched*. Pada jaringan TCP/ IP, sebuah komputer diidentifikasi dengan alamat IP. Tiap-tiap komputer memiliki alamat IP yang unik, masing-masing berbeda satu sama lainnya. Hal ini dilakukan untuk mencegah kesalahan pada transfer data. Untuk komunikasi data, IP mengimplementasikan dua fungsi dasar, yaitu *addressing* dan *fragmentasi*. Salah satu hal penting dari IP dalam pengiriman informasi adalah metode pengalamatan pengirim dan penerima.

2.1.4 SIP *(Session Initiation Protocol)*

SIP adalah standar IETF *(Internet Engineering Task Force)* yaitu protokol pensinyalan pada *layer* aplikasi yang berbasis ASCII dan berfungsi untuk membangun, memodifikasi, dan mengakhiri suatu sesi multimedia yang melibatkan satu atau beberapa pengguna. Sesi multimedia adalah pertukaran arus data antar pengguna yang meliputi suara, video, atau teks.

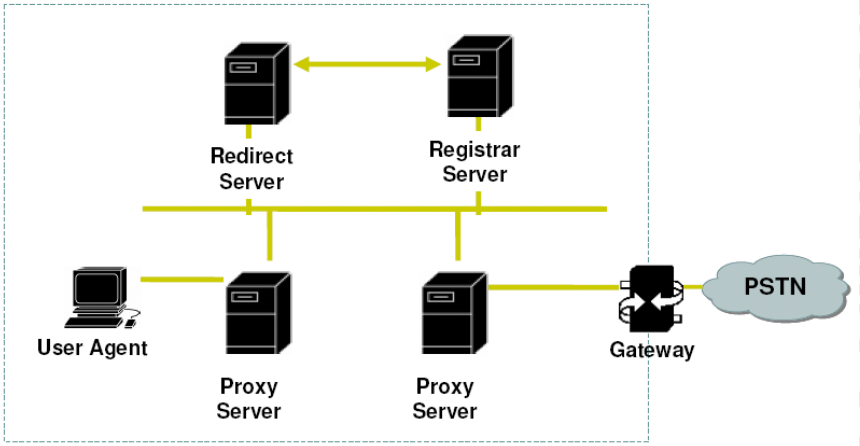
Kedudukan SIP dibandingkan dengan protokol lain seperti terlihat pada gambar 2.4.



Gambar 2.4 H.323 dan SIP
Sumber: Martinus Indra S., 2004: 5

2.1.4.1 Arsitektur SIP

Arsitektur SIP terdistribusi dan *scalable*, selain itu dapat diintegrasikan dengan protokol standar IETF lainnya untuk membuat suatu aplikasi yang berbasis SIP. Pada gambar 2.5 ditunjukkan arsitektur dari SIP.



Gambar 2.5 Arsitektur Sistem Berbasis SIP
Sumber: Anton Raharja, 2006: 14

Keterangan:

- *Proxy-server*

  Menerima *request* dari *user-agent-client*, melakukan autentikasi, memprosesnya, dan mengirimkan *request* tersebut kepada hop selanjutnya atas nama *client* tersebut.

- *Redirect-server*

  Menerima *request* dari *client*, membandingkan alamat tujuan yang ingin dicapai, setelah ditemukan, alamat tersebut dikembalikan kepada *client*.

- *Register-server*

  Menerima REGISTER *request* dari *client*.

- *Location-server*

  Digunakan oleh *proxy/ redirect server* untuk mendapatkan informasi mengenai alamat tujuan yang ingin dicapai.

2.1.4.2 Format Pengalamatan SIP

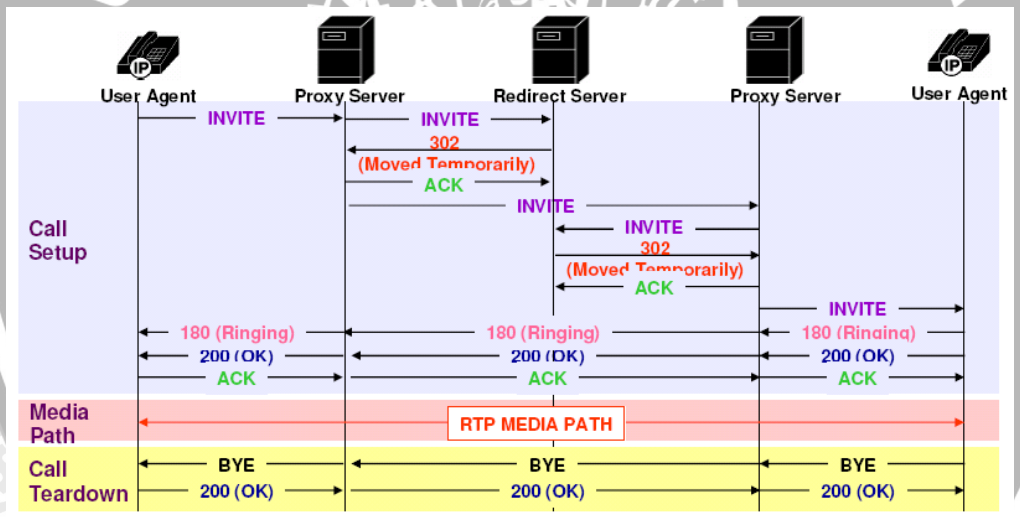
SIP mendefinisikan sebuah SIP URL yang didasarkan pada URL WWW dengan ekstensi-ekstensi yang dapat mengakomodasi bermacam-macam alamat seperti nama, port, URL, web, dan alamat e-mail. Suatu SIP URL merupakan bagian setiap pesan yang menandai asal, tujuan tertentu, dan penerima akhir dalam sebuah permintaan/ *request* SIP.

Format yang umum dalam SIP URL terdiri atas tiga bagian utama, yaitu informasi pengguna, informasi *host port*, dan parameter URI. Informasi pengguna

mengidentifikasi pengguna dalam *request* SIP dan sebuah pilihan *password* dapat berhubungan dengan bagian informasi pengguna. Bagian ini dapat berupa *username*, nomor telepon, atau kombinasi keduanya. *Host port* mengidentifikasi suatu nama *host*, dan *port* yang berhubungan dengan *host*. Bagian ini dapat berupa nama *host* yang sederhana, suatu nama *domain*, atau kombinasi antara keduanya. Parameter URI *(Universal Request Identifier)* ini memberikan fleksibilitas yang besar untuk mengidentifikasi parameter-parameter tertentu. Bagian ini terdiri atas parameter *layer network* dan transport yang terdiri atas TCP, UDP, dan SCTP.

2.1.4.3 *Messages* pada SIP

Hubungan yang dibangun oleh SIP pada proses *signalling* bersifat *client-server*. Dengan demikian ada dua jenis *message*, yaitu *request* dan *reponse*. Langkah-langkah yang lebih lengkap dapat dilihat pada tabel 2.2 dan 2.3, sedangkan gambaran singkat penggunaan *message* tersebut dapat dilihat pada gambar 2.6.



Gambar 2.6 Contoh Sesi Komunikasi pada SIP
Sumber: Anton Raharja, 2006: 16

Tabel 2.2 SIP *Request Message*

SIP Request Message	Deskripsi
INVITE	Mengidentifikasikan bahwa *user* atau *service* "diundang"*(invited)* untuk bergabung dalam *session*
ACK	Konfirmasi bahwa *client* telah menerima respon final terhadap *request* "INVITE"
BYE	Mengidentifikasikan bahwa *user* ingin mengakhiri panggilan/ *session*
CANCEL	Membatalkan *request* yang tertunda namun tidak mempengaruhi *request* yang telah selesai/ direspon
REGISTER	Mendaftarkan alamat yang ada pada *field header* "To" dengan *server* SIP
OPTIONS	Meminta informasi kapabilitas dari *server*
INFO	Memungkinkan tersalurkannya informasi *control* yang berkaitan dengan *session*, yang dihasilkan selama *session*.

Sumber: Anton Raharja, 2006: 7

Tabel 2.3 SIP *Response Message*

Jenis SIP Response Message	Deskripsi
1xx	Respon Informasi Contoh : 180 Ringing
2xx	Respon Sukses/ Berhasil Contoh : 200 OK
3xx	Respon *Redirection* Contoh : 302 *Moved Temporarily*
4xx	Respon kegagalan *request* Contoh : 403 *Forbidden*
5xx	Respon kegagalan *server* Contoh : 504 *Gateway Time-out*
6xx	Respon kegagalan global Contoh : 600 *Busy Everywhere*

Sumber: Anton Raharja, 2006: 7

2.1.4.4 Komunikasi dengan SIP

Pembangunan suatu komunikasi multimedia dengan SIP dilakukan melalui beberapa tahap:

- *User location*

  menentukan lokasi pengguna yang akan berkomunikasi
- *User availability*

  menentukan tingkat keinginan pihak yang dipanggil untuk terlibat dalam komunikasi
- *User capability*

menentukan media maupun parameter yang berhubungan dengan media yang akan digunakan untuk komunikasi

- *Session setup*

  "ringing", pembentukan hubungan antara pihak pemanggil dan pihak yang dipanggil

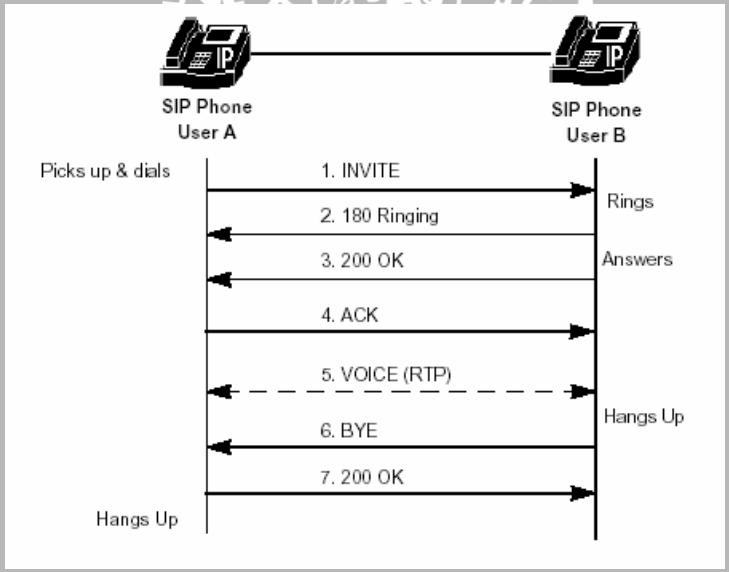
- *Session management*

  meliputi *transfer*, modifikasi, dan pemutusan sesi

Komunikasi pada SIP dilakukan dengan mengirimkan *message* yang berbasis HTTP. Setiap pengguna mempunyai alamat yang dinyatakan dengan SIP URI.

2.1.4.5 Operasi Dasar SIP

Pada operasi dasar VoIP dapat diilustrasikan seperti terlihat pada gambar 2.6, sebagai contoh User A menggunakan aplikasi SIP pada PC *(softphone)* untuk memanggil User B (juga menggunakan *softphone*) melalui internet.



Gambar 2.7 Diagram *Basic Call Flow*
Sumber: Martinus Indra S., 2004: 7

Dari ilustrasi gambar 2.7 dapat diuraikan dalam langkah-langkah seperti ditunjukkan pada tabel 2.4.

Tabel 2.4 *Call Flow Detail*

No.	Deskripsi
1	**INVITE** : User A memulai panggilan menuju User B
2	**180 Ringing** : User B mengirim sinyal dering menuju User A
3	**200 OK** : User B merespon
4	**ACK** : User A menjawab bahwa sudah menerima pesan ”200”
5	**VOICE** : Sebuah kanal 2 arah terbentuk melalui RTP dan percakapan terjadi antara A dan B
6	**BYE** : User B memutuskan koneksi
7	**200 OK** : Panggilan teputus dan User A menutup koneksi

Sumber: Martinus Indra S., 2004: 7

Proses **INVITE** dapat dituliskan sebagai berikut:

```
INVITE sip:udin@jakarta.com SIP/2.0
Via: SIP/2.0/UDP bandung.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Udin <sip:udin@jakarta.com>
From: Martin <sip:martin@bandung.com>;tag=1928301774
Call-ID: a84b4c76e66710@bandung.com
CSeq: 314159 INVITE
Contact: <sip:martin@bandung.com>
Content-Type: application/sdp
Content-Length: 142
```

Keterangan:

- Baris pertama merupakan *start-line* yang menunjukkan awal sebuah pesan. Pada *request*, *start-line* disebut *request-line*, yang terdiri dari <metode> <request-URI> <SIP-version>. Pada contoh di atas, metode dari *request* adalah INVITE, URI ditujukan ke sip:udin@jakarta.com, dan SIP yang digunakan adalah versi 2.0.
- Via berisi alamat *proxy* dimana *client* akan menerima *response*. Pada contoh di atas alamat *proxy*-nya adalah bandung.com. Via juga dapat berisi parameter *branch* yang menunjukkan identitas suatu *transaction* pada *proxy*.
- Max-Forwards menunjukkan jumlah hop maksimum yang dapat dilalui oleh pesan sebelum mencapai tujuan. Angka pada *max-forwards* di-*decrement* setiap bertemu hop.
- To berisi alamat dari tujuan. Alamat ini dapat dituliskan dalam bentuk *display name* (Udin), atau dalam bentuk SIP URI (sip: udin@jakarta.com).
- From berisi alamat pengirim yang dapat dituliskan dalam bentuk *display name* (Martin), atau dalam bentuk SIP URI (sip:martin@bandung.com). *Header* ini juga mempunyai parameter *tag* yang menunjukkan identitas *softphone client* yang digunakan.

- Call-ID berisi kombinasi karakter yang menunjukkan identitas dari suatu *call* yang dimulai pada waktu tertentu (di-*generate* berdasarkan *random string* dan *ip-address/ hostname* dari *softphone*). Kombinasi dari Call-ID, To-tag, dan From-tag menunjukkan suatu identitas dari hubungan *peer-to-peer* yang disebut dialog.
- CSeq *(Command Sequence)* berisi angka *integer* dan nama metode. Angka *integer* pada CSeq di-*increment* setiap ada *request* dalam suatu dialog.
- Content-Type menunjukkan deskripsi dari *message-body*.
- Content-Lenght menunjukkan panjang dari *message-body (byte)*.

Contoh **RESPONSE** dapat dituliskan sebagai berikut:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP jakarta.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bogor.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP bandung.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Udin <sip:udin@jakarta.com>;tag=a6c85cf
From: Martin <sip:martin@bandung.com>;tag=1928301774
Call-ID: a84b4c76e66710@bandung.com
CSeq: 314159 INVITE
Contact: <sip:udin@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

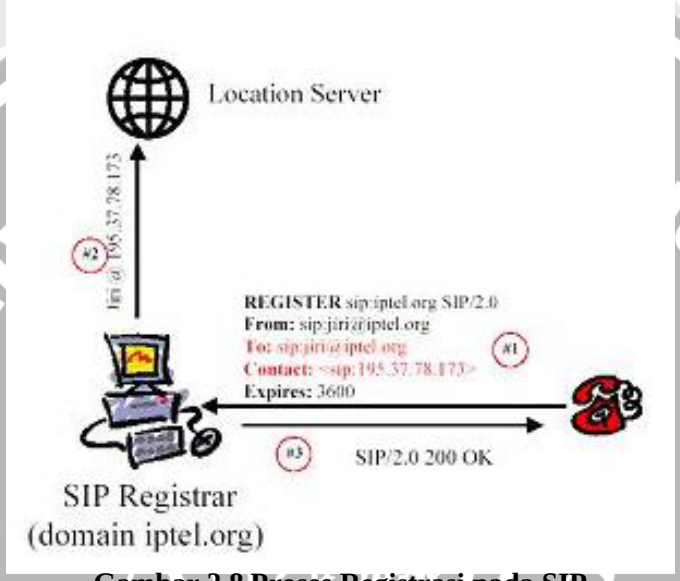
Keterangan :

- Baris pertama *(start-line)* pada *response message* disebut *status-line*. *Status-line* ini berisi <SIP-version=SIP/2.0><status-code=200><statusphrase=OK>.
- Via, To, From, Call-ID, dan CSeq di-*copy* dari INVITE *request*. Pada Via *header-field* sudah ditambahkan *header* tambahan yang berisi alamat tiap *proxy* yang dilewati. *softphone* Udin menambahkan parameter *tag* pada *header* To.
- Contact berisi alamat URI dari Udin yang berfungsi untuk mempermudah pertukaran *message* di waktu yang akan datang.

2.1.4.6 Registrasi pada SIP

Salah satu keistimewaan SIP adalah adanya mobilitas yang tinggi bagi pengguna. Untuk mengetahui keberadaan seseorang pada saat tertentu, suatu SIP *proxy-server* harus mendapatkan informasi dimana seseorang berada. Proses ini

disebut registrasi. Seorang pengguna mula-mula harus melakukan proses registrasi agar *proxy* terdekat mengetahui keberadaannya, sehingga *proxy* tersebut dapat menerima atau mengirimkan pesan kepada pengguna. Proses registrasi dilakukan dengan cara mengirimkan *request* yang disebut REGISTER kepada *proxy-server*. *Proxy-server* ini disebut *registrar*. Sebuah *registrar* bertugas menerima dan menyimpan data yang berisi alamat pengguna. Registrar dan *proxy-server* biasanya merupakan satu kesatuan.



Gambar 2.8 Proses Registrasi pada SIP
Sumber: Martinus Indra S., 2004: 12

Gambar 2.8 merupakan suatu contoh proses registrasi pada SIP. Suatu REGISTER *request* mempunyai *header-field* sebagai berikut:

- Request URI berisi nama domain dari *registrar*. *Header* ini tidak boleh mengandung karakter "@" dan nama pengguna. Pada contoh di atas sip:iptel.org.
- To berisi alamat seseorang yang akan disimpan pada *registrar*. Pasa contoh ini sip:jiri@iptel.org.
- From berisi alamat seseorang yang melakukan proses registrasi. Biasanya sama dengan alamat pada TO, kecuali registrasi dilakukan oleh pihak ketiga.
- Contact biasanya berisi SIP-URI yang menunjukkan suatu SIP *endpoint*. Pada contoh ini sip:195.37.78.173.

2.1.5 Protokol Transfer Data

2.1.5.1 RTP *(Real-Time Transport Protocol)*

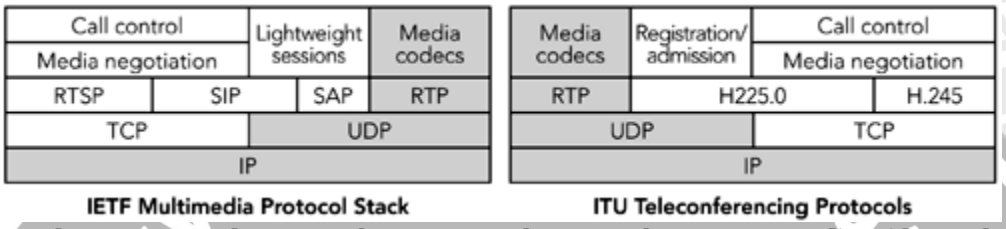
Metode standar untuk proses transmisi *audio*/*video* dalam jaringan berbasis IP saat ini adalah RTP. Menggunakan internet sebagai media untuk menghantarkan *content audio* dan *video* bukanlah hal baru. RTP bertujuan untuk menyediakan servis yang berguna untuk pertukaran media yang *real-time* seperti *audio* dan *video*, di atas jaringan IP. Servis tersebut meliputi perbaikan waktu, deteksi kesalahan dan koreksi, pengidentifikasian *payload* dan *source*, *reception quality feedback*, sinkronisasi media, dan menajemen keanggotaan. RTP asalnya didesain untuk digunakan pada konferensi *multicast*, menggunakan *session model* yang ringan. Sejak saat itu, RTP terbukti berguna untuk sebuah *range* aplikasi yang lain seperti pada H.323, *webcasting*, dan distribusi TV, baik pada media kabel dan telepon selular. Protokol ini telah diterapkan mulai dari skala *point-to-point* hingga skala *multicast session* dengan ribuan pengguna, serta diterapkan pada jalur telepon selular dengan *bandwidth* kecil hingga aplikasi yang mengantarkan *HDTV* dengan *bandwidth* yang ber-orde *gigabit*.

RTP dikembangkan oleh *Audio*/*Video Transport* working *group* oleh IETF dan telah diadopsi oleh ITU sebagai bagian dari seri rekomendasi dari H.323, serta badan standar internasional lainnya. Disamping RTP, sebuah sistem yang lengkap biasanya membutuhkan protokol yang lain dan standar sesi pengumuman, inisiasi, dan kontrol; kompresi media dan *network transport*. Gambar 2.9 menunjukkan format *header* dari paket RTP.

+ Bits	0-1	2	3	4-7	8	9-15	16-31
0	Ver.	P	X	CC	M	PT	Sequence Number
32	Timestamp						
64	SSRC identifier						
96	... CSRC identifiers ...						
96+(CC×32)	Extension header (optional).						
96+(CC×32) + (X×((EHL+1)×32))	Data						

Gambar 2.9 RTP *Sender*
Sumber : Collins Perkins, 2003

Gambar 2.10 menunjukkan negosiasi dan protokol *call control*, media *transport layer* (disediakan oleh RTP), algoritma kompresi dan dekompresi *(codec)*, serta dasar jaringan keduanya sangat berhubungan, baik yang menurut *framework* dari ITU ataupun IETF. Kedua set dari *call control* yang paralel dan standar negosiasi media menggunakan *framework* dari media *transport* yang sama. Media *codec* umumnya tidak mempermasalahkan bagaimana *session* bernegosiasi, dan terlepas dari dasar *transport* jaringan.

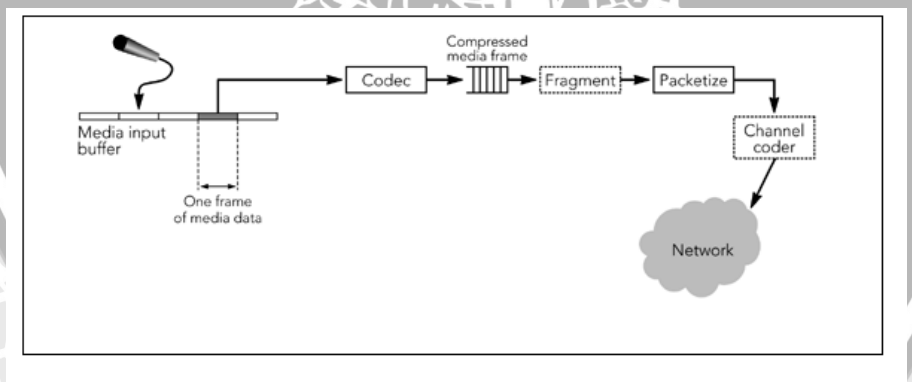


Gambar 2.10 Standar IETF dan ITU untuk *Protocol Transport Audio/Video* pada Jaringan

Sumber: Collins Perkins, 2003

- **Aktifitas RTP pada pengirim**

Pengirim bertanggung jawab untuk menangkap dan mengubah *audiovisual* menjadi data untuk dikirimkan, sebaik dalam membuat paket RTP. RTP membolehkan juga berpartisipasi pada koreksi kesalahan dan pengendalian kongesti dengan mengadaptasi media *stream* yang dikirim untuk merespon *receiver feedback.*



Gambar 2.11 Diagram RTP *Sender*

Sumber: Collins Perkins 2003

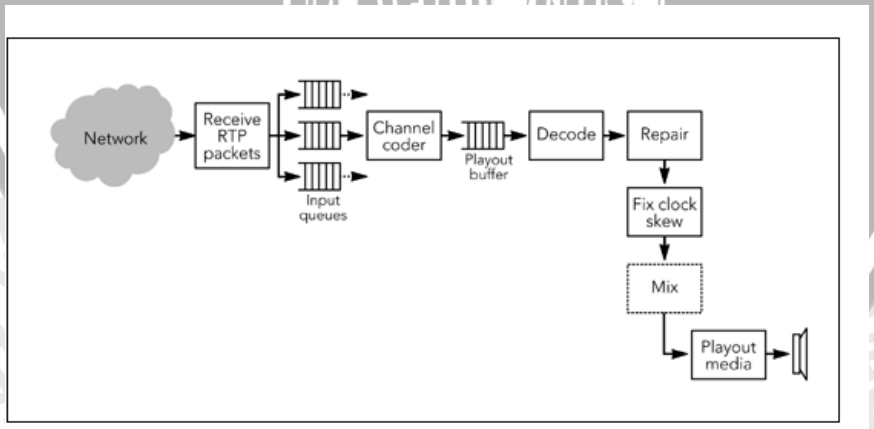
Media data yang tak terkompres *audio*/*video* ditangkap ke dalam *buffer*, dari *frame* dikompres yang diproduksi. *Frame* mungkin dienkodekan pada beberapa jalan tergantung dari algoritma pemrosesan yang digunakan, dan *frame* yang terenkode boleh digantungkan pada awal atau akhir data.

Frame yang terkompres dimasukkan dalam paket RTP, siap untuk dikirimkan. Jika *frame* besar, mungkin akan difragmentasi ke dalam beberapa paket. Jika terlalu kecil beberapa *frame* akan di satukan menjadi paket RTP tunggal. Tergantung pada prediksi koreksi kesalahan, sebuah kanal *coder* mungkin digunakan untuk menghasilkan paket *error correction* atau meminta paket sebelum dikirim. Setelah paket RTP dikirim, *buffer media* mencocokkan paket tersebut dan akhirnya dibebaskan. Pengirim tidak boleh menghapus data yang mungkin diperlukan untuk koreksi kesalahan atau untuk proses *encoding*. Keperluan ini mungkin berarti bahwa pengirim harus mem-*buffer* data untuk beberapa waktu setelah itu mencocokan paket yang telah dikirm, tergantung pada *codec* dan prediksi kesalahan yang digunakan.

Pengirim bertanggung jawab untuk menghasilkan status *report* secara periodik untuk media *stream* yang dihasilkan, termasuk yang dibutuhkan untuk sinkronisasi. Dan juga menerima *reception quality feedback*. Dari partisipan yang lain dan mungkin menggunakan informasi untuk beradaptasi pada pengiriman.

- **Aktifitas RTP pada penerima**

Sebuah penerima bertanggung jawab untuk mengumpulkan paket RTP dari jaringan, memeriksa semua kesalahan, memperbaiki waktu, mendekompres media dan menampilkannya pada *user*. Dan juga mengirimkan *reception quality feedback*, mengijinkan pengirim beradaptasi saat pengiriman ke penerima, dan menjaga *database* partisipan pada session.



Gambar 2.12 Diagram RTP *Receiver*
Sumber: Collins Perkins 2003

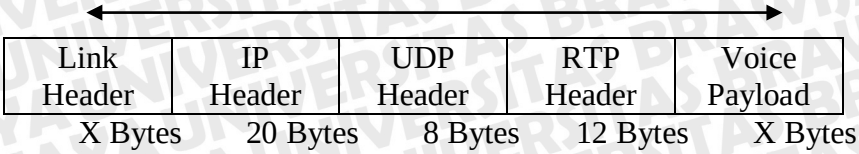
Langkah pertama dari proses penerimaan adalah mengumpulkan paket dari jaringan, memvalidasi untuk dikoreksi, dan memasukkannya kedalam input yang

spesifik. Paket dikumpulkan dari *input queue* dan dilewatkan ke *channel-coding* untuk dikoreksi kesalahannya. Mengikuti *channel coder*, paket dimasukkan ke dalam *playout buffer*. *Playout coder* diatur berdasarkan *timestamp*, dan proses pemasukan paket kedalam *buffer* membetulkan setiap permintaan yang terhalangi selama *transport*. Paket masih berada di layout *buffer* sampai *frame* telah selesai diterima, dan menambahkan *buffer* untuk menghilangkan setiap variasi pada *timing* interpaket yang disebabkan oleh jaringan. Perhitungan sejumlah *delay* untuk menambah salah satu aspek yang kritis pada desain dari implementasi RTP. Masing-masing paket dilabeli dengan waktu playout yang diinginkan untuk *frame* yang cocok.

Setelah waktu *playout* tercapai, paket dikumpulkan untuk dibentuk *frame* yang sempurna, dan setiap kerusakan atau kehilangan paket diperbaiki. Mengikuti setiap perbaikan yang perlu, *frame* dikodekan (tergantung dari *codec* yang digunakan, dan mungkin media perlu dikodekan sebelum paket yang hilang dapat diperbaiki). Pada poin ini mungkin ada perbedaan yang dapat di observasi pada *clock-rate* nominalnya pada pengirim maupun penerima. Beberapa perbedaan sebagai perwujudan sendiri sebagai penyimpangan pada harga dari *clock media* RTP relatif terhadap *playout clock*-nya. Penerima harus mengkompensasi untuk *clock* ini untuk menghindari gap pada playout.

Akhirnya, media data dimainkan user. Tergantung pada format dari media tersebut dan *output device*, hal itu memungkinkan untuk memainkan setiap *stream* sendiri-sendiri sebagai contoh, menampilkan beberapa *video stream*, pada masing-masing *window*. Alternatifnya, mungkin perlu untuk mencampur media dari beberapa sumber jadi satu *stream* tunggal untuk memainkan via sebuah satu set speaker.

RTP merupakan protokol yang dibuat untuk mengkompensasi *jitter* dan *de-sequencing* yang terjadi pada jaringan IP. RTP dapat digunakan untuk beberapa macam data *stream* yang *real-time* seperti data suara dan data video.



Gambar 2.13 Format Paket VoIP

Sumber : www.cisco.com.

Tiap paket VoIP terdiri atas dua bagian, yaitu *header* dan *payload* (beban). *Header* terdiri atas IP Header, RTP Header, UDP Header, dan Link Header. IP *Header* berfungsi menyimpan informasi *routing* untuk mengirimkan paket-paket ke tujuan. Pada tiap header IP disertakan tipe layanan atau *Type of Service* (ToS) yang memungkinkan paket VoIP mendapat perlakuan berbeda dibanding dengan paket yang tidak *realtime*.

UDP Header menunjukkan identitas dari paket VoIP, yaitu *connection less* artinya tidak ada jaminan paket sampai pada tujuan dan oleh karena itu UDP digunakan pada aplikasi *realtime* yang sangat peka terhadap *delay* dan *latency*.

RTP Header berfungsi untuk melakukan *framing* dan segmentasi data *realtime*. RTP berjalan diatas protokol UDP sehingga tidak menjamin paket sampai tujuan, akan tetapi RTP menggunakan RTCP *(Real-Time Transport Control Protocol)* untuk melakukan pemantauan kualitas dan distribusi data.

Besarnya *Link Header* tergantung dari media yang digunakan, berikut tabel *link header* untuk media yang berbeda.

Tabel 2.5 *Link Header*

Media	Link Header Size
Ethernet	14 Byte
PPP	6 Byte
Frame Relay	4 Byte
ATM	5 Byte tiap cell

Sumber : www.cisco.com

Voice Payload berisi informasi suara yang sudah melalui proses coding, berikut tabel *voice payload* untuk beberapa teknik coding.

Tabel 2.6 *Voice Payload*

Teknik Kompresi/ Coding	Ukuran *Payload* (byte)
G.711 (64kbps)	240
G.711 (64kbps)	160 (default)
G.723.1 (6.3kbps)	48
G.723.1 (6.3kbps)	24 (default)
G.723.1 (5.3kbps)	40
G.723.1 (5.3kbps)	20

Sumber : www.cisco.com

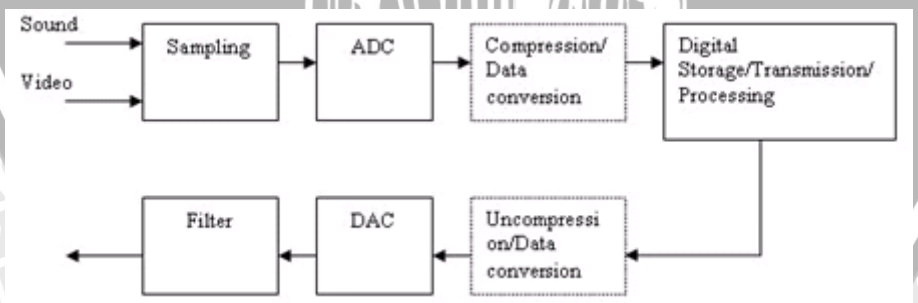
2.1.5.2 RTCP *(Real-Time Transport Control Protocol)*

Protokol RTCP merupakan protokol yang mengendalikan transfer media. Protokol ini biasanya digunakan bersama-sama dengan protocol RTP. RTCP digunakan untuk mengirimkan paket kontrol setiap terminal yang berpartisipasi pada percakapan yang digunakan sebagai informasi untuk kualitas transmisi pada jaringan.

Terdapat dua komponen penting pada paket RTCP, yang pertama adalah *sender report* yang berisikan informasi banyaknya data yang dikirimkan, pengecekan timestamp pada *header* RTP dan memastikan bahwa data yang dikirim tepat dengan *timestamp*-nya. Komponen yang kedua adalah *receiver report* yang dikirimkan oleh penerima panggilan. *Receiver report* berisi informasi mengenai jumlah paket yang hilang selama sesi percakapan, menampilkan *timestamp* terakhir dan *delay* sejak pengiriman *sender report* yang terakhir.

2.1.6 Perubahan Sinyal Analog menjadi Sinyal Digital

Aplikasi *multimedia streaming* membutuhkan *bandwidth* yang besar. Agar aplikasi *multimedia* dapat ditransmisikan pada sistem transmisi yang mempunyai kecepatan transmisi yang terbatas maka diperlukan kompresi pada data *multimedia* tersebut. Kompresi data hanya dapat dilakukan pada data digital. Setelah sinyal *audio* maupun *video* mengalami konversi dari data analog menjadi digital, data *audio* dan *video* dapat dikompresi menjadi data digital yang lebih kecil.



Gambar 2.14. Proses Konversi dan Kompresi Data
Sumber : W. Buchanan, 1997 : 11

Metode yang umum dipakai untuk konversi sinyal suara dan *audio* analog menjadi data digital adalah Pulse Code Modulation (PCM). PCM terdiri atas tiga proses yaitu :

1. *Sampling* (pencuplikan)

Sinyal analog di-*sampling* dengan frekuensi *sampling* tertentu yang mengacu pada teorema Nyquist seperti yang terlihat dalam persamaan berikut.

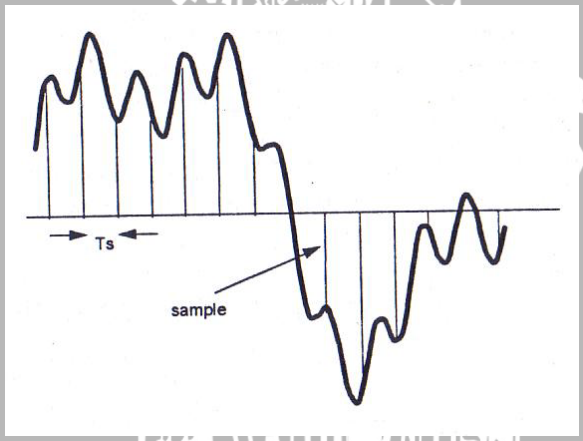
$$fs = 2xf(t)_{maks}$$

Dengan :

fs = frekuensi *sampling* (Hz)

$f(t)_{maks}$ = frekuensi maksimum sinyal analog (Hz)

Untuk suara, frekuensi maksimumnya 4 kHz sehingga harus di-*sampling* dengan frekuensi *sampling* 8 kHz. Untuk *audio* dengan kualitas Hi-Fi, frekuensi maksimumnya 20 kHz sehingga harus di-*sampling* dengan frekuensi *sampling* 40 kHz (untuk professional Hi-Fi menggunakan 44,1 kHz). Hasil dari proses *sampling* ini berupa *Pulse Amplitude Modulation* (PAM) dimana level amplitudonya masih berupa analog, seperti yang terlihat dalam Gambar 2.15.



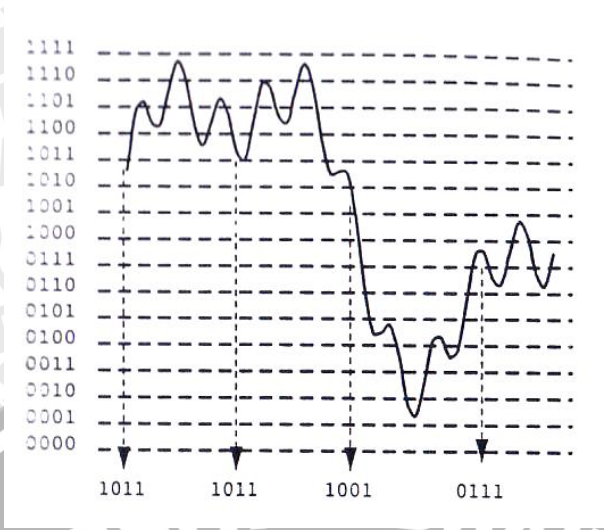
Gambar 2.15 Proses *Sampling*
Sumber : W. Buchanan, 1997 : 12

2. *Quantizing* (kuantisasi)

Quantizing merupakan proses mengubah level amplitudo analog hasil *sampling* menjadi level amplitudo diskrit dengan level kuantisasi tertentu.

3. *Coding* (pengkodean)

Coding merupakan proses mengubah level amplitudo diskrit hasil kuantisasi menjadi data digital yang mengacu pada level kuantisasi. Jika level kuantisasinya 16 = 24 maka setiap *sample* akan dikodekan dengan 4 bit, seperti yang terlihat dalam Gambar 2.16.



Gambar 2.16 Proses *Quantizing* dan Coding
Sumber : W. Buchanan, 1997 : 13

Data sering kali mengalami redundansi dalam penyimpanan datanya. Kompresi ditujukan untuk mengurangi kuantitas data dengan tetap mempertahankan kualitas hasil rekonstruksinya. Proses pengurangan kuantitas data tersebut dilakukan dengan membuang informasi-informasi yang tidak berguna sehingga yang dikirimkan atau disimpan hanya informasi yang penting saja.

2.1.7 Pengkodean Suara

Teknik PCM *(Pulse Code Modulation)* banyak digunakan dalam jaringan PSTN untuk merubah sinyal analog ke bentuk digital. PCM melakukan sampling sinyal analog yang berupa sinyal suara dengan *rate* 8000 sample/ detik (asumsi menggunakan prinsip Nyquist fs = 2 x fi, fi = 4000Hz) dan tiap sample direpresentasikan menjadi satu kode 8 bit. Sehingga *bandwidth* yang dibutuhkan dalam saluran untuk setiap percakapan adalah 8000 x 8 bit menjadi 64 Kbps.

ITU *(International Telecommunication Union)* mengeluarkan beberapa standar untuk *voice coding* yang membutuhkan *bandwidth* yang lebih kecil. Tabel 2.5 menunjukkan perbandingan beberapa *voice coding* serta konsumsi *bandwidth (Bitrate)* dan ukuran *frame* atau biasa disebut juga proses *delay*. Peran *voice coding* dalam aplikasi VoIP adalah bahwa *voice coding* berada di sisi *client* dan sifatnya permanen atau tidak bisa diubah-ubah baik *bitrate* maupun *frame size*-nya. Oleh karena itu pemilihan standar *voice coding* akan berpengaruh pada

penundaan saat melakukan transmisi data *(end-to-end delay)*, tetapi tidak pada kualitas suara.

Tabel 2.7 *Codec* Standar ITU

Standard	Algorithm	Bit Rate (Kbit/s)	Typical end-to-end delay (ms) (excluding channel delay)	Result;ant Voice Quality
G.711	PCM	48, 56, 64	<<1	Excellent
G.723.1	MPE/ACELP	5.3, 6.3	67-97	Good(6.3), Fair(5.3)
H.728	LD-CELP	16	<<2	Good
G.729	CS-ACELP	8	25-35	Good
G.729 annex A	CS-ACELP	8	25-35	Good
G.722	Sub-band ADPCM	48, 56, 64	<<2	Good
G.726	ADPCM	16,24,32,40	60	Good(40), Fair(24)
G.727	AEDPCM	16, 24, 32, 40	60	Good(40), Fair (24)

Sumber : www.cisco.com

2.2 VPN *(Virtual Private Network)*

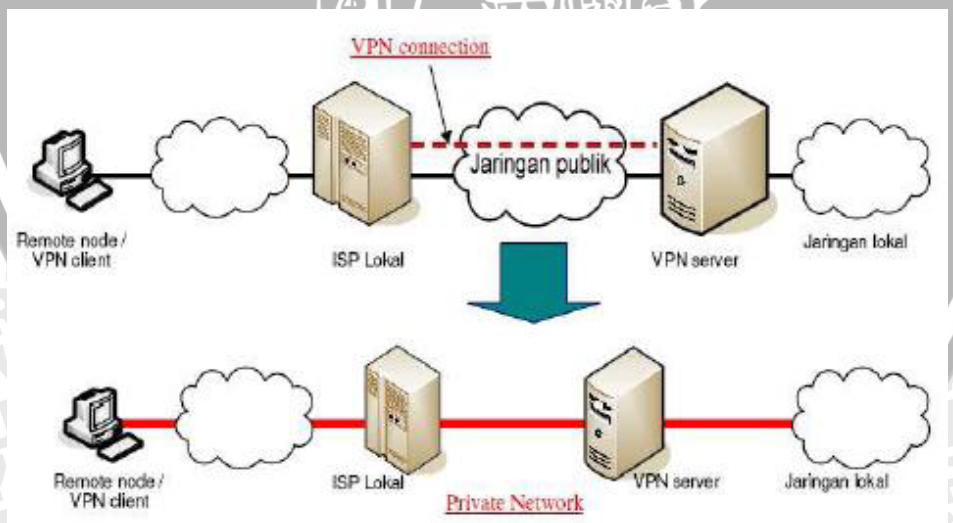
2.2.1 Konsep Dasar VPN

VPN merupakan suatu cara untuk membuat sebuah jaringan yang bersifat *private* dan aman dengan menggunakan jaringan publik misalnya internet. Sebuah jaringan *private* haruslah berada dalam kondisi VIP dan *top secret*. Masalah keamanan data, ketertutupan transfer data dari akses ilegal yang tidak diharapkan serta skalabilitas jaringan menjadi standar utama sebuah *private network*. Pembangunan *private network* secara fisik, akan lebih mahal daripada pembangunan sebuah VPN karena banyaknya perubahan atau penambahan jalur-jalur fisik baru pada sebuah *private network.*

Untuk mengatasi faktor mahalnya pembangunan sebuah *private network* secara fisik, maka dirintislah sebuah teknik baru, yang kemudian dikenal sebagai VPN. VPN menyediakan konektivitas *network* pada jarak jauh yang secara fisik masih memungkinkan. Keuntungan dari fitur VPN, yaitu kemampuan untuk menggunakan jaringan publik (internet) dibandingkan ketergantungan ke *private leased line (cost advantage)*. Teknologi VPN mengimplementasikan batasan akses *network* dengan menggunakan pengkabelan dan sistem *router* yang sama seperti jaringan publik tanpa mengorbankan fitur-fitur dasar keamanan.

Dalam dunia *IT networking*, istilah *virtual* yang berarti tidak memiliki wujud atau phisik yang sebenarnya di antara *link* di kedua jaringan tersebut, tetapi menggunakan jaringan yang telah ada dan juga digunakan secara bersama. Demikian pula dengan istilah *private* yang berarti pribadi dalam membentuk suatu koneksi di antara dua jaringan atau lebih. Pada teknologi jaringan komputer tradisional untuk menghubungkan suatu jaringan komputer dengan jaringan komputer lain yang berbeda tempatnya akan digunakan *dedicated-line* atau *leased-line*, dimana akses seseorang akan menjadi sangat terbatas dan tidak dapat melakukan pekerjaan secara remote.

Teknologi ini memang tidak mendukung seseorang yang memiliki pekerjaan secara *mobile* yang memerlukan akses data dimana saja dan kapan saja, sehingga solusi yang dapat diberikan disini adalah dengan menyediakan modem sebagai fasilitas *dial-up*. Sedangkan pada teknologi VPN yang secara umum merupakan suatu teknologi yang memungkinkan seseorang terkoneksi ke jaringan lokal melalui jaringan komputer publik dan membentuk suatu jaringan pribadi, sehingga dapat dimanfaatkan untuk mendapatkan hak dan pengaturan yang sama seperti ketika berada di kantor. Prinsip kerja layanan VPN seperti yang diilustrasikan pada gambar 2.17.



Gambar 2.17 Konfigurasi VPN
Sumber: Markus Feilner, 2006: 11

VPN dapat mengirimkan data antara dua komputer yang melewati jaringan publik sehingga seolah-olah terhubung secara *point to point* sehingga data dapat

melewati jaringan publik dan dapat mencapai akhir tujuan. Beberapa contoh layanan-layanan yang dapat didukung oleh VPN, antara lain *secure remote access client connections, LAN-to-LAN internetworking,* dan akses terkontrol dalam intranet/ internet.

2.2.2 Komponen VPN

Untuk membangun sebuah jaringan VPN *user* membutuhkan beberapa komponen yang berkaitan dengan jaringan tersebut. Komponen yang dibutuhkan dalam membangun jaringan VPN, antara lain :

a. VPN *Server*

VPN *Server* merupakan komponen utama yang harus ada dalam jaringan VPN. VPN *Server* merupakan sebuah komputer yang akan menerima koneksi VPN *Client.*

b. VPN *Client*

VPN *Client* merupakan *device* yang akan melakukan koneksi dengan VPN *Server.*

c. *Tunnel*

Tunnel merupakan cara dimana data yang ditransferkan (dari VPN *Server* ke *Client* atau sebaliknya) dibungkus dalam sebuah kapsul yang diamankan dengan pengkapsulan, dan kemudian dilakukan enkripsi.

d. VPN *Connection*

Merupakan sebuah koneksi antara VPN *Client* dan VPN *Server* dimana data yang digunakan harus dibungkus dan dienkripsi terlebih dahulu.

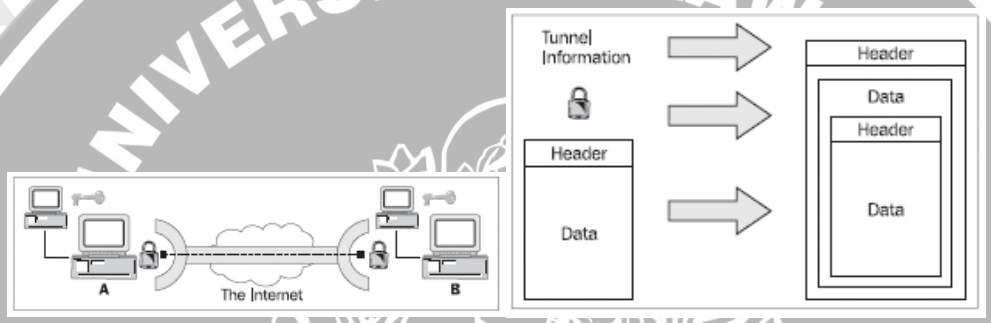
2.2.3 Proses *Tunneling* pada VPN

VPN hanya membuat *tunneling* (terowongan) yang aman antara dua buah *node* atau lebih melewati jaringan terbuka (internet). VPN dapat menghubungkan dua *network* yang kecil (maupun yang lebih besar) untuk membangun sebuah jaringan tunggal yang sebelumnya terpisah baik secara fisik maupun logika.

Tunneling merupakan metode untuk transfer data dari satu jaringan ke jaringan lain dengan memanfaatkan jaringan internet secara terselubung. Disebut *tunnel* atau saluran karena aplikasi yang memanfaatkan yang melihat dua *end point* atau ujung, sehingga paket yang lewat pada *tunnel* hanya akan melakukan

satu kali hop. Data yang akan ditransfer dapat berupa *frame* atau paket dari protokol lain.

Protokol *tunneling* tidak mengirimkan *frame* sebagaimana yang dihasilkan oleh *node* asalnya begitu saja, melainkan membungkusnya(mengenkapsulasi) dalam *header* tambahan. *Header* tambahan tersebut berisi informasi *routing* sehingga data *(frame)* yang dikirim dapat melewati jaringan internet. Jalur yang dilewati data dalam internet disebut *tunnel*. Saat data tiba pada jaringan tujuan, proses yang terjadi selanjutnya adalah dekapsulasi, kemudian data asli akan dikirim ke penerima terakhir. *Tunneling* mencakup secara keseluruhan mulai dari enkapsulasi, transmisi, dan dekapsulasi.



Gambar 2.18 *Tunneling*
Sumber: Markus Feilner, 2006: 12

Pada gambar tersebut *tunneling* terbentuk antara *client* dan *server*, sehingga data siap untuk dikirimkan. Apabila *tunnel client* ingin mengirimkan data pada *tunnel server* atau sebaliknya maka *client* harus menambahkan data transfer protokol *header* pada data (enkapsulasi). *Client* kemudian mengirim hasil dari enkapsulasi ini melalui internet untuk kemudian akan di-*routing* pada *tunnel server*. Setelah *tunnel server* menerima data tersebut, kemudian *tunnel server* akan memisahkan *header* data transfer protokol (dekapsulasi) dan kemudian mem-*forward* data ke jaringan yang dituju.

2.2.4 Autentikasi dan Enkripsi pada VPN

Jaringan VPN selalu tidak akan pernah lepas dari dua hal yakni autentikasi dan enkripsi. Autentikasi adalah suatu proses untuk memastikan bahwa kedua ujung koneksinya adalah benar *user*-nya sesuai dengan yang diberikan kewenangan untuk mengakses suatu *server*. Sedangkan enkripsi berfungsi untuk melakukan pengacakan terhadap suatu data agar tidak dapat dibaca oleh orang lain. Meskipun baik autentikasi dan enkripsi memiliki cara kerja sendiri-sendiri,

namun keduanya seringkali dipadukan agar didapatkan suatu sistem keamanan yang lebih baik.

Pada umumnya jenis autentikasi yang paling mudah dan banyak digunakan orang adalah *Login Password*, dimana untuk dapat melakukan koneksi maka seorang *user* harus memasukkan *username* dan *password* yang benar baru kemudian bisa melakukan akses ke *server*. Namun demikian hal tersebut hanya dilakukan di awal *session* saja. Permasalahan bisa saja terjadi ketika ada *attacker* yang berada di tengah antara *server* dan *client* tersebut yang kemudian mengambil alih *session* sehingga *client* yang sebenarnya telah terputus *session*-nya sementara *attacker* berhasil menguasai *session* tersebut dan dianggap *user* yang benar oleh *server*.

Saat ini kebutuhan akan autentikasi tidaklah cukup hanya menggunakan *form login* saja. Tipe autentikasi yang saat ini mulai banyak digunakan adalah penggunaan sertifikat dan *digital signature* atau tanda tangan digital.

Secara sederhana enkripsi adalah sebuah proses melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti. Enkripsi juga bisa diartikan sebagai *cipher*. Terdapat beberapa enkripsi yang sering dilakukan, yaitu IPSec dan SSL.

Metode enkripsi dibagi menjadi algoritma *symmetric key* dan algoritma *asymmetric key*. Pada algoritma *symmetric key* (misalkan, DES dan AES), pengirim dan penerima harus memiliki kunci yang digunakan bersama dan dijaga kerahasiaannya. Pengirim menggunkan kunci ini untuk enkripsi dan penerima menggunakan kunci yang sama untuk dekripsi. Pada algoritma *asymmetric key* (misalkan, RSA), terdapat dua kunci terpisah, sebuah *public key* diterbitkan dan membolehkan siapapun pengirimnya untuk melakukan enkripsi, sedangkan sebuah *private key* dijaga kerahasiannya oleh penerima dan digunakan untuk melakukan dekripsi.

2.2.5 Metode Pertukaran Kunci dan Autentikasi

Metode pertukaran kunci dan autentikasi ada beberapa macam, yaitu:

Metode Pertukaran kunci:

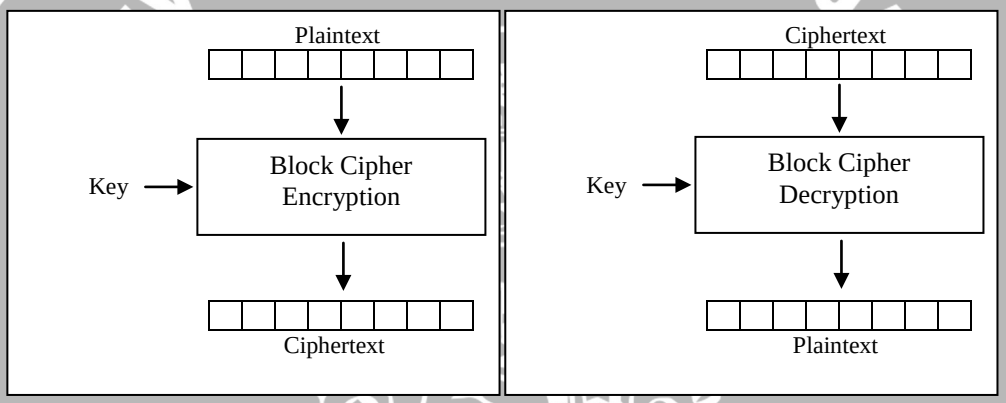
1. RSA, *client* mengirimkan kunci pre_master setelah melakukan enkripsi.
2. DH, *client* dan *server* menghasilkan kunci pre_master secara bebas.

Metode autentikasi:

1. *Server Authentication*, yaitu proses autentikasi dilakukan oleh *server*, algoritma yang digunakan RSA dan DSA.
2. *Client Authentication*, yaitu proses autentikasi dilakukan pada *client*, algoritma yang digunakan RSA dan DSA.
3. *Anonymous*, tidak terjadi proses autentikasi.

2.2.6 Kriptografi pada VPN

Istilah ‘‘*cryptography*’’ berarti tulisan yang bersifat rahasia berasal dari bahasa Yunani, ‘‘*kryptos*’’, ‘‘tersembunyi’’ dan ‘‘*graphein*’’, ‘‘menulis’’. Modern kriptografi mengenalkan beberapa bentuk penulisan kode, yaitu *Symmetric cryptography* dan *Public key (asymmetric)*.



Gambar 2.19 *Block Cipher* pada *Cryptography*
Sumber :http://www.wikipedia.org/wiki/Block_cipher_modes_of_operation

Pada gambar diatas, *block cipher* tidak hanya dipakai pada dekripsi. Pada *symmetric cryptography* dikenal *symmetric cipher* yang terdiri dari *block plaintext* dengan jumlah tetap dan *ciphertext* untuk setiap *block plaintext*. *Cipher* adalah algoritma untuk menampilkan enkripsi dan dekripsi, serangkaian langkah yang terdefinisi yang diikuti sebagai prosedur. Informasi yang original dikenal dengan nama *plaintext*, sedangkan bentuk enkripsinya disebut *ciphertext*.

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen teks terang/ *plaintext* dan yang berisi elemen teks sandi/ *ciphertext*.

Enkripsi : E(P) = C

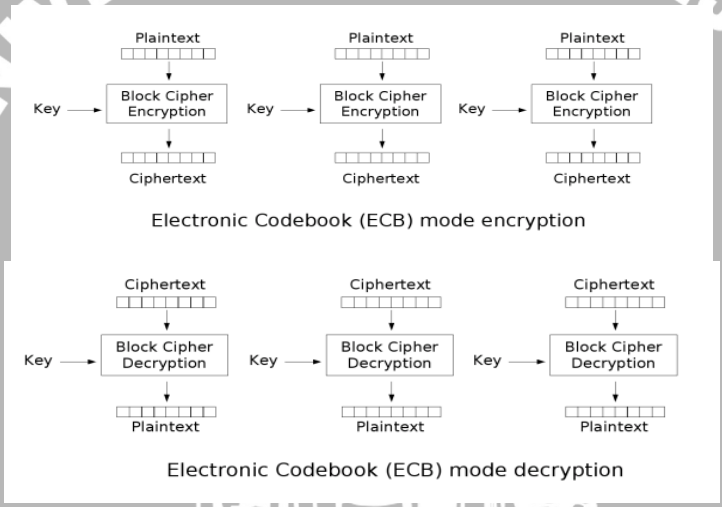
Dekripsi : D(C) = P atau D(E(P)) = P

Dimana elemen-elemen teks terang dinotasikan dengan P, elemen-elemen teks sandi dinotasikan dengan C, sedangkan untuk proses enkripsi dinotasikan dengan E dan dekripsi dinotasikan dengan D.

Block cipher dapat digunakan dalam beberapa mode untuk proses enkripsi dan dekripsi. Ada beberapa standard mode yang digunakan untuk *cipher*, yaitu:

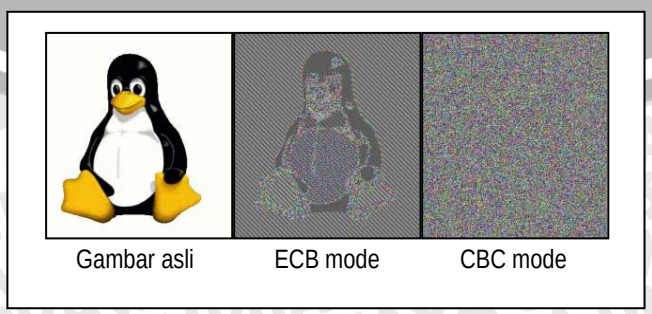
ECB (Electronic Code Book)

Mode ECB adalah mode yang paling mudah untuk diimplementasikan, dimana setiap block akan dienkripsi secara tersendiri. Kelemahan dari mode ini adalah mengenkripsi secara identik *plaintext* kedalam *ciphertext*. ECB tidak menyembunyikan *pattern* (pola) dari data tersebut. Sehingga masih terlihat samar dan bisa diraba bentuk asli *plaintext*.



Gambar 2.20 *ECB Mode Encryption* dan *Decryption*
Sumber : http://www.wikipedia.org/wiki/Block_cipher_modes_of_operation

Tiap pesan dimasukkan ke *block* dan masing-masing dienkripsi secara terpisah. Kelemahan dari metode ini adalah pola *ciphertext* yang dihasilkan masih kelihatan sehingga pesan-pesan menjadi tidak *confidentiality* dan tidak direkomendasikan dalam penggunaan protokol kriptografi.



Gambar 2.21 *Perbandingan ECB* dan *CBC*
Sumber : http://www.wikipedia.org/wiki/Block_cipher_modes_of_operation

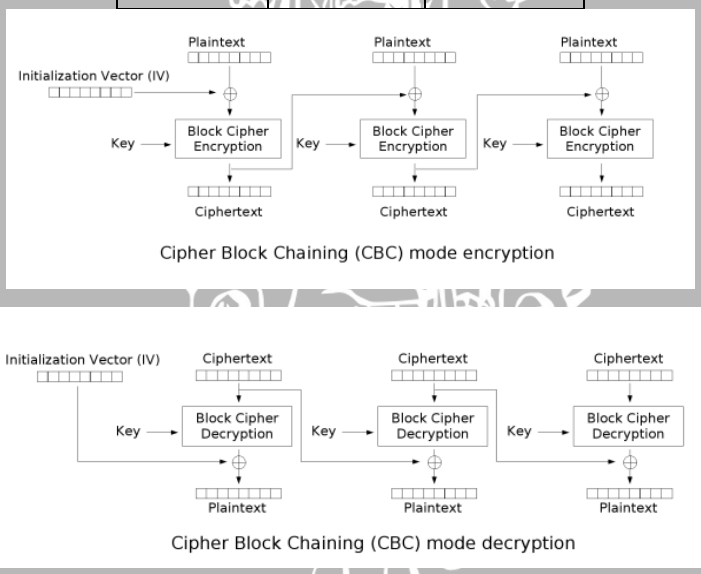
Gambar 2.20 menunjukkan bahwa hasil enkripsi antara ECB dan CBC, hal ini menyebabkan ECB sangat tidak disarankan untuk aplikasi yang membutuhkan tingkat sekuritas tinggi.

CBC (Cipher Block Chaining)

Setiap *block plaintext* di XOR-kan dengan *ciphertext* sebelumnya. *Output* dari XOR akan masuk kedalam proses enkripsi. Agar setiap data bersifat unik, *initialization vector* digunakan dengan kombinasi *plaintext* menggunakan operasi XOR pada proses awal enkripsi. Maksud dari operasi XOR adalah berdasarkan tabel kebenaran seperti tabel di bawah ini:

Tabel 2.8Tabel Kebenaran XOR

In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1



Gambar 2.22 *CBC Mode Encryption* dan *Decryption*
Sumber : http://www.wikipedia.org/wiki/Block_cipher_modes_of_operation

SSL menggunakan CBC sebagai mode pada *cipher block* algoritma enkripsi. Meskipun CBC lebih lambat dari ECB tapi jaminan keamanan yang lebih baik adalah alasan utamanya.

Untuk enkripsi, SSL dapat menggunakan beberapa algoritma, diantaranya :

DES (Data Encryption Standard)

DES adalah *cipher* yang digunakan oleh FIPS (Federal Information Processing Standard) untuk Amerika Serikat pada tahun 1976, dan secara

bertahap tersebar luas digunakan oleh seluruh dunia. Algoritma ini pada permulaan munculnya sangat kontroversi karena mempunyai *key length* yang sangat pendek. Selain itu ditengarai mempunyai *backdoor* oleh *National Security Agency* (NSA). Saat ini DES diketahui menjadi tidak aman untuk beberapa aplikasi. Hal ini dikarenakan ukuran *key*-nya hanya 56 bit. Algoritma ini telah diperbarui kedalam bentuk 3DES.

3DES (Triple DES)

3DES adalah sekumpulan *cipher (block cipher)* dari DES yang digunakan tiga kali. 3DES juga dikenal dengan TDES atau TDEA (Triple Data Encryption Algorithm). Semenjak ditemukan DES pertama kali yang menggunakan 56 bit, *key length*, DES sangat rentan terhadap serangan *brute force*. *Block cipher* pada 3DES sama dengan DES, yaitu 64 bit, sedangkan *key size* untuk 3DES 168 bit dengan pengulangan DES (56 bit) sebanyak tiga kali.

AES (Advanced Encryption Standard)

AES adalah bentuk baru dari *cipher*, yang merupakan penyempurnaan dari DES yang hanya terdiri dari 64 bit *block cipher* dan 56 bit *key*. Sedangkan AES menggunakan 128 bit *block cipher* serta 128, 192, dan 256 bit *key*.

Untuk *digest*/ autentikasi, SSL dapat menggunakan beberapa algoritma, diantaranya :

MD5 (Message Digest 5)

MD5 didesain oleh Ronald Rivest pada tahun 1991 untuk menggantikan *hash function* sebelumnya, MD4. Pada tahun 1996, sebuah kecacatan ditemukan dalam desainnya, walau bukan kelemahan fatal, pengguna kriptografi mulai menganjurkan menggunakan algoritma lain, seperti SHA-1.

MD5 ialah fungsi hash kriptografi yang digunakan secara luas dengan *hash value* 128 bit. Pada standar Internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah *file*. Pada tahun 2004, kecacatan-kecacatan yang lebih serius ditemukan. Hal ini menyebabkan penggunaan algoritma tersebut dalam tujuan untuk keamanan jadi makin dipertanyakan.

SHA-1 (Secure Hash 1)

Keluarga SHA juga didesain untuk keperluan keamanan *(cryptographic hash function)*. Pada umumnya fungsi terpenting dari SHA-1 dipakai pada aplikasi dan *protocol security* yang sangat luas termasuk TLS, SSL, PGP, SSH, S/ MIME, dan IPSec. Algoritma SHA didesain oleh NSA dan dipublikasikan oleh pemerintah Amerika Serikat.

Rilis pertama dari keluarga ini telah dipublikasikan pada tahun 1993, dan disebut SHA. Algoritma ini juga sering disebut SHA-0 untuk tidak membingungkan dengan SHA-1. SHA-0 dan SHA-1 mempunyai *key length* 160 bit. Dua tahun kemudian, SHA-1 dipublikasikan. Empat varian baru telah dirilis dan pembuatannya didesain secara berbeda pula, SHA-224, SHA-256, SHA-384, dan SHA 512, yang sering disebut juga sebagai SHA-2.

2.2.7 Algoritma Kompresi

Tidak hanya IPSec, SSL *(Security Socket Layer)* juga mendukung kompresi, namun proses kompresi dilakukan oleh openSSL yang mendukung SSL juga OpenVPN menggunakan algoritma LZO sebagai metode kompresinya.

LZO *(Lempel-Ziv-Oberhumer)* adalah algoritma kompresi data yang berfokus pada kecepatan dekompresi dan bersifat *lossless*. LZO merupakan algoritma kompresi blok, yaitu mengkompres dan dekompres sebuah blok data. Ukuran blok harus sama untuk kompresi dan dekompresi. Selain itu, algoritma LZO juga bersifat *realtime* sehingga dapat diterapkan dalam kompresi data *stream*.

2.2.8 *Open*VPN

*Open*VPN mempunyai fitur penuh yang mendukung VPN SSL yang mengakomodasi konfigurasi yang fleksibel dan luas, termasuk *remote access*, *site to site* VPN, Wifi *security*, dan skala *remote access* yang bersifat *enterprise*.

*Open*VPN mengimplementasikan model OSI *layer* dua atau tiga untuk keamanan jaringan yang mempunyai standar protokol SSL, mendukung autentikasi *client* yang menggunakan sertifikat dan autentikasi yang mengijinkan *user* untuk mengakses melalui *firewall* yang dipasang pada VPN.

*Open*VPN dapat berjalan diatas sistem operasi *Linux, Windows 2000/ XP, OpenBSD, FreeBSD, NetBSD, Ubuntu, Mac OS X*, dan *Solaris*. Layanan yang disediakan diantaranya :

- Menyediakan *tunnelling* dengan banyak IP *subnetwork* atau *virtual Ethernet adapter* dengan menggunakan *single* UDP dan TCP *port.*
- Konfigurasinya *scalable,* bervariasi sesuai dengan kebutuhan dan dapat dihubungkan dengan banyak *client* sebagai VPN *client.*
- Menggunakan enkripsi, autentikasi, dan sertifikat yang ada pada OpenSSL yang dapat melindungi jaringan *private* pada saat terhubung dengan internet.
- Menggunakan banyak *cipher*, ukuran *key* yang beragam, HMAC *digest* (untuk integritas dalam hal pengecekan data) yang didukung penuh oleh OpenSSL.
- Menyediakan *static key* dan *public key* pada enkripsinya.
- Menyediakan kompresi untuk menghemat *bandwidth*.

2.2.9 *Networking* Menggunakan *Open*VPN

James Yonan, kreator dari *Open*VPN, menggunakan *driver universal* TUN/ TAP untuk *layer networking* pada OpenVPN. *Driver* TUN/ TAP merupakan proyek *open-source* yang disertakan dalam semua distribusi Linux/ UNIX *modern,* juga pada *Windows* dan *Mac OS X*. Penggunaan TUN/ TAP menghilangkan banyak kerumitan dari struktur *Open*VPN. TUN/ TAP dikembangkan untuk menyediakan dukungan kernel Linux untuk *tunneling* trafik IP yang merupakan *interface network virtual*. Setiap aplikasi yang bisa menggunakan *interface network* dapat pula menggunakan *interface tunnel*. Setiap teknologi yang berjalan pada jaringan juga dapat berjalan pada interface TUN/ TAP. *Driver* ini adalah salah satu dari beberapa faktor utama yang menjadikan *Open*VPN mudah dimengerti, mudah dikonfigurasi, sekaligus sangat aman.

Aplikasi dapat membaca/ tulis dengan interface ini, *driver* akan mengambil semua data dan menggunakan *library* kriptografi SSL/ TLS untuk mengenkripsinya. Data dipaket dan dikirim ke ujung lain dari *tunnel*. Pemaketan ini dilakukan dengan UDP standar atau paket TCP opsional. UDP biasanya pilihan pertama, namun TCP kadang dipakai.

*Open*VPN ”mendengarkan” *(listening)* pada semua *device* TUN/ TAP, mengambil trafiknya, mengenkripsinya, dan mengirimnya menuju rekan VPN

lainnya. *Open*VPN di tempat lain menerima data, mendekripsikannya, menyerahkannya pada *device network virtual*, dimana aplikasi telah menunggu data tersebut.

Karena *Open*VPN menggunakan paket *network* standar, NAT dimungkinkan. *User* dalam sebuah jaringan lokal di Jakarta dengan IP lokal dapat memulai sebual *tunnel* menuju *user* lain dalam jaringan lokal di Surabaya. *Open*VPN juga dapat berjalan pada satu *port* saja, sehingga memudahkan administrasi pada *firewall*, misalnya.

Karena *interface* jaringan merupakan *interface* standar Linux, segala hal yang dapat diterapkan pada sebuah NIC, juga dapat diterapkan pada *tunnel* VPN, misalnya:

- *Firewall* dapat membatasi dan mengontrol trafik
- *Traffic shaping* atau pembatasan trafik dimungkinkan

2.2.10 *Command-command* VPN

Untuk mengaktifkan OpenVPN didahului dengan *command Openvpn* --config sample.ovpn. Dimana konfigurasi OpenVPN yang berekstensi*.ovpn adalah *file* yang berisi konfigurasi yang akan digunakan pada OpenVPN dan berisi *command-command* OpenVPN yang sesuai dengan jenis koneksi VPN yang dibutuhkan. Berikut *command-command* pada OpenVPN :

Tabel 2.9 *Command-command* yang digunakan pada OpenVPN

Parameter	Options	Function	Useage	Example
remote	<*hostname*><IP>	Menunjuk ke ujung lain (endpoint) dari *tunnel*	*Command line and config file*	--remote vpn.dyndns.org
dev	<*device*>	Perintah pemilihan *device* yang akan digunakan	*Command line and config file*	--dev tun --dev tap
ifconfig	For TUN *devices:* <local IP> <remote IP> For TAP *devices:* <local IP><subnet mask>	Mengeset IP *Virtual tunnel* endpoints dan netmask pada *tunnel*	*Command line and config file*	ifconfig 10.3.0.2.10.3.0.1 – ifconfig 10.3.0.2 255.255.255.0
Secret	*File* containing the pre-shared *key*	Perintah pengalokasian dari preshared *key*	*Command line and config file*	-- Secret *key*.txt
comp-lzo	<yes><no> <adaptive> (default)	*Open*VPN menggunakan lzo library untuk mengkompress *tunnel* traffic	*Command line and config file*	-- *comp-lzo*

port	<port number>	Mengspesifikasi port (baik local maupun remote) yang digunakan	*Command line and config file*	-- port 5001
proto	<udp><tcp-*client*> <tcp-*server*>	Mengeset *protocol* yang akan digunakan oleh *Open*VPN. TCP *client* akan mencoba untuk memulai koneksi, sedangkan TCP *server* hanya menunggu *client*	*Command line and config file*	-- proto udp -- proto tcp-*client* –tcp-*server*
n-mtu	<mtu size>	Mengeset transmisi unit secara maksimal	*Command line and config file*	--tun-mtu 1200
dev-node	<*interface*>*name*	Mengspesifikasi nama dari *interface* yang akan digunakan	*Command line and config file*	--dev-node *Open*vpn1
ping	<*Seconds*>	Mengirimkan ping ke ujung *tunnel* partner yang lain melalui *tunnel* setelah beberapa detik tanpa traffic	*Command line and config file*	--ping 10
ping-restart	<*Seconds*>	Setelah beberapa detik tanpa menerima paket dari computer yang telah terkoneksi VPN, *tunnel* akan melakukan restart	*Command line and config file*	--ping-restart 60
ping-timer-rem	-	ping-restart berjalan hanya saat remote *address* diberikan	*Command line and config file*	--ping-timer-rem
persist-tun	-	Menjaga *device* tun/tap tetap up saat *Open*VPN melakukan restart	*Command line and config file*	--persist-tun
persist-*key*	-	*Open*VPN tidak akan membaca ulang *keys* pada saat restart	*Command line and config file*	--persist-*key*
resolv-retry	<*Seconds*>	Mengeset waktu dimana *Open*VPN akan mencoba untuk memperbaiki *hostname* sebelum menyerah	*Command line and config file*	--resolv-retry 86400
verb	<verbosity level>	Mengeset level dari verbosity, 0 adalah yang paling rendah, 11 adalah detail level maksimal	*Command line and config file*	--verb 4
mute	<number of messages>	*Open*VPN akan mencetak hanya 10 pesan con*Secutive* dari kategori yang sama	*Command line and config file*	--mute 10
key	<*file*>	Mendefinisikan *file* local machine's *key*	*Command line and config file*	*--key keys*/VPN-*Client.key*
tls-*server*	-	Local machine belagak seolah-olah sebagai TLS *server*	*Command line and config file*	--tls-*server*
tls-*client*	-	Local machine berlagak seolah-olah sebaga TLS *client*	*Command line and config file*	--tls-*client*

2.3 Parameter Performansi VoIP-VPN

Menurut ITU-T E.800, QoS adalah : *"Sekumpulan efek performansi yang menentukan derajat kepuasan pengguna terhadap service yang diberikan oleh jaringan"*. Sedangkan dari sudut pandang jaringan telekomunikasi QoS adalah : *"Kemampuan suatu jaringan untuk menyediakan layanan yang lebih baik pada trafik data tertentu pada berbagai jenis platform teknologi"* (Onno W. Purbo, 2001: 125). Performansi sistem merupakan ukuran baik buruknya sebuah sistem. Sistem yang dimaksud di sini adalah sistem VoIP melalui VPN. Parameter-parameter yang digunakan di sini adalah *bandwidth*, *delay end-to-end*, *jitter*, *throughput*, dan *packet loss*.

2.3.1 *Bandwidth Per Call*

Bandwidth adalah kecepatan maksimum yang dapat digunakan untuk melakukan transmisi data antar komputer pada jaringan IP atau internet. Dalam VoIP, *bandwidth* merupakan suatu hal yang harus diperhitungkan agar dapat kualitas suara yang baik.

Bandwidth yang diperlukan saat berkomunikasi suara tergantung pada *codec* yang digunakan. Untuk menghitung *bandwidth* yang dibutuhkan, maka sangat penting untuk mengetahui model susunan protokol IP. Untuk VoIP, protokol yang terkait diantaranya: RTP, UDP, IP, dan *Network Interface* (seperti *ethernet* atau *tokenring*). Besarnya paket tersebut ditunjukkan oleh Gambar 2.8.

Kebutuhan *bandwidth* tiap panggilan didefinisikan dengan persamaan (Sayadian, 2006: 54):

$$BW(b/s) = (V + I + L)(B/pkt) * 8(b/B) * P(pkt/s) \quad (2\text{-}1)$$

dimana:

BW = *bandwidth* tiap panggilan (bps)

V = *voice payload* untuk tiap paket, tergantung *codec* yang digunakan (lihat Tabel 2.5)

I = *overhead header* IP/UDP/RTP per paket, bernilai 40 byte, kecuali kompresi *header* diaktifkan

L = *overhead header link-layer*, tergantung media yang digunakan (lihat Tabel 2.4)

8 = jumlah bit tiap byte

P = jumlah paket/ *frame* yang dihasilkan oleh *codec* tiap detik

Karena satu kanal *voice* pada VoIP digunakan untuk dua arah transmisi (pada satu pembicaraan telepon ada dua arah transmisi), maka *bandwidth* satu kanal *voice* adalah *bandwidth* pada masing-masing arah transmisi. Pemakaian *bandwidth* yang kecil merupakan salah satu kunci sukses penerapan VoIP.

2.3.2 *Delay End-to-end*

Delay (waktu tunda) adalah waktu yang dibutuhkan untuk mengirimkan paket data dari sumber sampai ke tujuan. *Delay end-to-end* pada jaringan IP merupakan penjumlahan *delay-delay* yang ada dalam perjalanan paket dari sumber ke tujuan.

Sesuai dengan rekomendasi dari ITU-T pada G.114, maka *delay* keseluruhan dalam jaringan yang menerapkan layanan komunikasi suara tidak boleh melebihi 400ms. Tabel 2.10 berikut ini menunjukkan batas *delay* yang diijinkan dalam komunikasi suara.

Tabel 2.10 Spesifikasi *Delay*

***Delay* (ms)**	**Keterangan**
0 – 150	Kualitas percakapan masih dapat diterima
150 – 400	Kualitas percakapan masih dapat ditoleransi
> 400	Tidak dapat diterima

Sumber: Sinreich, 2006: 303

Pada proses *audio streaming* dari *server* ke *client*, data ditransmisikan dalam bentuk paket-paket data kemudian dienkapsulasi dengan menambahkan *header* pada masing-masing layer pada jaringan UDP/IP. Setelah paket-paket data selesai dienkapsulasi, kemudian pada layer di bawahnya yaitu *internet layer* di-*rute*-kan ke *node* tujuan melalui media transmisi dalam bentuk *frame-frame* data. Pada *node* tujuan *frame-frame* tersebut didekapsulasi menjadi paket-paket data selanjutnya dibawa ke layer di atasnya.

Delay end-to-end dapat dituliskan sebagai berikut

$$t_{end-to-end} = t_{proc} + t_{trans} + t_w + t_p \quad (2\text{-}2)$$

dengan:

t_{proc} : *delay* proses (ms)

t_{trans} : *delay* transmisi/serialisasi (ms)

t_w : *delay* antrian (ms)

t_p : *delay* propagasi (ms)

a. *Delay* proses

Delay proses adalah waktu yang dibutuhkan untuk memproses paket data dan untuk menentukan ke mana data tersebut akan diteruskan. *Delay* proses berupa *delay* enkapsulasi dan *delay* dekapsulasi.

Apabila *node* sumber ingin data ke *node* tujuan, maka proses yang terjadi adalah data aplikasi akan dikirimkan ke *transport layer*.*Transport layer* yang digunakan adalah UDP karena paket yang dikirimkan adalah paket *video*. Pada layer TCP, data kemudian dienkapsulasi dengan *header*.

$$W_{segmen} = MSS + Header_{RTP} + Header_{UDP} \quad (2\text{-}3)$$

dengan

W_{segmen} = panjang segmen pada layer 4 (byte)

MSS = panjang segmen maksimum (byte)

$Header_{RTP}$ = panjang *header* RTP (12 byte)

$Header_{UDP}$ = panjang *header* UDP (8 byte)

Dari layer 4 atau layer *transport*, segmen kemudian dikirim ke layer 3 atau layer *network* untuk dienkapsulasi menjadi *datagram* IP. Apabila panjang segmen pada layer diatasnya melebihi MTU IP yaitu 1500 byte, maka segmen perlu untuk difragmentasi sebelum dienkapsulasi. Kemudian datagram IP dienkapsulasi dengan *heade*r IP, sehingga panjang *datagram* IP sebagai berikut.

$$W_{datagram} = W_{segmen} + Header_{IP} \quad (2\text{-}4)$$

dengan :

$W_{datagram}$ = panjang datagram IP (byte)

W_{segmen} = panjang segmen TCP (byte)

Header IP = panjang *header* IP (20 byte)

Kemudian *datagram* IP dienkapsulasi dengan *header* pada layer 2 pada skripsi ini menggunakan *ethernet* sebagai layer pada datalink.

$$W_{frame} = W_{datagram} + Header_{ethernet} \quad (2\text{-}5)$$

dengan

W_{frame} = panjang *frame ethernet* (byte)

$W_{datagram}$ = panjang *datagram* IP (byte)

Header = panjang header *ethernet* (14 byte)

Sedangkan *delay* enkapsulasi adalah :

$$t_{enc} = \frac{W_{frame} - MSS}{C_{pros}} x8 \quad (2\text{-}6)$$

dengan :

t_{enc} = *delay* enkapsulasi (s)

W_{frame} = panjang *frame Ethernet* (byte)

MSS = *Maximum Segment Size* (byte)

C_{pros} = kecepatan pemrosesan data (bps)

Sedangkan *delay* dekapsulasi dirumuskan :

$$t_{dec} = \frac{W_{frame} - MSS}{C_{pros}} x8 \quad (2\text{-}7)$$

dengan :

t_{dec} = *delay* dekapsulasi (s)

W_{frame} = panjang *frame Ethernet* (byte)

MSS = *Maximum Segment Size* (byte)

C_{pros} = kecepatan pemrosesan data (bps)

Sehingga delay proses dapat dituliskan sebagai berikut.

$$t_{proc} = t_{enc} + t_{dec} \quad (2\text{-}8)$$

dengan :

t_{proc} = *delay* proses

t_{enc} = *delay* enkapsulasi (s)

t_{dec} = *delay* dekapsulasi (s)

***b. Delay* transmisi/ serialisasi**

Delay serialisasi adalah waktu yang diperlukan untuk menempatkan semua *frame* data pada media transmisi atau sebaliknya, sehingga efektif dan efisien dalam penggunaan kapasitas saluran/media transmisi. *Delay* serialisasi menjadi masalah saat kapasitas saluran semakin kecil. Panjang paket multimedia dipresentasikan dengan satuan *byte*, dimana setiap 1 *byte* = 8 *bit*. *Delay* serialisasi dapat dituliskan sebagai berikut.

$$t_{trans} = \frac{(L + L')}{C} x8 = \frac{W_{frame}}{C} x8 \quad (2\text{-}9)$$

dengan:

t_{trans} = *delay* transmisi (s)

L = panjang paket multimedia (byte)

L' = panjang *header* (*byte*)

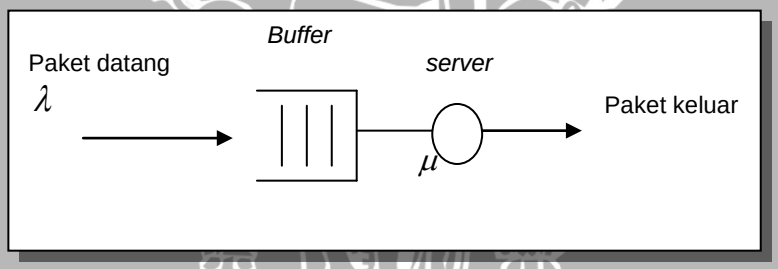
C = kapasitas kanal (bps)

W_{frame} = panjang *frame Ethernet*

c. *Delay* antrian

Delay antrian adalah waktu dimana paket data tersebut berada dalam antrian untuk ditransmisikan. Selama waktu ini paket data menunggu sampai selesainya paket lain ditransmisikan. *Delay* antrian terjadi sangat dinamis, jika antrian kosong dan tidak ada paket data lain yang sedang ditransmisikan maka *delay* antrian tidak terjadi atau sama dengan nol.

Model antrian yang digunakan dalam analisis ini menggunakan model antrian M/M/1. M pertama menunjukkan distribusi *Poisson* yang berarti acak, M kedua berarti distribusi waktu pelayanan eksponensial, angka 1 menunjukkan bahwa jumlah *server* adalah tunggal. Disiplin antrian yang digunakan adalah FIFO (First In First Out), yaitu paket yang datang akan diproses terlebih dahulu. Gambar 2.23 menunjukkan sistem antrian M/M/1.



Gambar 2.23 Sistem Antrian M/M/1
Sumber: I Made Wiryana, 1992: 2

Parameter – parameter pada model tersebut adalah :

1. Kapasitas link adalah C *bit/sec* dan panjang paket data adalah m bit. Besarnya kapasitas link akan menentukan kecepatan pelayanan ($\mu = C / m$ (1/detik)).
2. Interval waktu untuk permintaan (*request*) merupakan distribusi *poisson* dengan kecepatan kedatangan data adalah λ (1/detik).

Besarnya *delay* antrian yang terjadi pada *client* ditentukan dengan persamaan 2-10 (I Made Wiryana, 1992: 2)

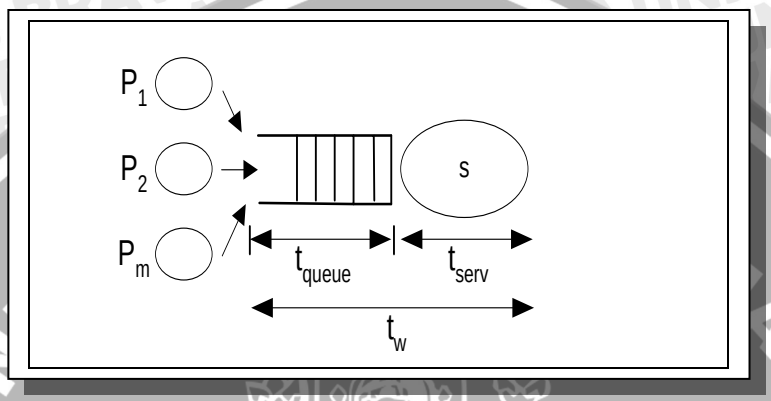
$$t_w = t_{queue} + t_{serv} \quad (2\text{-}10)$$

dengan:

t_w = *delay* antrian pada *client* (s)

t_{queue} = waktu tunggu paket pada *client* (s)

t_{serv} = waktu rata-rata pelayanan *client* (s)



Gambar 2.24 Analisis *Delay* Antrian
Sumber : I Made Wiryana, 1992: 2

Sementara itu:

$$t_{serv} = \frac{1}{\mu} \quad (2\text{-}11)$$

dengan :

t_{serv} = waktu rata-rata pelayanan *client* (s)

μ = kecepatan pelayanan *client* (1/s)

Nilai kecepatan pelayanan *client* diperoleh dengan persamaan 2-12 (Mischa Schwartz, 1987: 23):

$$\mu = \frac{C}{L} \quad (2\text{-}12)$$

dengan :

μ = kecepatan pelayanan *client* (1/s)

C = kapasitas kanal (bps)

L = panjang paket multimedia (bit)

Performansi sistem antrian ditunjukkan dalam bentuk utilitas:

$$\rho = \frac{\lambda}{\mu} \Rightarrow \lambda = \mu\rho \quad (2\text{-}13)$$

dengan :

ρ = utilitas ($0 < \rho < 1$)

λ = kecepatan kedatangan paket pada *client* (1/s)

μ = kecepatan pelayanan *client* (1/s)

Dengan menggunakan teori Little diperoleh nilai *delay* antrian (I Made Wiryana, 1999: 7) :

$$t_w = \frac{1}{\mu(1-\rho)} \quad (2\text{-}14)$$

dengan :

t_w = *delay* antrian (s)

μ = kecepatan pelayanan *client* (1/s)

ρ = utilitas ($0 < \rho < 1$)

Dari persamaan di atas maka waktu tunggu paket dirumuskan:

$$t_{queue} = t_w - t_{serv} = \frac{\lambda}{\mu(\mu-\lambda)} \quad (2\text{-}15)$$

dengan :

t_{queue} = waktu tunggu paket pada *client* (s)

t_w = *delay* antrian (s)

t_{serv} = waktu rata-rata pelayanan *client* (s)

λ = kecepatan kedatangan paket pada *client* (1/s)

μ = kecepatan pelayanan *client* (1/s)

Dari Persamaan (2-11) dan (2-15) maka *delay* antrian dapat dituliskan sebagai :

$$t_w = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu} \quad (2\text{-}16)$$

dengan:

t_w = *delay* antrian (s)

λ = kecepatan kedatangan paket pada *client* (1/s)

μ = kecepatan pelayanan *client* (1/s)

***d. Delay* propagasi**

Delay propagasi adalah waktu yang dibutuhkan untuk merambatkan paket multimedia melalui media transmisi dari *server* ke *client*. *Delay* propagasi

tergantung antara sumber dan tujuan ataupun antara sumber dengan *hop* terdekat, dan juga dipengaruhi oleh jenis saluran yang dilewatinya. *Delay* propagasi akan menjadi masalah yang serius pada hubungan jarak jauh.

$$t_p = t_{DTE} + t_{UTP} + t_{Router} \quad (2\text{-}17)$$

dengan :

t_p = *delay* propagasi (s)

t_{DTE} = *delay* pada NIC (s)

t_{UTP} = *delay* pada UTP CAT 5 (s)

t_{Router} = *delay* pada *router* (s)

Tabel 2.11 Komponen *Delay Fast Ethernet*

Kompenen	Delay per meter	Delay maksimum	Round – Trip per Meter	Maksimum Rount Trip Delay
Two TX/FX DTEs	-	25		100
Categaory 3	0,57	57 (100 meter)	1,14	114 (100 meter)
Categaory 4	0,57	57 (100 meter)	1,14	114 (100 meter)
Categaory 5	0,556	55,6 (100 meter)	1,112	111,2 (100 meter)
Shielded Twisted Pair	0,556	55,6 (100 meter)	1,112	111,2 (100 meter)
Fiber Optic cable	0,5	206 (412 lembar)	1,0	412 (412 meter)

Sumber: www.oreilly.com/catalog/enettds/CH.13.htm

2.3.3 *Throughput*

Throughput adalah jumlah data per satuan waktu yang dikirimkan melalui sebuah saluran fisik atau melewati sebuah *node* jaringan. *Throughput* merupakan kecepatan transfer data efektif. Sistem *throughput* atau *aggregate throughput* adalah jumlah total data *rate* yang dikirimkan menuju semua terminal dalam jaringan. *Throughput* maksimum sinonim dengan kapasitas sebuah saluran jaringan, yaitu ketika jumlah beban (jumlah data yang datang) adalah sangat besar.

2.3.4 *Jitter*

Jitter adalah perbedaan waktu kedatangan dari suatu paket ke penerima dengan waktu yang diharapkan. Paket VoIP akan dikirim ke penerima setiap interval waktu tertentu dan seharusnya akan diterima juga dalam setiap waktu tertentu, tapi kenyataannya waktu kedatangan paket tidak sama. *Jitter* dapat menyebabkan sampling di sisi penerima menjadi tidak tepat sasaran sehingga informasi menjadi rusak. *Jitter* disebabkan factor antrian *(queue)* yang melebihi

batas waktu yang ditentukan dan atau ukuran paket yang terlalu besar sehingga tidak mungkin ditransmisikan dalam jaringan yang kecepatannya rendah.

2.3.5 *Packet Loss*

Packet Loss merupakan hilangnya paket data dalam transmisi data. Hal ini dapat terjadi karena berbagai hal, mulai dari *layer* fisik dengan contoh degradasi sinyal, tumbukan antar paket dalam sebuah *switch*, hingga layer aplikasi dengan contoh kesalahan *software*.

2.3.6 Probabilitas *Packet Loss*

Probabilitas *packet loss* pada jaringan IP dihitung dari probabilitas *packet loss* yang terjadi pada jaringan lokal serta pada jaringan IP, dengan persamaan:

$$\rho_{network} = 1 - [(1 - \rho_{eth})\ (1 - \rho_{IPB})^{h}] \quad (2\text{-}18)$$

$\rho_{network}$ = probabilitas *packet loss* pada jaringan

ρ_{eth} = probabilitas *packet loss* pada jaringan *ethernet*

ρ_{IPB} = probabilitas *packet loss* pada IP

h = jumlah hop pada jaringan IP

Probabilitas *packet loss* pada suatu jaringan dapat dihitung dari probabilitas *bit error* (BER) di jaringan tersebut, dengan persamaan (Schwartz, 1987: 132):

$$\rho_{net} = 1 - (1 - \rho_e)^{l+l'} \quad (2\text{-}19)$$

dengan:

ρ_{net} = probabilitas *bit error* pada suatu jaringan

l = panjang paket (bit)

l’ = panjang header (bit)

ρ_e = probabilitas *bit error* (BER) standar

2.3.7 Pengukuran Kualitas Suara VoIP

2.3.7.1 Perhitungan MOS (Mean Opinion Score) dengan *Polling*

MOS memberikan indikasi numerik dari kualitas suara secara subyektif. MOS diekspresikan sebagai nomor tunggal bernilai 1 hingga 5, dimana 1 merupakan kualitas terburuk, dan 5 merupakan kualitas terbaik. Penggunaan MOS untuk uji coba terhadap kualitas suara diatur dalam standar ITU-T P.800.

Ukuran ini menggunakan metode taksiran kualitas secara subyektif. Metode tersebut dibagi lagi menjadi dua, *conversation opinion test* dan *listening*

opinion test. Subyek tes memutuskan kualitas dari sistem transmisi suara dengan cara melakukan panggilan atau dengan cara mendengarkan sampel suara. Setelah itu kualitas suara di-*ranking* berdasarkan skala seperti pada tabel 2.11.

MOS kemudian dihitung berdasarkan nilai rata-rata dari tes terhadap subyek tersebut. Dengan menggunakan skala, skor rata-rata 4 keatas dinamakan *toll-quality*. MOS adalah tes yang relevan, karena manusialah pengguna layanan suara tersebut dan perhitungan opini berdasarkan penilaian manusia. Bagaimanapun, tes subyektif yang melibatkan subyek manusia dapat membuang-buang waktu dalam pelaksanaannya.

Tabel 2.12 *Mean Opinion Score (MOS)*

MOS	Kualitas
5	Sangat Bagus
4	Bagus
3	Rata-rata
2	Buruk
1	Sangat Buruk

Sumber: www.wikipedia.com

2.3.7.2 E-Model

Uji kualitas suara selama ini bersifat subyektif, yaitu mengangkat telepon dan mendengarkan kualitas suaranya. Pengukuran kualitas suara secara subyektif yang populer adalah MOS yang dideskripsikan dalam rekomendasi ITU-T P.800. Namun, meminta orang untuk mendengar panggilan telepon secara berulang dapat menjadi sulit. Beberapa kemajuan telah dibuat dalam mengadakan pengukuran kualitas suara secara obyektif, diantaranya adalah:

- PSQM (ITU P.861)/ PSQM+ → Perceptual Speech Quality Measure
- MNB (ITU P.861) → Measuring Normalized Blocks
- PESQ (ITU P.862) → Perceptual Evaluation of Speech Quality
- PAMS (British Telecom) → Perceptual Analysis Measurement System
- The E-Model (ITU G.107) → ETSI Computation Model

PSQM, PSQM+, MNB, dan PESQ adalah bagian dari kemajuan modifikasi algoritma yang dimulai oleh ITU-T P.861. British Telecom mengembangkan PAMS, yang mirip dengan PSQM. Pengukuran PSQM dan PAMS mengirim sinyal referensi melalui jaringan telepon dan membandingkan

sinyal tersebut dengan sinyal yang diterima di sisi penerima, menggunakan algoritma pengolahan sinyal digital.

Namun, beberapa pendekatan tersebut tidak terlalu cocok untuk kualitas suara pada jaringan data. Model yang digunakan tidak berdasar pada sifat-sifat jaringan data, sehingga tidak sesuai dengan sifat-sifat jaringan, seperti *delay, jitter*, dan *packet loss*.

Dalam jaringan VoIP, kualitas transmisi memainkan peran yang sangat penting dalam menentukan kualitas suara. Kualitas transmisi meliputi *packet loss*, *delay*, dan *jitter*. Pendekatan lain dalam pengujian kualitas suara adalah mengukur langsung kualitas transmisi tersebut dan kemudian memprediksi kualitas suara dari beberapa parameter gangguan tersebut. E-Model dikenalkan oleh International Telecommunication Union melalui ITU-T Recommendation G.107 pada Maret 2005. E-Model pada awalnya dikembangkan sebagai piranti untuk perencanaan jaringan, namun saat ini telah digunakan secara luas untuk memprediksi dan menganalisis kualitas suara pada aplikasi VoIP. E-Model didasarkan pada konsep yang dikembangkan oleh J. Allnatt pada tahun 1975, yaitu *"Psychological factors on the psychological scale are additive"*. Konsep ini digunakan untuk mendeskripsikan efek-efek dari berbagai gangguan *(impairment)* yang terjadi secara simultan di dalam suatu sambungan komunikasi. E-Model termasuk tes pasif. Persamaan dasar E-Model adalah adalah sebagai berikut, dengan semua besaran tanpa satuan (Boutremans, 2003:20):

$$R = R_o - I_s - I_d - Ief + A \qquad (2\text{-}20)$$

dengan:

R = nilai factor R

R_o = penjumlahan dari efek *noise,* seperti *noise* ruangan dan *noise* rangkaian

I_s = penjumlahan dari perusakan transmisi secara simultan, seperti *distorsi quantizing* (level keras sinyal yang tidak sesuai, gangguan kuantisasi, dan level bunyi suara sampling yang tidak sesuai

I_d = *error* yang berhubungan dengan *mouth-to-ear delay* pada jalur hubungan

Ief = *error* yang disebabkan oleh peralatan, termasuk efek *packet loss*

A = *Advantage factor*

Advantage factor digunakan jika kualitas yang diinginkan pengguna lebih rendah, misalkan pengguna telepon seluler akan mentolerir kualitas suara yang lebih rendah. Sehingga pada perencanaan jaringan seluler *advantage factor* diberi nilai 10. Untuk penggunaan pada jaringan IP belum ada persetujuan tentang nilai faktor ini, untuk itu pada perencanaan jaringan IP besarnya *advantage factor* adalah 0 (Boutremans, 2003: 20).

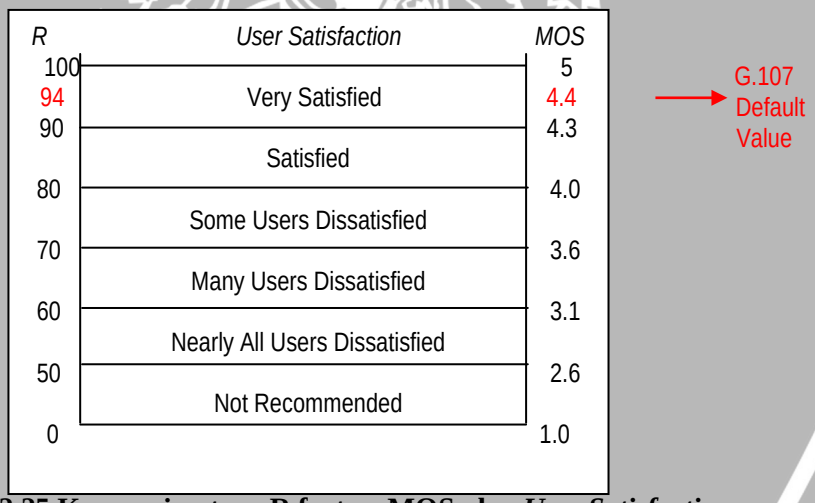
Setelah faktor-faktor gangguan transmisi jaringan IP telah dihitung, E-Model dapat digunakan untuk menghitung *transmission rating factor (R)*. R dapat ditransformasikan ke MOS menggunakan persamaan berikut (Jiang, 2002: 3):

Untuk R <= 0 $MOS = 1$

Untuk 0 <R<100 $MOS = 1 + 0{,}035R + 7.10^{-6}R(R\text{-}60)(100\text{-}R)$ (2-21)

Untuk R >= 100 $MOS = 4{,}5$

R factor mempunyai jarak nilai antara 100 (sempurna) sampai 0 (jelek). MOS mempunyai jarak nilai dari 5 sampai 1. Hubungan antara R factor, MOS, dan *user satisfaction* ditunjukkan pada gambar 2.24.



Gambar 2.25 Konversi antara R factor, MOS, dan *User Satisfaction*
Sumber : IXIA Team, op.cit.

Tabel 2.13 Opini Pengguna dalam Faktor R dan MOS

Opini Pengguna	Faktor R	Nilai MOS
Maksimum yang didapat G.711	93	4.4
Sangat memuaskan	90-100	4.3-5.0
Memuaskan	80-90	4.0-4.3
Baik	70-80	3.6-4.0
Banyak yang tidak puas	60-70	3.1-3.6
Buruk	50-60	2.6-3.1
Tidak direkomendasikan	0-50	1.0-2.6

Sumber: Onno, 2007: 21

Persamaan 2-20 dapat direduksi menggunakan nilai *default* untuk R_o dan I_s sesuai dengan E-Model rancangan ITU-T, menjadi persamaan:

$$R = 94{,}2 - I_d(Ta) - I_{ef}(codec,loss) \quad (2\text{-}22)$$

I_d(Ta) = *Absolute one-way delay*

I_{ef}(codec) = *Codec type impairment* (gangguan karena *codec*)

I_{ef}(loss) = *Packet loss impairment* (gangguan karena *packet loss*)

a) Kerusakan oleh *delay* (I_d)

Berdasarkan nilai standar pada ITU-T G. 107 yang digunakan untuk semua parameter pada persamaan I_d, maka besarnya kerusakan oleh *delay* dapat ditentukan oleh persamaan (Boutremans, 2003: 125):

$$I_d = 0{,}024d + 0{,}11(d\text{-}177{,}3)H(d\text{-}177{,}3) \quad (2\text{-}23)$$

dengan:

d = *delay* satu arah(ms)

dan $H(x)$ fungsi Heavyside: $H(x) = 0$ jika $x < 0$

$H(x) = 1$ jika $x \geq 0$

b) Kerusakan oleh peralatan (I_{ef})

Sampai saat ini belum ada rumusan analitis untuk faktor perusakan oleh peralatan *(equipment impairment factor)*. Untuk mendapatkan harga I_{ef}, harus dilakukan pengukuran subyektif kualitas suara untuk tiap *codec* yang berbeda-beda, dan kondisi operasi yang berbeda-beda pula. Pada Appendix I standar ITU-T G.113 diberikan nilai I_{ef} untuk beberapa jenis *codec* sebagai fungsi dari rata-rata *packet loss.* Dengan mem-*plot* harga-harga I_{ef} untuk tiap *packet loss* dari kurva I_{ef} pada standar tersebut diperoleh ekspresi untuk I_{ef} sebagai berikut (Cole and Rosenbluth, 2001: 6):

$$I_{ef} \approx \gamma_1 + \gamma_2 \ln(1+\gamma_3 e) \quad (2\text{-}24)$$

dengan:

γ_n = konstanta *fitting parameter*, tergantung dari *codec* yang digunakan

e = adalah probabilitas *packet loss* total

Konstanta *fitting parameter* untuk *codec* G.711, dengan panjang *frame* 20 ms adalah (Zhou, 2006: 3):

$\gamma_1 = 0$ $\gamma_2 = 30$ $\gamma_3 = 15$

BAB III
METODOLOGI PENELITIAN

Pada bab ini diuraikan metode yang dilakukan dalam perancangan sistem VoIP melalui VPN. Perancangan ini mengacu pada rumusan masalah yang telah dibuat sebelumnya. Metodologi yang digunakan secara umum meliputi pengumpulan data berupa studi literatur, desain dan implementasi sistem, pengambilan data, analisis data dan pembahasan, serta pengambilan kesimpulan dan saran.

3.1 Studi Literatur

Sebagai bentuk kajian awal metode penelitian dalam penulisan ini agar dapat memperoleh hasil yang optimal. Studi literatur dilakukan untuk mendapatkan pengetahuan dasar tentang segala sesuatu yang mendukung perancangan sistem VoIP – VPN untuk diketahui performansinya. Dalam pembuatan skripsi ini diambil dari buku-buku maupun internet untuk mengetahui konsep dasar dari sistem VoIP dan VPN kerja serta teori yang menunjang.

3.2 Implementasi Sistem

Studi literatur dipelajari kemudian dari permasalahan yang ada dibuat desain dan implementasi sistem, yang meliputi perencanaan arsitektur sistem, konfigurasi *server*, dan konfigurasi *client* .

- Perencanaan Arsitektur Sistem VoIP-VPN

Pada sistem dibutuhkan empat buah komputer. Satu komputer berfungsi sebagai *server*, satu komputer berfungsi sebagai *router*, dan dua komputer sebagai *client*. *Server* menggunakan sistem operasi Ubuntu 8.04 dengan aplikasi utama Asterisk VoIP Server dan OpenVPN Server. *Router* dan *Client* menggunakan sistem operasi MS Windows XP dengan aplikasi utama X-Lite VoIP Client dan OpenVPN Client.

- Konfigurasi *Server*

Konfigurasi server meliputi konfigurasi *ethernet card*, instalasi Asterisk VoIP Server, dan konfigurasi OpenVPN Server.

- Konfigurasi *Client*

Konfigurasi *client* meliputi instalasi X-Lite dan instalasi OpenVPN Client.

3.3 Pengujian dan Pengambilan Data

Pengujian dilakukan pada sistem VoIP yang melalui VPN dan Non-VPN Dengan demikian dapat diketahui perbedaan dari segi keamanan dari komunikasi VoIP melalui VPN dan tanpa. Pengujian sistem meliputi uji koneksi dari *client* ke *server*, dari *router* ke *server*, dari *router* ke *client*, dan dari *client* ke *client* serta uji panggilan dari Client 1 ke Client 2.

Data dalam skripsi ini berupa data primer dan data sekunder. Data primer didapatkan dari pengujian implementasi sistem, sedangkan data sekunder untuk perhitungan parameter secara teoritis. Beberapa batasan dalam pengambilan data secara pengukuran:

- Pengambilan data dilakukan saat Client 1 melakukan panggilan ke Client 2.
- Pada masing-masing konfigurasi (Non-VPN, VPN *less secure*, dan VPN *most secure*) dilakukan pengambilan data sebanyak 10 kali, selama masing-masing dua menit.
- Pengambilan data menggunakan program aplikasi *wireshark*.
- Parameter-parameter yang diamati, antara lain dari pengujian *(throughput, delay, jitter,* dan *packet loss)* dan perhitungan (*bandwidth* yang dibutuhkan, *delay end to end*, dan kualitas suara).
- Pengukuran kualitas suara dilakukan dengan dua metode, yaitu *Polling* dan *E-Model* (ITU-T G.107).

Data sekunder diperoleh dari buku referensi, jurnal, skripsi, dan standarisasi dari ITU serta IETF. Data-data tersebut antara lain:

- *Codec bitrate* untuk G.711 adalah 64 Kbps dengan *frame time* 20 ms.
- Panjang *header* IP = 20 byte.
- Panjang *header* UDP = 8 byte.
- Panjang *header* RTP = 12 byte.
- Panjang *header ethernet* = 14 byte.

3.4 Analisis Data

Analisis data dilakukan dengan mengamati parameter-parameter performansi yang diperoleh dari proses pengukuran dan perhitungan secara teoritis.

- Analisis kuantitatif

Analisis data kuantitatif dilakukan melalui perhitungan matematis sesuai dengan persamaan-persamaan yang dicantumkan dalam dasar teori. Untuk selanjutnya diamati sudah sesuai belum dengan nilai yang direkomendasikan ITU-T G.1010, tentang kategori QoS Multimedia.

Analisis yang dilakukan yaitu:

- ✓ Analisis *delay end-to-end*
- ✓ Analisis *throughput*
- ✓ Analisis *jitter*
- ✓ Analisis *packet loss*
- ✓ Analisis *bandwidth per call*
- ✓ Analisis kualitas suara

- Analisis kualitatif

Dengan perhitungan yang dilakukan pada analisis kuantitatif, maka dapat dibuat tabel hasil perhitungan dan grafik karakteristik. Tabel dan grafik yang diperoleh kemudian diamati dan dianalisis untuk menarik suatu kesimpulan.

3.5 Pengambilan Kesimpulan dan Saran

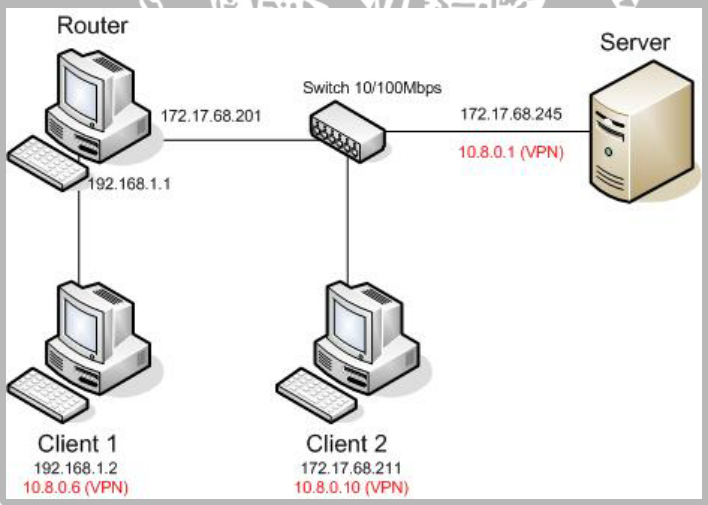
Kesimpulan didapat berdasarkan dari hasil analisis data beserta pembahasannya sesuai dengan tujuan dan rumusan masalah. Saran diberikan setelah melihat adanya kekurangan dalam sistem yang telah dibuat, dengan harapan agar nantinya alat ini dapat dikembangkan lebih baik.

BAB IV
IMPLEMENTASI SISTEM

Bab ini akan menguraikan tentang penerapan VoIP dengan SIP melalui VPN yang meliputi perencanaan sistem yang digunakan, baik *server* maupun *client*. Dimulai dengan perencanaan topologi, persiapan *software* dan *hardware* yang diperlukan, instalasi, konfigurasi, serta persiapan pengujian dan pengambilan data.

4.1 Perencanaan Arsitektur Sistem VoIP-VPN

Perancangan dilakukan di TPT-FT Universitas Brawijaya dengan membuat suatu jaringan VPN sederhana, dimana sistem yang digunakan membutuhkan sebuah komputer yang difungsikan sebagai VPN *server*, sebuah komputer yang difungsikan sebagai *router/ intercept*, dan dua buah komputer sebagai VPN *client.* Arsitektur sistem dirancang seperti pada gambar 4.1. Hub pada arsitektur tersebut menggunakan *switch* 3COM dengan kapasitas *bandwidth* 100 Mbps menggunakan kabel UTP dengan panjang rata-rata 12m.



Gambar 4.1 Arsitektur Sistem VoIP-VPN
Sumber: Perancangan

Komputer	Username + nomor VoIP	IP + netmask + gateway non-VPN	IP + netmask + gateway VPN
Client 1	Client 1 101	192.168.1.2 255.255.255.0 172.17.68.245	10.8.0.6 255.255.255.0 10.8.0.1
Client 2	Client 2 102	172.17.68.211 255.255.255.0 172.17.68.245	10.8.0.10 255.255.255.0 10.8.0.1

Komputer-komputer tersebut masing-masing memiliki NIC dengan kemampuan *bandwidth* 10/100 Mbps dan memiliki spesifikasi sebagai berikut:

- *Server* : Intel Pentium III 2.66 GHz, 256MB RAM
- *Client* 1 : Intel Pentium IV 1.90 GHz, 256MB RAM
- *Client* 2 : Intel Pentium III 650 MHz, 512MB RAM
- *Router* : Intel Pentium III 350 MHz, 64 MB RAM

Perangkat lunak PC *Server* yang digunakan adalah sistem operasi *Ubuntu* 8.04 beserta aplikasi utama, yaitu *Asterisk* VoIP *Server* dan *Open*VPN *Server*. Dipilih *Ubuntu* 8.04 karena memberikan kemudahan dalam penginstalan dan merupakan versi LTS yang memiliki dukungan terhadap pengembangan paket baru, *update* paket, *bug*, atau *security update* selama tiga tahun untuk versi desktop. Pada PC *Router* dan *Client* digunakan sistem operasi *Windows XP* beserta aplikasi utama, yaitu *X-Lite* VoIP *Client* dan *Open*VPN *Client*. Dipilih *Windows XP* karena tergolong stabil dan memudahkan instalasi VoIP *Client* yang membutuhkan kompatibilitas tinggi. *Software* lain yang digunakan adalah *Wireshark* yang diinstal pada PC *Client* untuk membaca paket-paket data yang lewat pada jaringan. Sehingga *Wireshark* berperan dalam proses pengambilan data. Perangkat lunak yang digunakan pada sistem ini bersifat *freeware* yang dapat diperoleh dari situs produsennya secara cuma-cuma.

4.2 Konfigurasi *Server*

Implementasi ini dimulai dengan mengkonfigurasikan *ethernet card*, kemudian menginstal *Asterisk* VoIP *Server* dan *Open*VPN *Server*. Setelah itu melakukan sertifikasi, kemudian mengkonfigurasikan *Open*VPN apakah dapat berjalan dengan baik. Langkah-langkah instalasi dan konfigurasi pada *server* seperti pada Lampiran.

4.2.1 Konfigurasi *Ethernet Card*

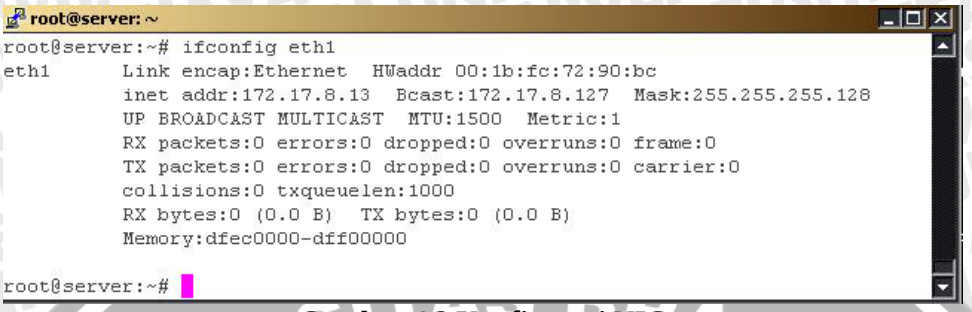
NIC yang digunakan pada *server* adalah Intel NIC *onboard* dengan kecepatan data 100Mbps *(Fast Ethernet)*, diwakili dengan nama eth1. Konfigurasi NIC pada *server* dilakukan dengan men-*setting* IP *Address server*. *File* konfigurasi NIC terletak di **/etc/network/interfaces** :

```
auto lo
iface lo inet loopback

auto eth1
```

```
iface eth1 inet static
address 192.168.1.2
netmask 255.255.255.0
gateway 172.17.68.245
```

Berikut merupakan gambar konfigurasi NIC :



Gambar 4.2 Konfigurasi NIC
Sumber: Perancangan

Agar konfigurasi tersebut berlaku, maka modul jaringan pada *server* perlu di-*restart*. Untuk itu diberikan perintah sebagai berikut :

```
root@server:~# /etc/init.d/networking restart
```

Agar daftar aplikasi yang dapat diinstall pada *server* ter-*update*, maka diberikan perintah sebagai berikut :

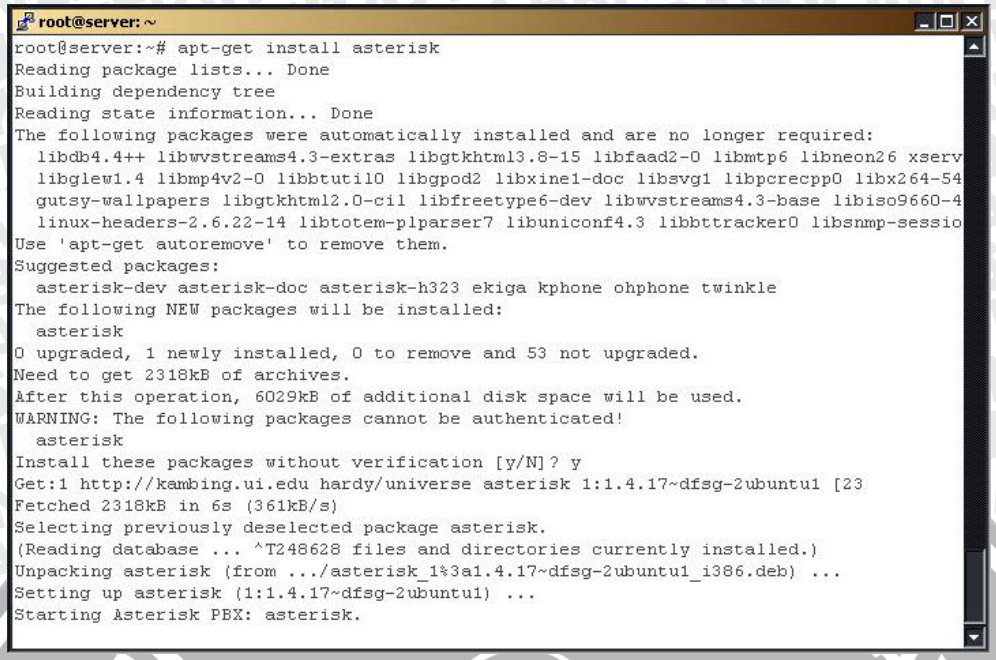
```
root@server:~# apt-get update
root@server:~# apt-get upgrade
```

4.2.2 Instalasi dan Konfigurasi *Asterisk* VoIP *Server*

Perintah untuk menginstal *Asterisk* VoIP *Server* adalah sebagai berikut :

```
root@server:~# apt-get install asterisk
```

Gambar 4.3 memperlihatkan proses instalasi *Asterisk* pada *server*.



Gambar 4.3 Proses Instalasi *Asterisk*
Sumber: Perancangan

File untuk mengatur konfigurasi *Asterisk* dan menampung data *username* serta *password* yang nantinya akan digunakan oleh *user* terletak di **/etc/asterisk/sip.conf.**

Berikut konfigurasi sederhana pada sistem Non-VPN :

```
[general]
port=5060
bindaddr=172.17.68.245
context=extension

[101]
type=friend
context=phone
username=user1
secret=1234
host=dynamic

[102]
type=friend
context=phone
username=user2
secret=1234
host=dynamic
```

Konfigurasi pada sistem yang melalui VPN :

```
[general]
port=5060
bindaddr=10.8.0.1
context=extension
```

```
[101]
type=friend
context=phone
username=user1
secret=1234
host=dynamic

[102]
type=friend
context=phone
username=user2
secret=1234
host=dynamic
```

Untuk menetapkan nomor VoIP kepada tiap *client* digunakan *file* extensions.conf. *File* konfigurasi extensions.conf terletak di **/etc/asterisk/extensions.conf.**

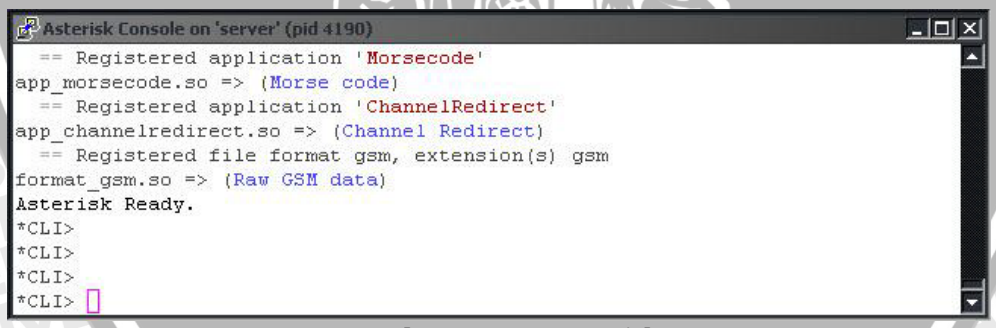
```
[extension]
[phone]
exten => _10x,1,Dial(SIP/${EXTEN},20,rt)
exten => _10x,2,HangUp
```

Dalam skripsi ini digunakan dua SIP *account*, yaitu *Client* 1(101) dan *Client* 2(102).

Untuk menjalankan *asterisk* digunakan perintah sebagai berikut :

```
root@server:~# asterisk –vvvvvc
```

Jika *asterisk* dapat berjalan dengan benar maka akan masuk ke CLI *Asterisk* seperti pada gambar 4.4 di bawah ini.



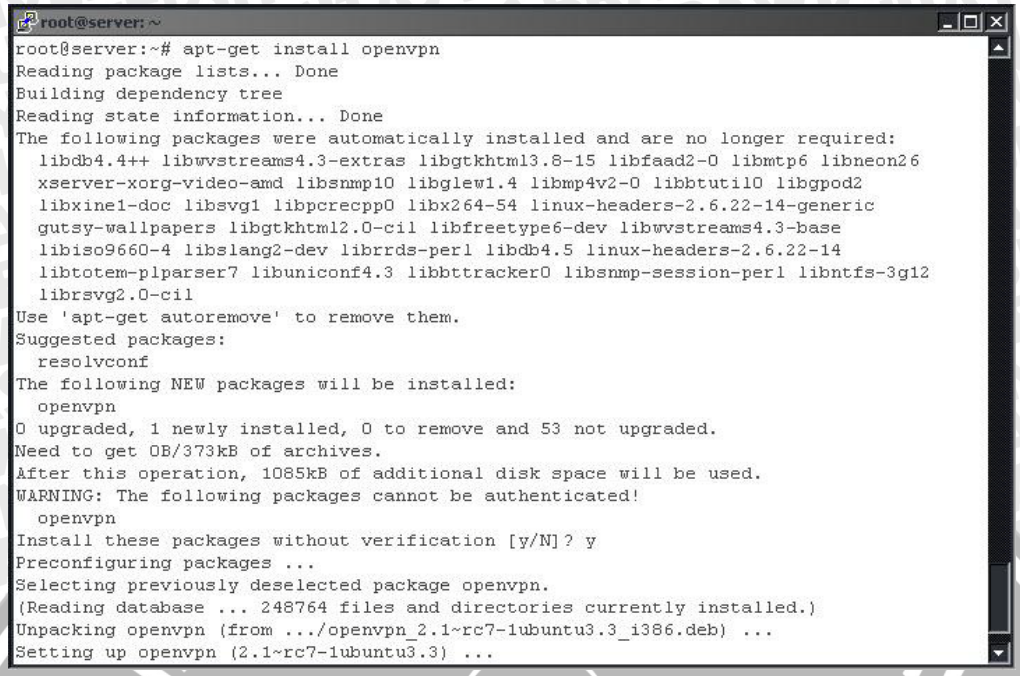
Gambar 4.4 CLI *Asterisk*
Sumber: Perancangan

4.2.3 Instalasi *Open*VPN *Server*

Instalasi *Open*VPN *Server* menggunakan perintah sebagai berikut :

```
root@server:~# apt-get install openvpn
```

Gambar 4.5 memperlihatkan proses instalasi *Open*VPN pada *server*.



Gambar 4.5 Instalasi *Open*VPN
Sumber: Perancangan

Untuk konfigurasi *Open*VPN yang pertama harus dilakukan adalah men-*setting* beberapa *option* yang ada pada *file* ***vars*** yang terletak di **/usr/share/doc/openvpn/examples/easy-rsa/2.0/.** Proses pembuatan setifikat juga dilakukan di direktori ini. Isi *file vars* yang diubah adalah sebagai berikut :

```
export KEY_COUNTRY=ID
export KEY_PROVINCE=JATIM
export KEY_CITY=MALANG
export KEY_ORG="TPTIFT"
export KEY_EMAIL="teknik@brawijaya.ac.id"
```

Setelah mengedit beberapa *option* pada *file vars*, maka proses pembuatan sertifikat bisa dilakukan dengan perintah sebagai berikut :

```
root@server:/etc/openvpn/key/# source .vars
root@server:/etc/openvpn/key/# ./clean-all
root@server:/etc/openvpn/key/# ./build-dh
root@server:/etc/openvpn/key/# ./pkitool --initca
root@server:/etc/openvpn/key/# ./pkitool –server
root@server:/etc/openvpn/key/# ./pkitool client1
root@server:/etc/openvpn/key/# ./pkitool client2
```

Proses ini akan menghasilkan *file* ca.crt, ca.key, server.crt, server.key, client1.crt, client1.key, client2.crt, client2.key, dan file dh1024.pem. Selanjutnya file ca.crt, ca.key, server.crt, server.key, dan dh1024.pem di-*copy* ke direktori **/etc/openvpn/keys.** Sedangkan untuk *file* lainnya didistribusikan ke masing-masing *client.*

File konfigurasi yang digunakan, yakni openvpn.conf yang akan diletakkan pada direktori **/etc/openvpn** adalah sebagai berikut :

```
port 1194
proto udp
dev tun
ca /etc/openvpn/key/keys/ca.crt
cert /etc/openvpn/key/keys/server.crt
key /etc/openvpn/key/keys/server.key  # This file should be kept secret
dh /etc/openvpn/key/keys/dh1024.pem
server 10.8.0.1 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
comp-lzo
persist-key
persist-tun
client-to-client
status openvpn-status.log
verb 3
```

Port yang digunakan adalah 1194 dengan IP Tun 10.8.0.1 pada *server*.

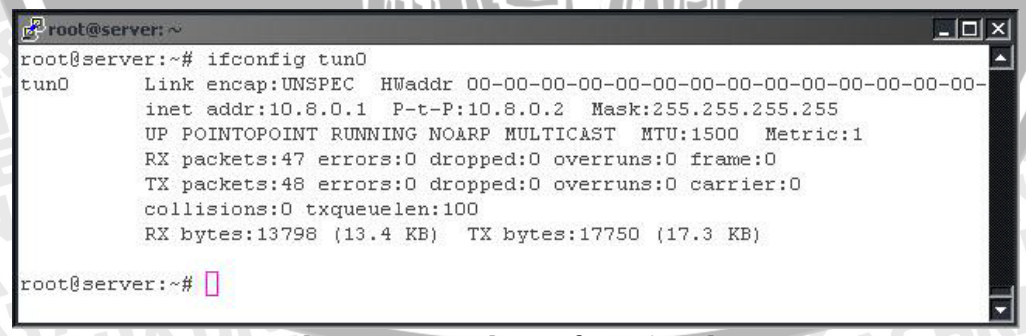
Perintah menjalankan *Open*VPN *Server* sebagai berikut :

```
root@server: :~# openvpn –config /etc/openvpn/server.conf
```

Sedangkan untuk menghentikan *Open*VPN *Server* digunakan perintah sebagai berikut :

```
root@server: :~# /etc/init.d/openvpn stop
```

Bila konfigurasi tersebut berhasil, maka OpenVPN akan membuat *network interface virtual* berjenis ”tun”, biasanya disebut ”tun0” dan ini dapat diperiksa dengan perintah ”ifconfig tun0”, seperti terlihat pada gambar 4.7.



Gambar 4.6 *Network Interface Virtual ”tun”*
Sumber: Perancangan

4.3 Konfigurasi *Client*

Client dalam skripsi ini menggunakan sistem operasi *Windows* dan aplikasi *softphone X-Lite* yang bersifat *freeware* dan dapat diperoleh dari situs

produsennya, www.xten.com. *X-Lite* yang digunakan versi 3.0 yang berbasis pada protokol SIP dan *Open*VPN GUI untuk *Windows* versi 1.0.3. Langkah-langkah dalam instalasi dan konfigurasi *client* seperti pada Lampiran.

4.3.1 Instalasi dan Konfigurasi *X-Lite*

Insatalasi *X-Lite* sangat mudah, tipikal dengan kemudahan instalasi *software* di *Windows*. Langkah–langkah yang dilakukan untuk menjalankan *X-Lite*, sebagai berikut :

a. Penambahan SIP Account

Pada tampilan *X-Lite* klik kanan, pilih menu *"SIP Account Settings"*, maka akan muncul *"SIP Accounts"*. Klik *"Add"* untuk menambah SIP *account*, kemudian pilih *"Properties"*, yang perlu di-*setting* adalah :

- ***Display Name***

  Nama yang akan muncul pada tampilan *X-Lite*.

- ***User name***

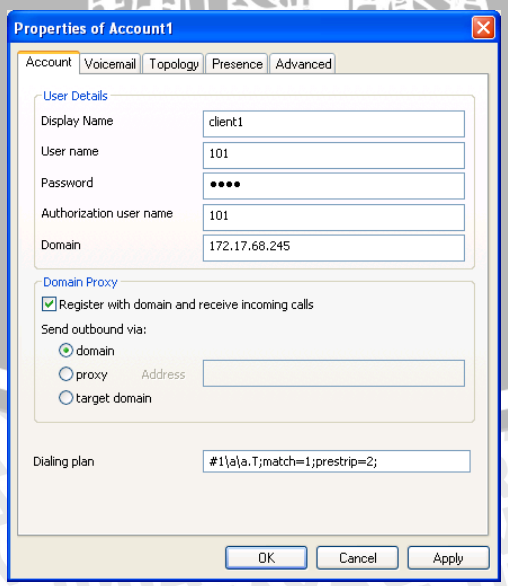
  User name dari SIP *account* pada *server*.

- ***Authorization username***

  Sama seperti *username*.

- ***Domain***

  IP *address server*, yang diisi dengan dua jenis IP *address*, VPN atau Non-VPN. IP harus sesuai dengan *network* yang bersangkutan.



Gambar 4.7 Penambahan SIP *Account X-Lite* pada *Client*
Sumber: Perancangan

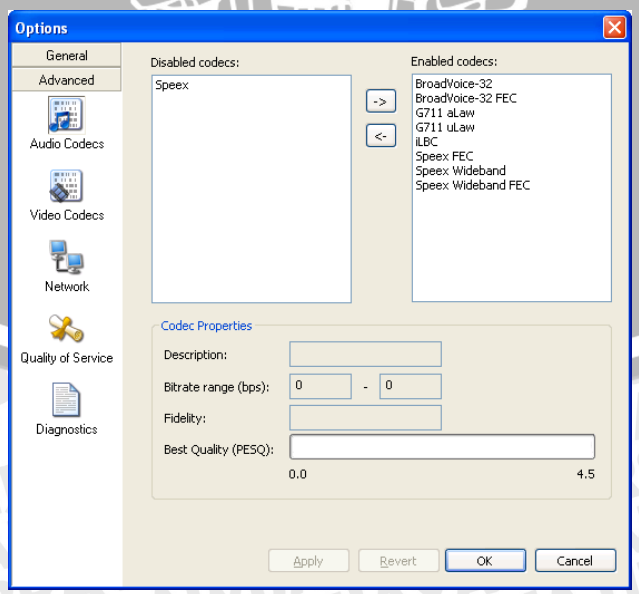
b. Koneksi ke *Server*

Setelah pembuatan SIP *account* selesai, *X-Lite* otomatis akan meregister *account* tersebut ke *server*. Bila berhasil , maka akan muncul pesan *: "Ready, Your username is: <username>"*, seperti terlihat pada gambar 4.10. Sedangkan bila gagal, maka pesan akan berganti dengan : *"Registration error : 403 - Forbidden*". Setelah kedua *client* ter-*register* pada *server*, maka kedua *client* tersebut dapat melakukan panggilan satu dengan lainnya dengan memanggil nomor extensi sesuai dengan yang ada pada file **extensions.conf.**

Gambar 4.8 *Register Account* ke *Server* SIP Sukses
Sumber: Perancangan

c. *Setting codec* yang digunakan

Pada skripsi ini digunakan *codec* G.711 μ-Law, yang ditentukan oleh *client X-Lite* seperti pada gambar berikut :



Gambar 4.9 *Setting Codec* pada *X-Lite*
Sumber: Perancangan

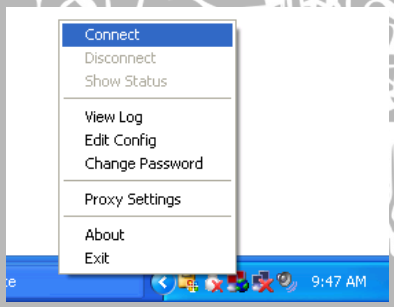
4.3.2 Instalasi dan Konfigurasi *Open*VPN *Client*

*Open*VPN pada *Windows* dapat diperoleh dari situs http://openvpn.se dan bersifat *freeware*. Pada skripsi ini digunakan versi 1.0.3. Proses instalasi juga akan menginstal *network interface virtual*, sama seperti pada *server*.

Setelah instalasi selesai, *copy*-kan *file* ca.crt, client.key, dan client.crt dari *server* ke direktori config di direktori kerja *Open*VPN. Kemudian dibuat *file* client.ovpn pada direktori config, yang berisi sebagai berikut :

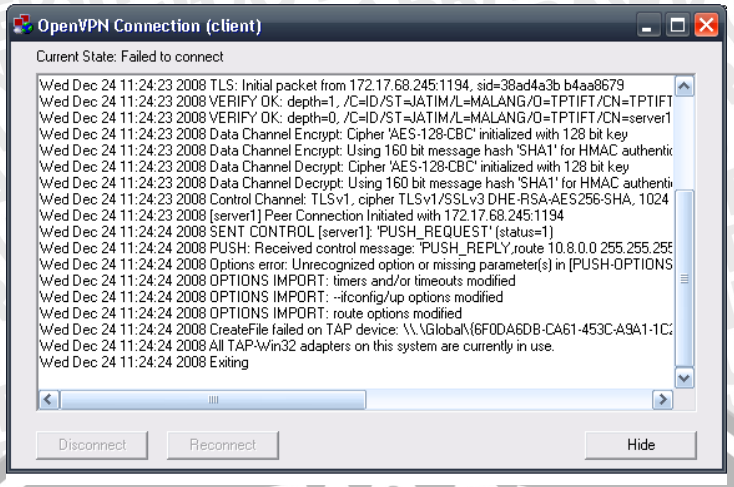
```
port 1194
proto udp
dev tun
ca /etc/openvpn/key/keys/ca.crt
cert /etc/openvpn/key/keys/client.crt
key /etc/openvpn/key/keys/client.key  # This file should be kept secret
dh /etc/openvpn/key/keys/dh1024.pem
server 10.8.0.10 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
comp-lzo
persist-key
persist-tun
client-to-client
status openvpn-status.log
```

Klik kanan icon program pada sistem tray dan pilih "connect" untuk menjalankan program *Open*VPN-GUI.



Gambar 4.10 Menjalankan *Open*VPN-GUI
Sumber: Perancangan

Proses *Open*VPN dijalankan dengan memulai proses negosiasi antara *client* dan *server* dengan saling mempertukarkan masing-masing sertifikat yang dimilikinya untuk kemudian membentuk sebuah *tunnel* diantara *client* dan *server* yang terhubung tersebut.



Gambar 4.11 Koneksi VPN yang berhasil
Sumber: Perancangan

Ketika *client* memberikan *request* untuk melakukan koneksi VPN, maka di kedua sisi dilakukan pembentukan aturan-aturan terhadap *tunnel* yang akan dibangun. Ketika *client* memulai koneksi, maka muncul parameter-parameter sebelum akhirnya terhubung secara VPN. *Security policies* antara VPN *server* dan VPN *client* sudah terbentuk sehingga selanjutnya paket yang akan dikirimkan antara keduanya akan dilewatkan melalui *tunnel* yang terbentuk tersebut.

Bila berhasil, *network interface* yang bersangkutan menjadi aktif dan memperoleh IP Address 10.8.0.6.

BAB V
PENGUJIAN DAN PENGAMBILAN DATA

5.1 Pengujian Sistem

Pengujian meliputi uji koneksi dan uji panggilan. Uji koneksi dilakukan pada sistem tanpa melalui VPN (Non-VPN) dan pada sistem melalui VPN. Pengujian ini untuk mengetahui apakah data yang dilewatkan pada sistem yang melalui VPN sudah aman. Istilah aman dalam hal ini adalah data yang dikirim dari *Client* 1 ke *Client* 2 maupun sebaliknya tidak dapat terbaca oleh *router* yang berfungsi sebagai *intercept.* Uji panggilan dilakukan untuk mengetahui apakah *Client* 1 dan *Client* 2 sudah siap melakukan komunikasi VoIP baik tanpa melalui VPN maupun melalui VPN.

5.1.1 Pengujian Sistem Non-VPN

5.1.1.1 Uji Koneksi

Uji koneksi dilakukan dengan perintah ping antara *Client*, *Router*, dan *Server*. Berikut ini tampilan uji koneksi menggunakan perintah *ping*:

```
C:\Documents and Settings\Alit Mahendra>ping 172.17.68.245

Pinging 172.17.68.245 with 32 bytes of data:

Reply from 172.17.68.245: bytes=32 time<1ms TTL=63
Reply from 172.17.68.245: bytes=32 time<1ms TTL=63
Reply from 172.17.68.245: bytes=32 time<1ms TTL=63
Reply from 172.17.68.245: bytes=32 time<1ms TTL=63

Ping statistics for 172.17.68.245:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.1 Tampilan *Ping* dari *Client* 1 ke *Server* pada sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Alit Mahendra>ping 172.17.68.211

Pinging 172.17.68.211 with 32 bytes of data:

Reply from 172.17.68.211: bytes=32 time<1ms TTL=126
Reply from 172.17.68.211: bytes=32 time<1ms TTL=127
Reply from 172.17.68.211: bytes=32 time<1ms TTL=127
Reply from 172.17.68.211: bytes=32 time<1ms TTL=127

Ping statistics for 172.17.68.211:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.2 Tampilan *Ping* dari *Client* 1 ke *Client* 2 pada sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\drembis>ping 172.17.68.245

Pinging 172.17.68.245 with 32 bytes of data:

Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64

Ping statistics for 172.17.68.245:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.3 Tampilan *Ping* dari *Client* 2 ke *Server* pada sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\drembis>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=127
Reply from 192.168.1.2: bytes=32 time<1ms TTL=127
Reply from 192.168.1.2: bytes=32 time<1ms TTL=127
Reply from 192.168.1.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.4 Tampilan *Ping* dari *Client* 2 ke *Client* 1 pada Sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 172.17.68.245

Pinging 172.17.68.245 with 32 bytes of data:

Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64
Reply from 172.17.68.245: bytes=32 time<1ms TTL=64

Ping statistics for 172.17.68.245:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.5 Tampilan *Ping* dari *Router* ke *Server* pada sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.6 Tampilan *Ping* dari *Router* ke *Client* 1 pada sistem Non-VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 172.17.68.211

Pinging 172.17.68.211 with 32 bytes of data:

Reply from 172.17.68.211: bytes=32 time<1ms TTL=128
Reply from 172.17.68.211: bytes=32 time<1ms TTL=128
Reply from 172.17.68.211: bytes=32 time<1ms TTL=128
Reply from 172.17.68.211: bytes=32 time<1ms TTL=128

Ping statistics for 172.17.68.211:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.7 Tampilan *Ping* dari *Router* ke *Client* 2 pada sistem Non-VPN
Sumber: Hasil Pengujian

Dari tampilan uji koneksi dengan melakukan ping pada sistem Non-VPN memperlihatkan bahwa antara *Client*, *Router*, dan *Server* telah terkoneksi dan dapat melakukan komunikasi VoIP satu sama lain.

5.1.1.2 Uji Panggilan dari *Client* 1 ke *Client* 2

Uji panggilan dari *Client* 1 ke *Client* 2 untuk mengetahui apakah panggilan yang dilakukan sukses atau tidak.

Gambar 5.8 *Client* 2 saat ada panggilan dari *Client* 1 pada sistem Non-VPN
Sumber: Hasil Pengujian

Gambar 5.9 *Client* 2 menerima panggilan dari *Client* 1 pada sistem Non-VPN
Sumber: Hasil Pengujian

Gambar 5.8 dan 5.9 memperlihatkan bahwa komunikasi VoIP antara *Client* 1 dan *Client* 2 dengan *X-Lite* dapat dijalankan.

5.1.2 Pengujian Sistem melalui VPN

5.1.2.1 Uji koneksi

Uji koneksi pada sistem VPN juga dilakukan dengan perintah ping.

```
C:\Documents and Settings\Alit Mahendra>ping 10.8.0.1

Pinging 10.8.0.1 with 32 bytes of data:

Reply from 10.8.0.1: bytes=32 time<1ms TTL=64
Reply from 10.8.0.1: bytes=32 time<1ms TTL=64
Reply from 10.8.0.1: bytes=32 time<1ms TTL=64
Reply from 10.8.0.1: bytes=32 time<1ms TTL=64

Ping statistics for 10.8.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.10 Tampilan *Ping* dari *Client* 1 ke *Server* pada sistem VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Alit Mahendra>ping 10.8.0.10

Pinging 10.8.0.10 with 32 bytes of data:

Reply from 10.8.0.10: bytes=32 time=1ms TTL=128
Reply from 10.8.0.10: bytes=32 time=1ms TTL=128
Reply from 10.8.0.10: bytes=32 time=1ms TTL=128
Reply from 10.8.0.10: bytes=32 time=1ms TTL=128

Ping statistics for 10.8.0.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Gambar 5.11 Tampilan *Ping* dari *Client* 1 ke *Client* 2 pada sistem VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\drembis>ping 10.8.0.1

Pinging 10.8.0.1 with 32 bytes of data:

Reply from 10.8.0.1: bytes=32 time<1ms TTL=64
Reply from 10.8.0.1: bytes=32 time<1ms TTL=64

Ping statistics for 10.8.0.1:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Gambar 5.12 Tampilan *Ping* dari *Client* 2 ke *Server* pada sistem VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 10.8.0.1

Pinging 10.8.0.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.8.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Gambar 5.13 Tampilan *Ping* dari *Router* ke *Server* pada sistem VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 10.8.0.6

Pinging 10.8.0.6 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.8.0.6:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Gambar 5.14 Tampilan *Ping* dari *Router* ke *Client* 1 pada sistem VPN
Sumber: Hasil Pengujian

```
C:\Documents and Settings\Si Hitam>ping 10.8.0.10

Pinging 10.8.0.10 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.8.0.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Gambar 5.15 Tampilan *Ping* dari *Router* ke *Client* 2 pada sistem VPN
Sumber: Hasil Pengujian

Dari tampilan uji koneksi dengan melakukan ping dari *Server* ke *Client* 1 dan *Client* 2 memperlihatkan adanya koneksi sehingga antara *Client* 1 dan *Client* 2 dapat melakukan komunikasi VoIP. Tetapi untuk uji koneksi dengan melakukan ping dari *Router* ke *Server*, *Router* ke *Client* 1, dan *Router* ke *Client* 2 memperlihatkan tidak adanya koneksi. Hal ini menunjukkan bahwa *Router* sebagai *intercept* tidak dapat mengakses segala data yang dikirimkan dari *Client* 1 ke *Client* 2 maupun sebaliknya.

5.1.2.2 Uji Panggilan dari *Client* 1 ke *Client* 2

Uji panggilan dari *Client* 1 ke *Client* 2 untuk mengetahui apakah panggilan yang dilakukan sukses atau tidak.

Gambar 5.16 *Client* 2 pada saat ada Panggilan dari *Client* 1 pada Sistem VPN
Sumber: Hasil Pengujian

Gambar 5.17 *Client* 2 Menerima Panggilan dari *Client* 1 pada Sistem VPN
Sumber: Hasil Pengujian

Gambar 5.16 dan 5.17 memperlihatkan bahwa komunikasi VoIP antara *Client* 1 dan *Client* 2 dengan *X-Lite* pada sistem VPN dapat dijalankan.

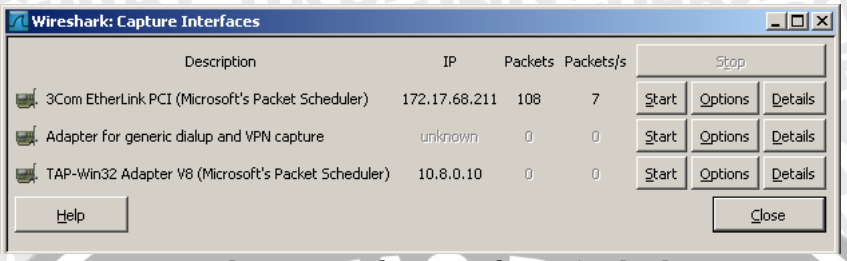
5.2 Pengambilan Data

Pada skripsi ini, dipilih data komunikasi searah, yaitu dari *Client* 1 menuju *Client* 2. Pada masing-masing konfigurasi dilakukan pengambilan data sebanyak 10 kali, selama masing-masing dua menit. Dari sini akan diamati *payload* dan dapat diperoleh data, antara lain *throughput*, *delay*, *jitter*, dan *packet loss*. Suara yang digunakan merupakan hasil rekaman pada *Client* 1, yang kemudian diperdengarkan menggunakan aplikasi Media Player, dan keluaran suaranya di-inputkan untuk *recording*, sehingga *X-Lite* akan menerima suara tersebut seolah-olah dari mikrofon. Spesifikasi rekaman 22,05 kHz 16 bit mono, pembicara perempuan. Kalimat yang dieja "Di depan stadion, tiba-tiba aku teringat akanmu. Dulu pernah kita bertemu di sini". Untuk proses pengambilan data, maka prosedur yang harus dilakukan dijelaskan dengan diagram alir pada Lampiran.

Konfigurasi algoritma *cipher* dalam skripsi ini direncanakan menggunakan jenis yang berbeda-beda. Pada sistem *less secure*, *cipher* menggunakan BF-CBC, sedangkan pada *most secure* menggunakan AES-128-CBC. Pada kedua konfigurasi menggunakan algoritma *tls-cipher* DHE-RSA-AES256-SHA (Markus Feilner, 2006: 143).

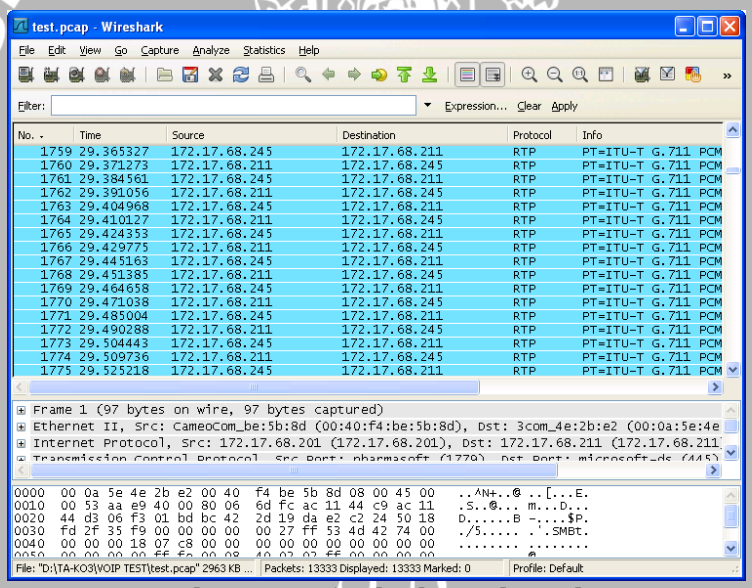
Konfigurasi "*less secure*" menggunakan enkripsi dan autentikasi yang lebih mudah dan menggunakan sedikit *overhead* pada enkapsulasi paket-paketnya, menjadikan proses *tunneling* menjadi relatif cepat. Sebaliknya konfigurasi "*most secure*" menggunakan enkripsi dan autentikasi yang lebih rumit, sehingga dapat menambah *overhead* pada paket-paketnya, menjadikan proses relatif lebih lambat.

Wireshark mampu membaca paket-paket data yang lewat pada jaringan dan menganalisanya. SIP merupakan salah satu protokol yang didukung oleh *Wireshark*. Untuk menggunakan *Wireshark*, pilih menu *"Capture"* lalu *"Interface"* hingga muncul daftar *Network Interface* seperti pada Gambar 5.18.



Gambar 5.18 Daftar *Interface Wireshark*
Sumber: Hasil Pengujian

Untuk memulai menjalankan program *Wireshark*, pilih *"Start"* sesuai dengan *interface* yang dikehendaki, yaitu Non-VPN atau VPN. Selanjutnya *Wireshark* akan membaca data seperti pada gambar 5.19.

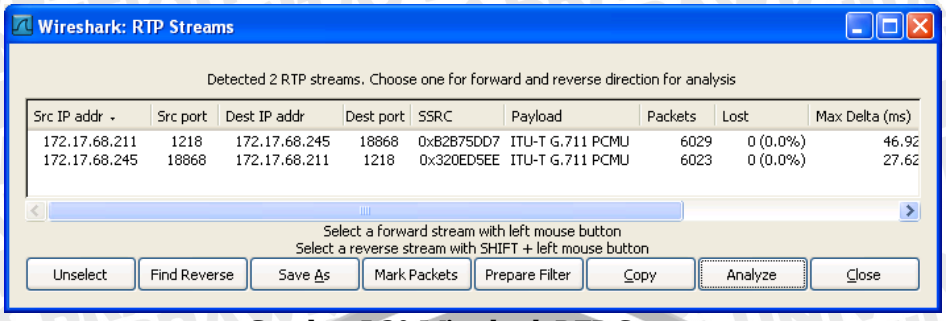


Gambar 5.19 *Wireshark* membaca data
Sumber: Hasil Pengujian

Untuk menghentikan pilih *"Stop"*. Kemudian data disimpan sebagai *file* dengan ekstensi *.pcap.

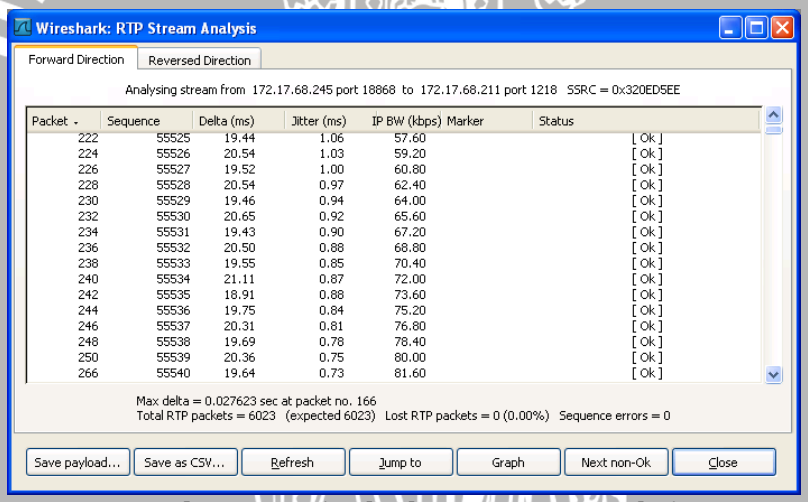
5.2.1 *Throughput*

Setelah data diperoleh, pilih *"Statistics"*, *"RTP"*, *"Show All Streams"* hingga nampak data *throughput* (kbps) seperti gambar 5.20 berikut :



Gambar 5.20 *Wireshark RTP Stream*
Sumber: Hasil Pengujian

Kemudian pilih trafiknya dan pilih *"Analyze"*, hingga muncul tampilan seperti pada gambar 5.21. Nampak data *throughput* yang diinginkan, untuk kemudian dihitung nilai rata–ratanya. Caranya, pilih *"Save as CSV"*, lalu simpan data tersebut sebagai *file* CSV. *File* ini lalu dibuka menggunakan aplikasi *Spreadsheet* seperti *MS Excel*, lalu semua data *throughput* tersebut dihitung nilai rata–ratanya.



Gambar 5.21 *Wireshark RTP Stream Analysis*
Sumber: Hasil Pengujian

Tabel 5.1 Data *Throughput (kbps)*

Data ke-	Non-VPN (kbps)	VPN-less (kbps)	VPN-most (kbps)
1	80,38	80,24	79,82
2	80,36	80,13	79,91
3	80,05	80,24	79,93
4	80,35	80,35	80,14
5	80,39	80,22	79,49
6	80,51	80,25	80,14
7	80,51	80,53	80,11
8	80,18	80,25	80,18
9	80,39	80,35	80,14
10	80,32	80,29	79,89
Rata - rata	**80,34**	**80,29**	**79,98**

Sumber: Hasil Pengujian

5.2.2 *Jitter*

Sama halnya dengan pengambilan data *throughput*, data *jitter* diperoleh dari tampilan seperti terlihat pada gambar 5.21. Rata–rata nilai *jitter* diperoleh dari *file* CSV yang sama seperti yang digunakan untuk pengambilan data *throughput*.

Tabel 5.2 Data *Jitter (ms)*

Data ke -	Non-VPN (ms)	VPN-less (ms)	VPN-most (ms)
1	3,90	5,88	9,79
2	0,54	3,07	10,70
3	0,56	5,14	10,96
4	4,76	7,30	8,94
5	0,83	10,08	7,72
6	0,91	8,74	9,33
7	0,67	10,54	11,67
8	0,54	11,33	8,76
9	2,89	10,11	8,13
10	0,97	6,58	8,26
Rata - rata	**1,66**	**7,88**	**9,43**

Sumber: Hasil Pengujian

5.2.3 *Packet Loss*

Demikian pula dengan data *packet loss* juga diperoleh dari tampilan seperti terlihat pada gambar 5.21. Pada *file* tersebut sudah tercantum *packet loss* berdasarkan *packet* yang hilang.

Tabel 5.3 Data *Packet Loss* (packets)

Data ke -	Non-VPN (packets)	VPN-less (packets)	VPN-most (packets)
1	5	7	15
2	7	6	7
3	1	10	25
4	11	9	9
5	11	5	8
6	5	4	24
7	2	7	5
8	9	11	10
9	10	8	6
10	7	2	11
Rata - rata	**7**	**8**	**12**

Sumber: Hasil Pengujian

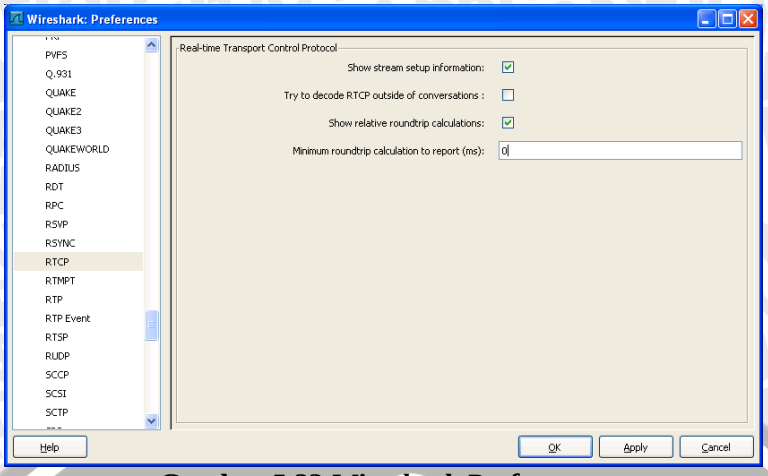
Tabel 5.4 Data *Packet Loss* (%)

Data ke -	Non-VPN (%)	VPN-less (%)	VPN-most (%)
1	0,08	0,11	0,25
2	0,12	0,10	0,12
3	0,02	0,17	0,42
4	0,18	0,15	0,15
5	0,19	0,08	0,13
6	0,08	0,07	0,40
7	0,03	0,12	0,08
8	0,15	0,18	0,17
9	0,17	0,13	0,10
10	0,12	0,20	0,18
Rata - rata	**0,11**	**0,13**	**0,20**

Sumber: Hasil Pengujian

5.2.4 *Delay*

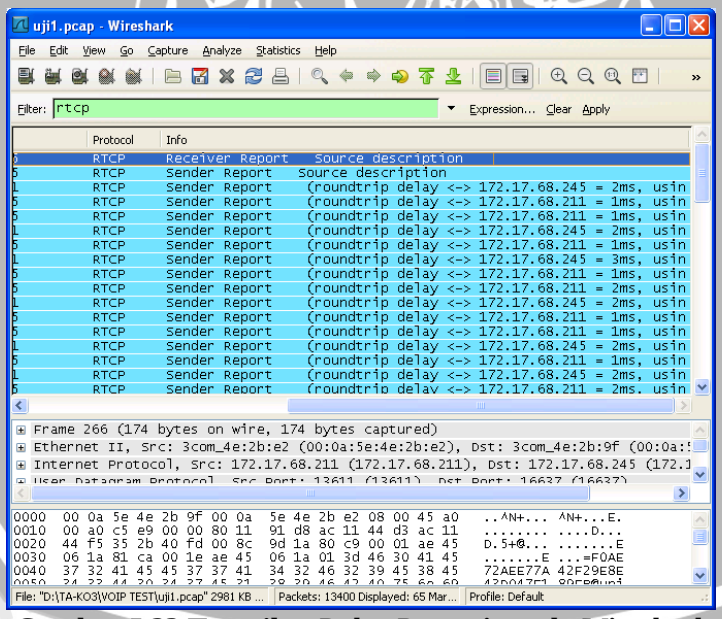
Untuk memperoleh data *delay*, *Wireshark* perlu di-*setting* terlebih dahulu agar dapat menampilkan data tersebut. Pilih *"Edit"*, lalu *"Preferences"*. Kemudian pilihan *"Protocols"*, pilih *"RTCP"* seperti pada gambar 5.22.



Gambar 5.22 *Wireshark Preferences*
Sumber: Hasil Pengujian

Pilih *"Show relative rountrip calculations"*, dan ubah nilainya menjadi 0. Dengan begini *Wireshark* akan menghitung *delay* berdasarkan informasi dari paket RTCP yang dihasilkan aplikasi VoIP.

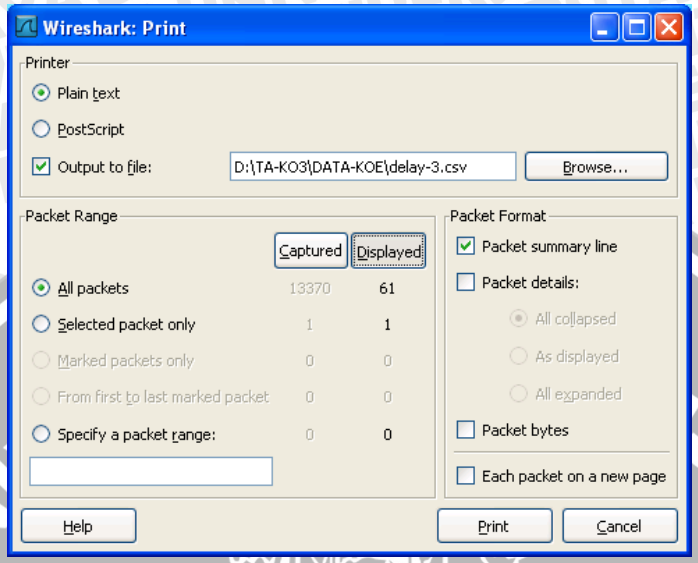
Setelah itu, tampilan paket–paket pada *Wireshark* perlu di-*setting* agar hanya menampilkan paket RTCP saja. Untuk itu ketik *"RTCP"* pada *"Filter"*, dan pilih *"Apply"*. Hasilnya seperti tampak pada gambar 5.23.



Gambar 5.23 Tampilan *Delay Rountrip* pada *Wireshark*
Sumber: Hasil Pengujian

Untuk memperoleh nilai rata–rata, data tersebut perlu dikonversi menjadi file CSV. Klik kanan pada paket–paket tersebut, pilih *"Print"* hingga tampak seperti gambar 5.24. Pilih *"Output to File"*, tentukan nama *file*-nya. Lalu pilih *"All Packets"* dan *"Displayed"*, lalu hilangkan pilihan *"Packet Details"*. Setelah

itu pilih *"Print"*, maka *file* tersebut akan terbentuk dan berisi tampilan paket–paket RTCP tersebut lengkap dengan *delay*. Untuk menghitung nilai rata–rata *delay*, buka *file* CSV tersebut dengan *MS Excel*.



Gambar 5.24 *Wireshark Print* untuk Menghitung *Delay*
Sumber: Hasil Pengujian

Tabel 5.5 Data *Delay (ms)*

Data ke -	Non-VPN (ms)	VPN-less (ms)	VPN-most (ms)
1	1,68	12,91	32,17
2	3,32	8,53	30,20
3	1,83	8,86	28,91
4	2,00	21,97	25,39
5	4,69	22.17	12,37
6	2,25	23,95	17,58
7	2,04	15,54	18,12
8	1,70	13,50	20,01
9	2,08	11,52	22,15
10	1,95	11,10	22,27
Rata - rata	**2,35**	**15,01**	**22,92**

Sumber: Hasil Pengujian

5.2.5 Kualitas Suara

Data kualitas suara diperoleh dari *polling* terhadap 50 orang dengan menggunakan metode MOS. *Polling* dilakukan dengan memperdengarkan rekaman suara VoIP yang diterima di *Client* 2 menggunakan peralatan yang sama. Rekaman suara tersebut diperoleh dari paket RTP dengan cara *Wireshark* melakukan konversi dari paket RTP yang sudah dibaca menjadi *file* suara dengan

ekstensi *.au. Data yang diperdengarkan sesuai dengan jumlah konfigurasi yang digunakan, yaitu tiga data. Responden menilai kualitas suara yang didengar berdasarkan Tabel 2.12. Suara asli yang diinputkan pada *Client* 1 menjadi referensi dan diberi nilai MOS sebesar 5. Nilai MOS akan dibandingkan untuk masing-masing konfigurasi. Apabila nilai tersebut kurang dari nilai referensi, maka kualitas suara masih buruk, bisa ditunjukkan dengan suara yang terputus-putus.

Tabel 5.6 Data MOS *Polling*

No.	Peserta	Non-VPN	VPN-less	VPN-most
1.	Ali	4	3	3
2.	Nike	4	4	3
3.	Ciput	4	3	4
4.	Ani	4	3	2
5.	Norma	3	2	2
6.	Ayik	4	4	2
7.	Idris	4	3	3
8.	Marcelia	4	4	4
9.	Erriyon	4	3	3
10.	Galih	4	3	3
11.	Anne	4	2	2
12.	Niken	4	4	3
13.	Arif	4	4	2
14.	Dading	3	3	2
15.	Devi	4	3	3
16.	Yudha	4	4	3
17.	Pipit	3	2	2
18.	Heri	4	3	3
19.	Gatra	4	4	3
20.	Andik	3	3	2
21.	Bagus	4	3	3
22.	Wisnu	4	3	4
23.	Rendy	4	3	3
24.	Firman	4	3	2
25.	Fajar	4	4	3
26.	Koko	4	3	3
27.	Ferin	4	3	2
28.	Teguh	4	3	3
29.	Antonio	4	4	2
30.	Cone	4	3	2
31.	Eva	4	4	3
32.	Shinta	4	3	3
33.	Gusti	4	4	4
34.	Fredy	4	3	3
35.	Erwan	4	4	2
36.	Hari	4	4	3
37.	Faizal	4	3	2
38.	Ari	4	3	3
39.	Ihsan	4	3	3
40.	Fira	4	3	3
41.	Naufal	4	4	3
42.	Indirra	4	2	2
43.	Rara	4	4	3
44.	Yoyon	3	3	2
45.	Evey	4	4	3
46.	Indarti	4	3	2
47.	Weti	4	3	3
48.	Lukman	4	3	2
49.	Novan	4	3	3
50.	Hari	4	3	3
Rata - rata		**3,90**	**3,24**	**2,72**

Sumber : Hasil Pengujian

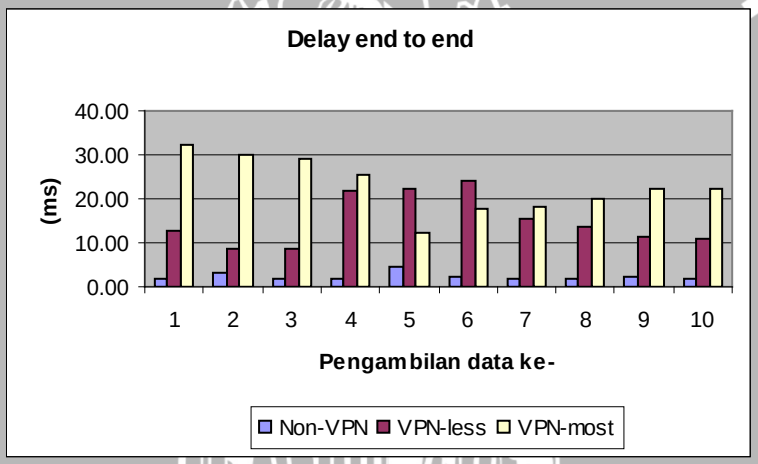
BAB VI
ANALISIS DATA

6.1 Analisis Performansi Pengujian

Setelah dilakukan proses pengambilan data dengan program *Wireshark*, selanjutnya rata-rata dari masing-masing konfigurasi ditampilkan dalam grafik. Dari grafik akan diketahui konfigurasi mana yang memiliki performansi lebih baik dan apakah sudah sesuai dengan rekomendasi ITU-T G.1010 tentang kategori QoS multimedia.

6.1.1 Analisis *Delay*

Delay yang diambil merupakan *delay end-to-end*, yaitu *delay* yang terjadi saat paket RTP ditransmisikan dari *Client* 1 menuju *Client* 2.



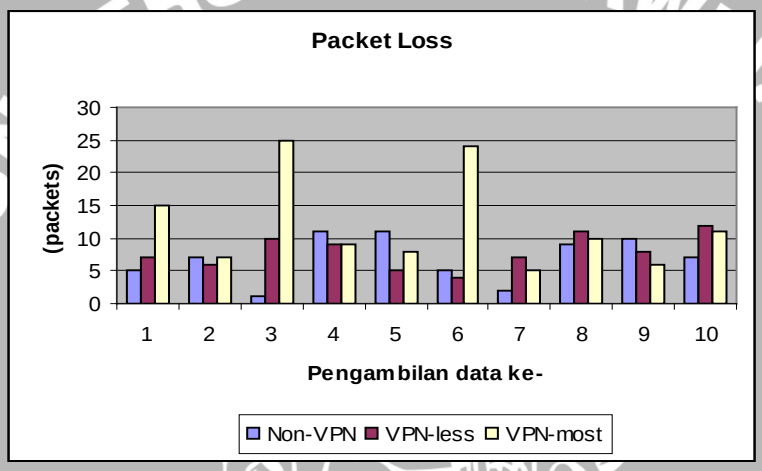
Gambar 6.1 Grafik *Delay* (ms)
Sumber: Analisis

Pada gambar 6.1, pada konfigurasi Non-VPN, diperoleh nilai *delay* rata-rata sebesar 2,35 ms. Pada konfigurasi *less secure* diperoleh nilai *delay* rata-rata sebesar 15,01 ms dan 22,92 ms untuk konfigurasi *most secure*. Nampak dari gambar tersebut nilai *delay* untuk konfigurasi Non-VPN cenderung lebih kecil daripada konfigurasi *less secure* dan *most secure*. *Delay* pada masing-masing konfigurasi relatif kecil dan masih sesuai dengan rekomendasi ITU-T G.1010, nilai delay maksimum adalah sebesar 150 ms. Nilai *delay* yang kecil disebabkan oleh kecilnya ukuran paket suara yang dikirim.

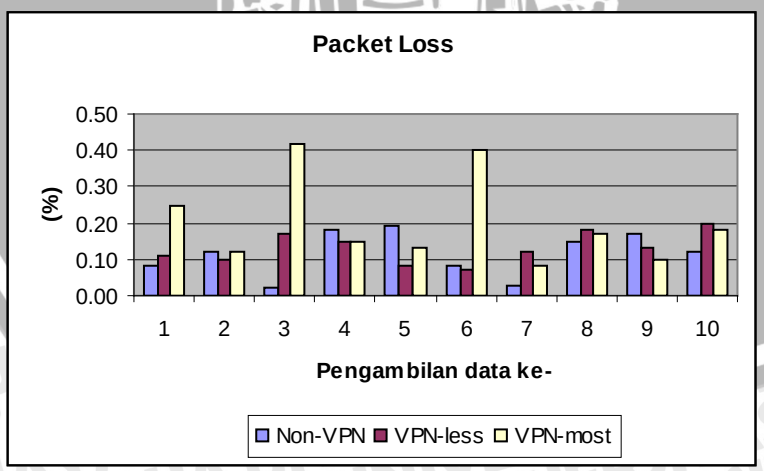
Selain ukuran paket, koneksi dengan VPN juga mempengaruhi lama proses enkapsulasi, sehingga paket dengan konfigurasi *most secure* akan diproses lebih lama daripada paket dengan konfigurasi *less secure*. Hal ini menjadikan paket-paket VoIP yang diproses menggunakan konfigurasi *most secure* mengalami *delay* yang lebih besar daripada paket yang diproses menggunakan konfigurasi *less secure*.

6.1.2 Analisis *Packet Loss*

Packet loss merupakan merupakan hilangnya paket data dalam transmisi data. Hal ini dapat terjadi karena berbagai hal, antara lain degradasi sinyal, tumbukan antar paket dalam sebuah *switch*, serta kesalahan *software*.



Gambar 6.2 Grafik *Packet Loss (packets)*
Sumber: Analisis



Gambar 6.3 Grafik *Packet Loss* (%)
Sumber: Analisis

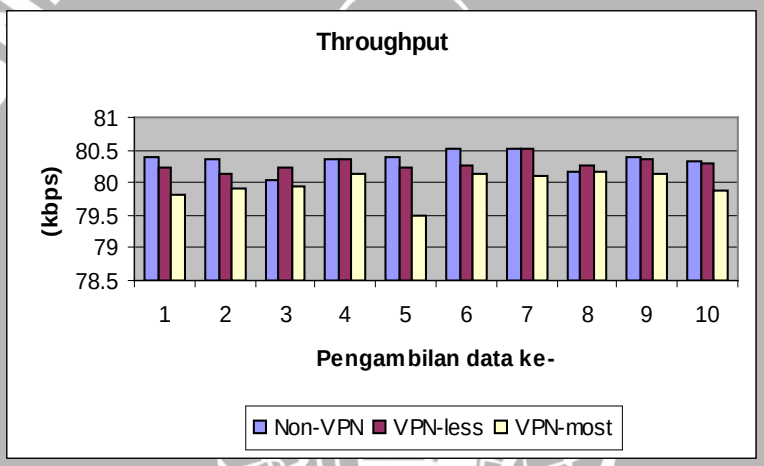
Pada gambar 6.3, nilai *packet loss* pada konfigurasi Non-VPN rata-rata sebesar 0,11%. Pada konfigurasi *less secure* nilai rata-rata sebesar 0,13% serta

pada konfigurasi *most secure* nilai rata-rata sebesar 0,20%. Menurut rekomendasi ITU-T G.1010 tentang kategori QoS multimedia, nilai *packet loss* maksimal sebesar 3%. Jadi secara keseluruhan, nilai *packet loss* pada ketiga konfigurasi masih sesuai dengan rekomendasi.

Persen paket hilang terbesar muncul pada konfigurasi *most secure*, yaitu pada pengambilan data ketiga, sebanyak 0,42%. Sedangkan persen paket hilang terkecil muncul pada konfigurasi Non-VPN, pada pengambilan data ketiga sebesar 0,02%. Jumlah *packet loss* yang relatif kecil bisa dikarenakan kapasitas link yang masih mencukupi.

6.1.3 Analisis *Throughput*

Throughput yang dimaksud adalah *throughput* paket-paket RTP, yang membawa data *voice*.



Gambar 6.4 Grafik *Throughput* (kbps)
Sumber: Analisis

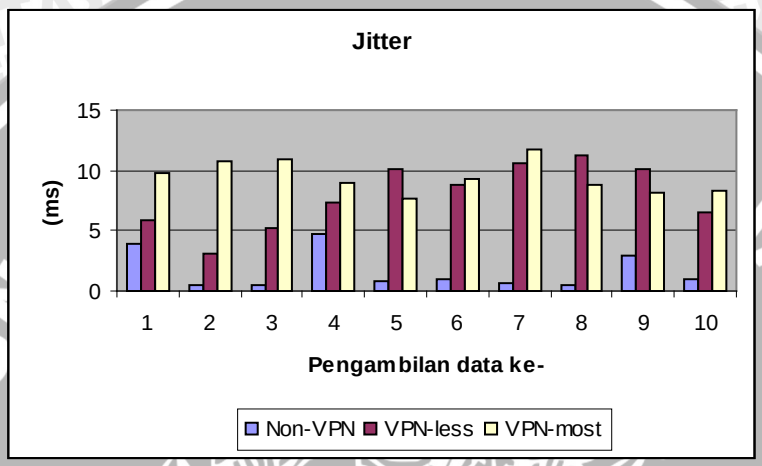
Pada gambar 6.4, pada konfigurasi Non-VPN diperoleh nilai *throughput* rata-rata sebesar 80,34 kbps. Pada konfigurasi *less secure* diperoleh nilai rata-rata sebesar 80,29 kbps dan 79,98 kbps untuk konfigurasi *most secure*. Nampak dari gambar tersebut, nilai *throughput* untuk konfigurasi Non-VPN cenderung lebih stabil dibandingkan dengan nilai *throughput* pada konfigurasi *less secure* dan *most secure*. Menurut rekomendasi ITU-T G.1010, nilai *throughput* minimal sebesar 80 kbps. Nilai *throughput* pada konfigurasi VPN *most secure* di bawah rekomendasi disebabkan terjadinya *packet loss.*

Nilai *throughput* yang semakin besar akan menunjukkan semakin bagus pula konfigurasi sistem tersebut memiliki kemampuan jaringan dalam

mentransmisikan data. Dari penggunaan VPN mengakibatkan penurunan besar *throughput* dibandingkan pada sistem Non-VPN.

6.1.4 Analisis *Jitter*

Jitter yang diperoleh merupakan *jitter* rata-rata, yang dihitung dari tiap paket RTP yang kemudian dihitung nilai rata-ratanya. Sesuai dengan RFC 3550 tentang RTP, setiap paket RTP memiliki nilai *jitter* tersendiri. Dari *Wireshark*, nilai tersebut dihitung rata-ratanya.



Gambar 6.5 Grafik *Jitter* (ms)
Sumber: Analisis

Pada gambar 6.5, pada konfigurasi Non-VPN, diperoleh nilai *jitter* rata-rata sebesar 1,66 ms. Pada konfigurasi *less secure* diperoleh nilai *jitter* rata-rata sebesar 7,88 ms dan 9,43 ms untuk konfigurasi *most secure*. Menurut rekomendasi ITU-T G.1010, nilai *jitter* maksimum adalah sebesar 5 ms, sehingga hanya konfigurasi Non-VPN yang masih sesuai rekomendasi. Sedangkan pada konfigurasi sistem VPN nilai *jitter* cenderung lebih tinggi. Penggunaan VPN menaikkan nilai *jitter* pada konfigurasi *less secure* dan *most secure*. Hal ini dipengaruhi adanya koneksi VPN, maka data mengalami beberapa proses *delay* untuk proses *security* yang lengkap dibandingkan dengan koneksi tanpa VPN.

6.1.5 Analisis MOS dari *Polling*

Dari pengujian kualitas suara secara *polling* diperoleh rata-rata untuk masing-masing konfigurasi, yaitu Non-VPN sebesar 3,90, konfigurasi *less secure* sebesar 3,24, serta konfigurasi *most secure* sebesar 2,72. Berdasarkan Tabel 2.12 MOS diketahui bahwa pada konfigurasi Non-VPN memiliki nilai MOS tertinggi yang diindikasikan mendekati sangat bagus. Sedangkan konfigurasi *less secure*

dan *most secure* memiliki nilai sedikit dibawah konfigurasi Non-VPN diindikasikan memiliki kualitas rata-rata.

Penurunan kualitas suara dari konfigurasi Non-VPN sampai konfigurasi *most secure* bisa diakibatkan oleh nilai *delay* dan *packet loss.* Nilai *delay* dan *packet loss* yang semakin besar akan mempengaruhi penurunan kualitas suara.

6.2 Analisis Performansi Teoritis

6.2.1 Analisis *Bandwidth Per Call*

Bandwidth yang diperlukan saat berkomunikasi suara tergantung pada *codec* yang digunakan. *Codec* yang digunakan adalah G.711 dan mempunyai karakteristik sebagai berikut:

- *Bitrate* 64 Kbps
- *Frame size* 20 ms *sample* periode
- 1 paket dikirim tiap 20 ms, atau 50 paket per detik

Voice payload (V) G.711 dapat diperoleh dengan mengalikan *bitrate codec* (B_c) dengan *frame size* (f_s).

$V = B_c \; x \; f_s$

$= 64 \text{ kbps } x \; 20.10^{-3} \text{ s}$

$= 1280 \text{ bit}$

$= 160 \text{ byte}$

Jadi besarnya *voice payload codec* G.711 adalah 160 byte.

Kebutuhan *bandwidth* VoIP dipengaruhi oleh ukuran *voice payload* dan *header* pada lapisan di bawahnya. Jika diketahui besarnya *overhead header* RTP/ UDP/ IP adalah 40 byte dan *ethernet* yang mempunyai ukuran *header* 14 byte dijadikan sebagai media *link layer*, maka besarnya paket/ kebutuhan *bandwidth*(BW) VoIP dengan menggunakan *codec* G.711 dapat diperoleh dengan persamaan 2-1.

$$BW(b/s) = (V + I + L)x8xP$$

$= (160 + 40 + 14) \; x \; 8 \; x \; 50$

$= 214 \; x \; 8 \; x \; 50$

$= 85600 \text{ bps}$

$= 85{,}6 \text{ kbps}$

Untuk memperkecil *bandwidth* yang dibutuhkan, *overhead header* dapat dikompresi sampai dengan 4 byte. Sehingga *bandwidth* yang dibutuhkan:

$$BW(b/s) = (V + I + L)x8xP$$
$$= (160 + 4 + 14)\ x\ 8\ x\ 50$$
$$= 178\ x\ 8\ x\ 50$$
$$= 71200 \text{ bps}$$
$$= 71{,}2 \text{ kbps}$$

Jadi kebutuhan *bandwidth* tanpa kompresi *header* sebesar 85,6 kbps dan 71,2 kbps dengan kompresi *header*. Sehingga dapat disimpulkan bahwa semakin besar paket data yang dikirimkan, maka makin besar pula *bandwidth* yang dibutuhkan.

6.2.2 Analisis *Delay End-to-end*

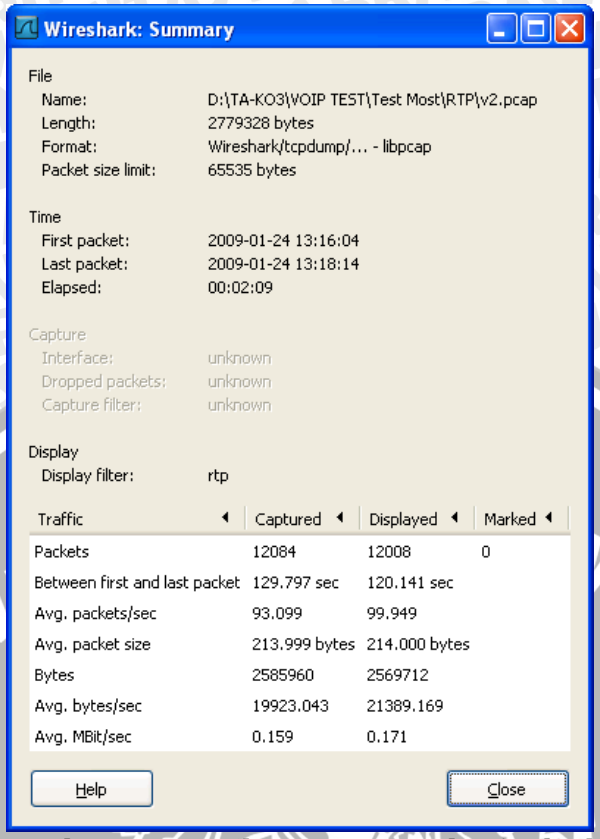
Delay end to end merupakan *delay* antara *node* sumber dan *node* tujuan. *Delay end to end* bersifat fluktuatif, karena paket untuk sampai ke tujuan melewati jalur yang berbeda-beda. *Delay end to end* dihitung dengan menggunakan persamaan 2-2.

$$t_{end-to-end} = t_{trans} + t_w + t_{proc} + t_p$$

dengan :

t_{trans} : *delay* transmisi/serialisasi (ms)
t_w : *delay* antrian (ms)
t_{proc} : *delay* proses (ms)
t_p : *delay* propagasi (ms)

Dari proses pengambilan data dengan program *wireshark* pada *Client* 2 didapatkan data seperti gambar 6.6.



Gambar 6.6 Tampilan *summary* pada *wireshark*
Sumber: Hasil pengujian

Tabel 6.1 Parameter-parameter hasil pengujian

No	Nama Data	Keterangan
1.	Jumlah paket diterima	12008 packets
2.	Rentang waktu	120,141 sec
3.	Kapasitas kanal	100 Mbps
4.	Panjang paket	214 byte
5.	*Bitrate* transmisi	0,171 Mbps
6.	*Header* RTP	12 byte
7.	*Header* UDP	8 byte
8.	*Header* IP	20 byte
9.	*Header ethernet*	14 byte

Sumber: Hasil pengujian

a. Delay proses

Delay proses adalah waktu yang dibutuhkan untuk memproses paket data dan untuk menentukan kemana data tersebut akan diteruskan. *Delay* proses berupa *delay* enkapsulasi dan *delay* dekapsulasi.

Pada saat paket berada pada *transport layer*, maka panjang segmen sesuai dengan persamaan 2-3 adalah sebagai berikut.

$$W_{segmen} = MSS + Header_{RTP} + Header_{UDP}$$

$$= 214 + 12 + 8 = 234 \text{ byte}$$

Setelah melalui *transport layer*, paket melalui *internet layer* dengan panjang datagram sesuai persamaan 2-4 adalah sebagai berikut.

$$W_{datagram} = W_{segmen} + Header_{IP}$$

$$= 234 + 20 = 254 \text{ byte}$$

Setelah itu paket melalui *network interface layer* baru kemudian dikirimkan dengan panjang *frame* sesuai Persamaan 2-5 adalah sebagai berikut.

$$W_{frame} = W_{datagram} + Header_{ethernet}$$

$$= 254 + 14 = 268 \text{ byte}$$

- *Delay* enkapsulasi

$$t_{enc} = \frac{W_{frame} - MSS}{C_{pros}} x8$$

$$= \frac{268 - 214}{171x1024} x8$$

$$= \frac{432}{175104}$$

$$= 0{,}002467$$

$$= 2{,}467ms$$

- *Delay* dekapsulasi

$$t_{dec} = \frac{W_{frame} - MSS}{C_{pros}} x8$$

$$= \frac{268 - 214}{171x1024} x8$$

$$= \frac{432}{175104}$$

$$= 0{,}002467$$

$$= 2{,}467ms$$

Sehingga delay proses dapat dituliskan sebagai berikut.

$$t_{proc} = t_{enc} + t_{dec}$$

$$= 2{,}467ms + 2{,}467ms$$

$$= 4{,}934ms$$

b. *Delay* transmisi/ serialisasi

Delay serialisasi adalah waktu yang diperlukan untuk menempatkan semua *frame* data pada media transmisi atau sebaliknya, sehingga efektif dan efisien dalam penggunaan kapasitas saluran/media transmisi.

Delay serialisasi dapat dituliskan sebagai berikut:

$$t_{trans} = \frac{(L + L')}{C} x8 = \frac{W_{frame}}{C} x8$$
$$= \frac{268}{100x1024x1024} x8$$
$$= \frac{2144}{104857600}$$
$$= 0{,}00002045\ s$$
$$= 0{,}02045\ ms$$

c. *Delay* antrian

Delay antrian adalah waktu dimana paket multimedia berada dalam antrian untuk diproses oleh *client*. *Delay* antrian dapat dihitung sesuai persamaan 2-16 adalah sebagai berikut.

$$t_w = \frac{\lambda}{\mu(\mu - \lambda)} + \frac{1}{\mu}$$

Dimana μ dapat dicari berdasarkan persamaan 2-12.

$$\mu = \frac{C_{trans}}{lx8}$$
$$= \frac{100x1024x1024}{214x8}$$
$$= \frac{104857600}{1712} = 61248{,}598/s$$

Dari pengambilan data didapat, paket yang ditangkap sebanyak 12008 paket dalam rentang waktu 120,141 s, maka nilai λ dapat dihitung dengan persamaan berikut.

$$\lambda = \frac{1}{t_v}$$
$$= \frac{12008}{120{,}141}$$
$$= 99{,}949/s$$

Dengan demikian *delay* antrian diperoleh dengan mensubstitusi nilai μ dan λ, yang ditunjukkan sebagai berikut.

$$t_w = \frac{\lambda}{\mu(\mu-\lambda)} + \frac{1}{\mu}$$
$$= \frac{99{,}949}{(61248{,}598)(61248{,}598-99{,}949)} + \frac{1}{61248{,}598}$$
$$= \frac{99{,}949}{3745269020{,}844} + \frac{1}{61248{,}598}$$
$$= 0{,}0000000267 + 0{,}0000163269$$
$$= 0{,}0000163536\ s$$
$$= 0{,}01635\ ms$$

d. *Delay* propagasi

Delay propagasi adalah waktu yang dibutuhkan untuk merambatkan paket multimedia melalui media transmisi dari *server* ke *client*. Dengan mengasumsikan panjang kabel UTP CAT 5 untuk *server* dan *client* adalah 12 meter, NIC yang digunakan pada *server* adalah Intel *NIC onboard* (Fast Ethernet), dan *switch* 3COM dengan kapasitas *bandwitdh* 100 Mbps. Maka *delay* propagasi dengan menggunakan persamaan 2-17 adalah:

$$t_p = t_{DTE} + t_{UTP} + t_{Router}$$
$$= 25\frac{1}{104857600} + 0{,}556x12\frac{1}{104857600} + 1{,}27x12\frac{1}{104857600}$$
$$= 0{,}0000002384 + 0{,}0000000636 + 0{,}000000145$$
$$= 0{,}0000003165\ s$$
$$= 0{,}0003165\ ms$$

Dari semua perhitungan *delay* proses, *delay* transmisi, *delay* antrian dan *delay* propagasi sebelumnya maka akan didapat *delay* total yaitu *delay end to end* secara teoritis.

$$t_{end-to-end} = \mathrm{t}_{\mathrm{proc}} + t_{trans} + t_w + t_p$$
$$= 4{,}934 + 0{,}02045 + 0{,}01635 + 0{,}0003165$$
$$= 4{,}9711165\ \mathrm{ms}$$
$$= 4{,}97\ \mathrm{ms}$$

6.2.3 Analisis Kualitas Suara dari E-Model

Berdasarkan rekomendasi ITU-T G.107, E-Model digunakan untuk mengukur kualitas suara yang dinyatakan oleh faktor R dan nilai MOS. Kualitas suara pada VoIP dipengaruhi oleh *delay end-to-end* serta probabilitas *packet loss* pada jaringan tersebut.

a. Perhitungan probabilitas *packet loss*

Jika diketahui nilai BER standar untuk saluran transmisi kabel tembaga (*ethernet*) 10^{-5} dan serat optik (IP *backbone*) 10^{-9}, serta panjang paket termasuk header ($l+l'$) adalah 2144 bit, maka besarnya probabilitas *packet loss* pada jaringan *ethernet* (ρ_{eth}) dan jaringan IP *backbone* (ρ_{IPB}) adalah:

$$\rho_{eth} = 1 - (1 - \rho_e)^{l+l'}$$
$$= 1 - (1 - 10^{-5})^{2144}$$
$$= 1 - 978{,}788.10^{-3}$$
$$= 21{,}212.10^{-3}$$

$$\rho_{IPB} = 1 - (1 - \rho_e)^{l+l'}$$
$$= 1 - (1 - 10^{-9})^{2144}$$
$$= 1 - 999{,}998.10^{-3}$$
$$= 2{,}144.10^{-6}$$

Maka probabilitas *packet loss* total adalah:

$$\rho_{network} = 1 - [(1 - \rho_{eth})(1 - \rho_{IPB})^h]$$
$$= 1 - [(1 - 21{,}212.10^{-3})(1 - 2{,}144.10^{-6})^1]$$
$$= 1 - 0{,}979$$
$$= 0{,}021$$

b. Perhitungan faktor R

Faktor R sebagai salah satu cara untuk mengetahui kualitas suara dari suatu sambungan telepon dapat diperoleh dengan menggunakan persamaan 2-22.

$$R = 94{,}2 - I_d(Ta) - I_{ef}(codec,loss)$$

Besarnya kerusakan kualitas suara oleh *delay* (*Id*) dapat ditentukan dengan persamaan 2-23.

$$I_d = 0{,}024d + 0{,}11(d\text{-}177{,}3)H(d\text{-}177{,}3)$$
$$= (0{,}024 \times 4{,}97) + 0{,}11\ (4{,}97\text{-}177{,}3)H(4{,}97\text{-}177{,}3)$$
$$= 0{,}11928 + 0$$
$$= 0{,}119$$

Sedangkan besarnya kerusakan kualitas suara oleh probabilitas *packet loss* adalah:

$$I_{ef} = \gamma_1 + \gamma_2 \ln(1+\gamma_3 e)$$
$$= 0 + 30 \ln(1 + 15 e)$$
$$= 0 + 30 \ln[1 + (15 \times 0{,}021)]$$

= 30 ln(1 + 0,315)

= 30 ln1,315

= 30 *x* 0,274

= 8,22

Sehingga besarnya faktor R adalah:

$R = 94{,}2 - I_d - I_{ef}$

= 94,2 – 0,119 – 8,22

= 85,861

Berdasarkan Tabel 2.13, R-*factor* dengan nilai 85,861 mempunyai predikat memuaskan.

c. Perhitungan MOS

Rating kualitas suara yang dinyatakan oleh nilai MOS dapat diperoleh dengan menggunakan persamaan:

$MOS = 1 + 0{,}035R + 7.10^{-6}R(R\text{-}60)(100\text{-}R)$

$= 1 + (0{,}035x88{,}141) + (7.10^{-6}x88{,}141)(88{,}141 - 60)(100 - 88{,}141)$

= 1 + 3,085 + 0,206

= 4,29

Berdasarkan Tabel 2.12, MOS dengan nilai 4,291 mempunyai predikat memuaskan.

6.3 Perbandingan Pengujian dan Teori

Perbandingan antara perhitungan *delay end to end* dan pengujian pada masing-masing konfigurasi dengan teori adalah sesuai dengan tabel berikut.

Tabel 6.2 Perbandingan *delay end to end* hasil pengujian dan hasil perhitungan

Delay end to end			
Hasil pengujian			Hasil perhitungan
Non-VPN	VPN-less	VPN-most	
2,35 ms	15,01 ms	22,92 ms	4,97 ms

Sumber: hasil perhitungan

Dari hasil pengujian dan perhitungan diketahui bahwa sistem ini layak untuk diterapkan, parameter yang mendukung adalah *delay end to end* untuk mengirimkan satu paket data multimedia rata-rata dibawah 150 ms.

Besarnya *delay end to end* berbeda-beda atau mengalami fluktuasi, hal ini dikarenakan besarnya *bandwidth server* yang tidak konstan karena jalur paket data tidak hanya digunakan untuk pengujian skripsi ini namun juga untuk publik.

Perbandingan nilai MOS hasil melalui *polling* pada masing-masing konfigurasi dan melalui perhitungan E-Model adalah sesuai dengan tabel berikut.

Tabel 6. 3 Perbandingan MOS melalui *Polling* dan E-Model

Mean Opinion Score			
Hasil *Polling*			Hasil E-Model
Non-VPN	VPN-less	VPN-most	
3,90	3,24	2,72	4,29

Sumber: hasil perhitungan

Dari tabel 6.3 nampak bahwa hasil melalui *polling* dan melalui perhitungan E-Model adalah berbeda. Hal ini dipengaruhi oleh subyektivitas peserta *polling* dan peralatan *speaker* yang digunakan. Sedangkan nilai MOS melalui perhitungan E-Model cenderung mendekati nilai *user satisfaction* untuk *codec* G.711 pada tabel 2.13.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan, maka kesimpulan yang dapat diambil adalah sebagai berikut:

1. Penggunaan *codec* G.711 yang mempunyai karakteristik *bitrate* 64 kbps dan *framesize* 20 ms berpengaruh terhadap kebutuhuan *bandwidh* adalah sebesar 85,6 kbps.
2. Pada pengujian *delay end to end* dan *jitter,* konfigurasi VPN *most secure* memiliki nilai yang paling tinggi, yaitu 22,92 ms dan 9,43 ms. Hal ini menunjukkan bahwa lambatnya waktu *delay* dan *jitter* pada koneksi VPN disebabkan adanya waktu tunda akibat proses enkapsulasi, dekapsulasi, dan autentifikasi dengan maksud *security* jalur VPN.
3. *Delay end to end* hasil perhitungan sebesar 4,29 ms. Besarnya *delay end to end* setiap paket berbeda-beda dikarenakan tidak konstannya *bandwidth server* diakibatkan jalur digunakan tidak hanya untuk pengujian skripsi ini namun juga digunakan untuk jalur publik.
4. Untuk analisis *packet loss* diperoleh nilai terendah pada konfigurasi Non-VPN, yaitu sebesar 0,11% dan nilai tertinggi pada konfigurasi VPN *most secure,* yaitu sebesar 0,20%. Pada ketiga konfigurasi persen *packet loss* masih memenuhi rekomendasi ITU-T G.1010 maksimal sebesar 3%.
5. Penggunaan VPN menurunkan besar *throughput.* Konfigurasi VPN *most secure* memiliki nilai *throughput* terendah sebesar 79,98 kbps dan di bawah rekomendasi G.1010 sebesar 80 kbps.
6. Nilai MOS dari *polling* terhadap 50 orang menghasilkan nilai rata-rata 3,90 pada konfigurasi Non-VPN, pada konfigurasi VPN *less secure* sebesar 3,24, dan pada konfigurasi VPN *most secure* sebesar 2,72. Nilai tersebut tergolong di bawah rata-rata dan responden banyak yang tidak puas untuk komunikasi VoIP.
7. Nilai MOS dari perhitungan E-Model menghasilkan nilai 4,29. Nilai tersebut lebih tinggi dibandingkan nilai MOS yang diperoleh melalui

polling, hal tersebut dipengaruhi oleh subyektifitas dari peserta *polling* dan kualitas speaker yang digunakan.

8. Berdasarkan besar *delay, jitter, packet loss, throughput*, dan kualitas suara maka dapat diketahui bahwa performansi sistem VoIP dengan SIP melalui VPN tergolong baik, meskipun ada perbedaan yang tidak terlalu signifikan dengan sistem Non-VPN.

7.2 Saran

Beberapa hal yang disarankan untuk pengembangan penelitian lebih lanjut antara lain :

1. Pengambilan data dilakukan pada skala jaringan yang lebih luas, misalnya jaringan intranet kampus.
2. SIP bisa juga diterapkan untuk aplikasi multimedia yang lain, seperti *video streaming.*
3. Ditambahkan konfigurasi VPN yang berbeda, yaitu pada metode enkripsi atau autentikasi.
4. Digunakan *codec* suara selain G.711

DAFTAR PUSTAKA

Iskandarsyah, MH. 2005. *Dasar-dasar Jaringan VoIP*. *http://www.ilmukomputer.com*. (diakses tanggal 28 April 2008)

Hendra Cahya. 2007. *H.323 versus SIP*. *http://www.ilmukomputer.com*. (diakses tanggal 29 April 2008)

Martinus Indra S. 2004. *Mekanisme dan Implementasi Keamanan pada SIP*. *http://www.ilmukomputer.com*. (diakses tanggal 29 April 2008)

Raharja, Anton. 2006. *Session Initiation Protocol*. *http://www.ilmukomputer.com*. (diakses tanggal 29 April 2008)

Purbo, Onno W., et al., 2001. *Buku Pintar Internet : TCP/IP*. Jakarta: PT. Elex Media Komputindo Kelompok Gramedia.

Schwartz, Mischa, 1987. *Telecommunication Networks: Protocols, Modelling and Analysis*. New York: Addison-Wesley Publishing Company.

Forouzan, Behrouz A. 2001. *Data Communications and Networking*. Singapore: McGraw-Hill Book, Co.

ITU-T. 1996. *General Characteristic of International Telephone Connections and International Telephone Circuits*. Rekomendasi ITU-T G.114.

ITU-T. 2005. *The E-model, a computational model for use in transmission planning*. Rekomendasi ITU-T G.107.

Markus Feilner. 2006. *Open VPN, Building and Integrating Virtual Private Network*. United Kingdom: Packt Publishing.

Cisco System. 2000. *VoIP-Per Call Bandwidth Consumption*. *http://www.cisco.com*. (diakses tanggal 17 Juli 2008)

Zhou, et.al. 2006. *Estimation of Voice Over IP Quality in Netherland*. The Netherlands: Delft University of Tech.

Sinreich, Henry and Johnson, Alan B. 2006. *Internet Communication Using SIP, 2nd edition*. Canada: Wiley Publishing Inc.

Anonim. 2005. *Block cipher modes of operation*. Encyclopedia-Wikipedia. *http://www.wikipedia.org/wiki/Block_cipher_modes_of_operation*.

Anonim. 2008. *Mean opinion score*. Encyclopedia-Wikipedia. *http://www.wikipedia.org/wiki/Mean_opinion_score*.

Anonim. 2008. *Lempel Ziv Oberhumer*. Encyclopedia-Wikipedia. *http://www.wikipedia.org/wiki/ Lempel_ziv_oberhumer*.

Rukmana, Nanang. 2007. *Penerapan SIP pada Peer to Peer Internet Telephony*. Skripsi Tidak Diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, 2007.

Winanti, Nur. 2009. *Analisis Paket Trafik dan Video Streaming Mobile-VPN pada Teknologi GPRS/ GSMA dan CDMA*. Skripsi Tidak Diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, 2009.

LAMPIRAN

Lampiran 1. Nilai Ie (codec)

Dari rekomendasi ITU-T G.113, Appendix I

Codec Type	Reference	Operating Rate kbit/s	Ie value
Waveform Codecs			
PCM	G.711	64	0
ADPCM	G.726, G.727	40	2
	G.721, G.726, G.727	32	7
	G.726, G.727	24	25
	G.726, G.727	16	50
Speech Compression Codecs			
LD-CELP	G.728	16	7
		12.8	20
CS-ACELP	G.729	8	10
	G.729-A + VAD	8	11
ACELP	G.723.1	5.3	19
MP-MLQ	G.723.1	6.3	15

Lampiran 2. Diagram Alir

Diagram Alir Konfigurasi *Server*

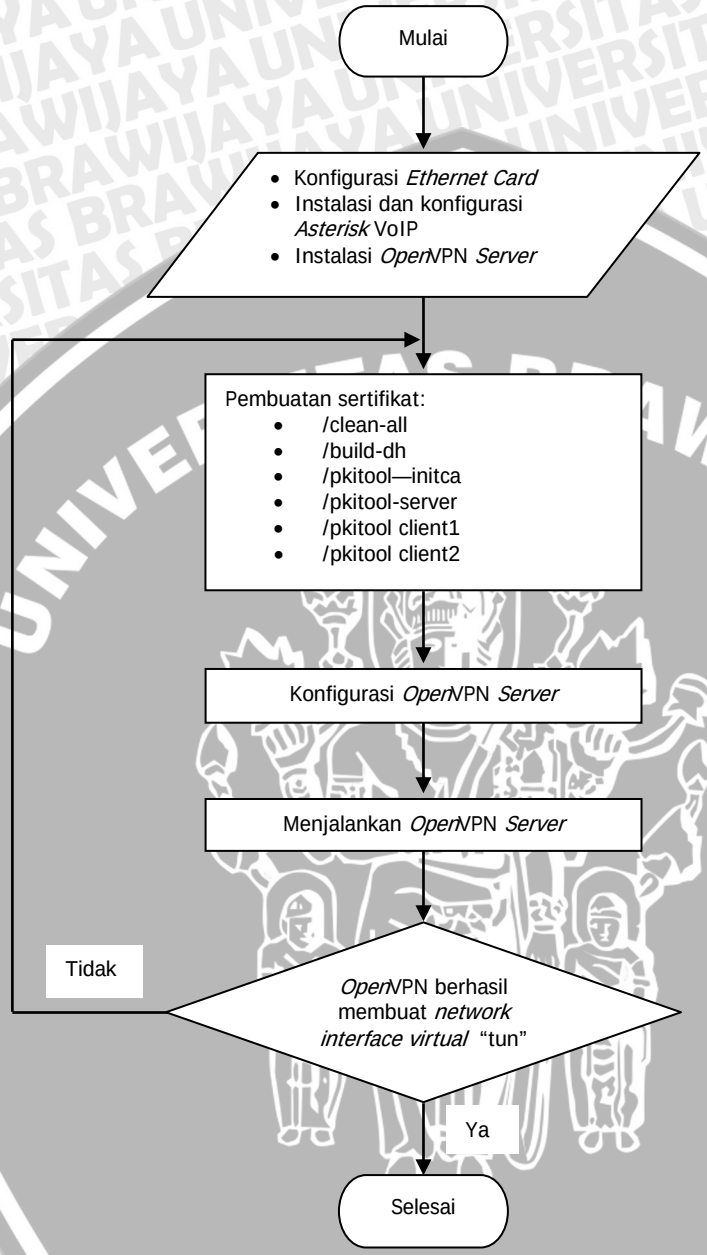


Diagram Alir Konfigurasi *Client*

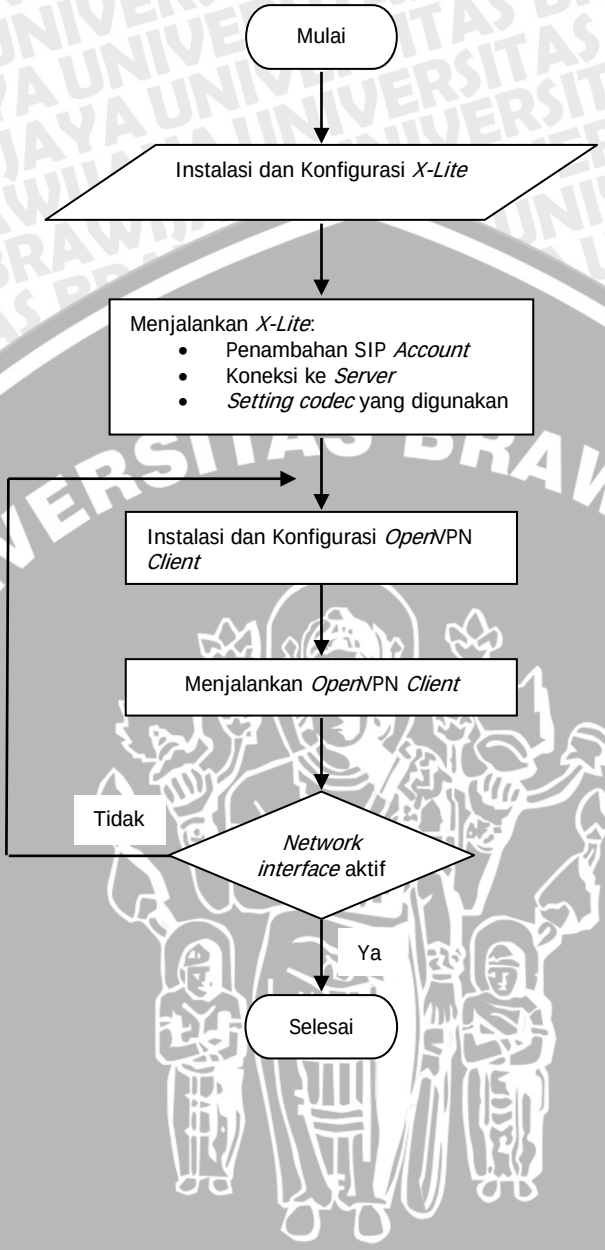


Diagram Alir Pengambilan Data dengan *Wireshark*

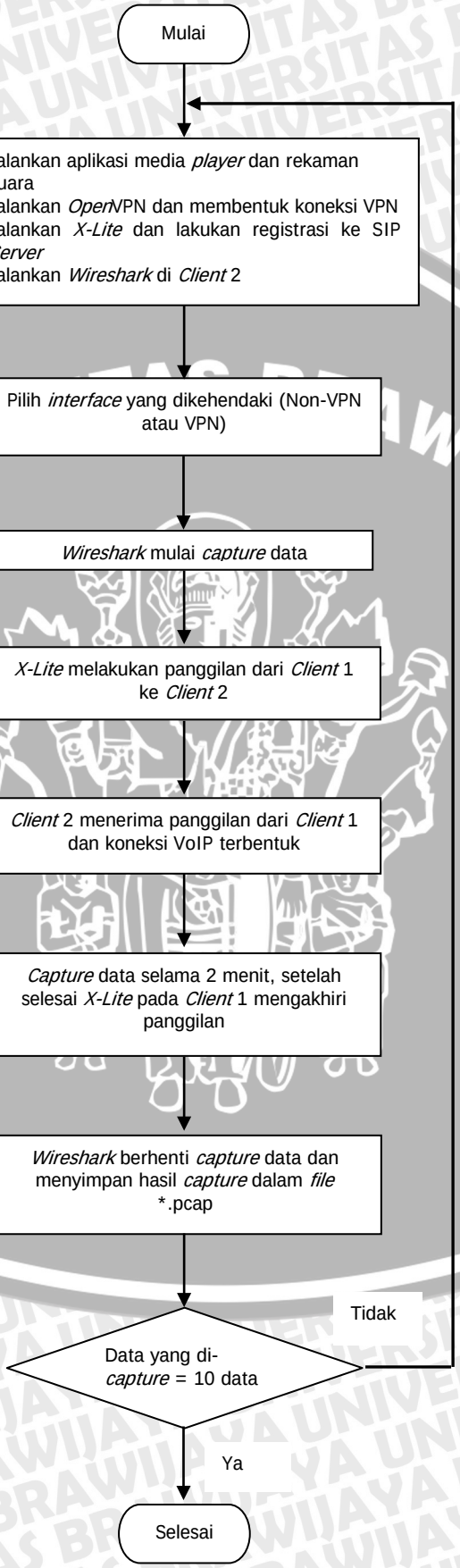


Diagram Alir Penerapan *Session Initiation Protocol* pada *Virtual Private Network*

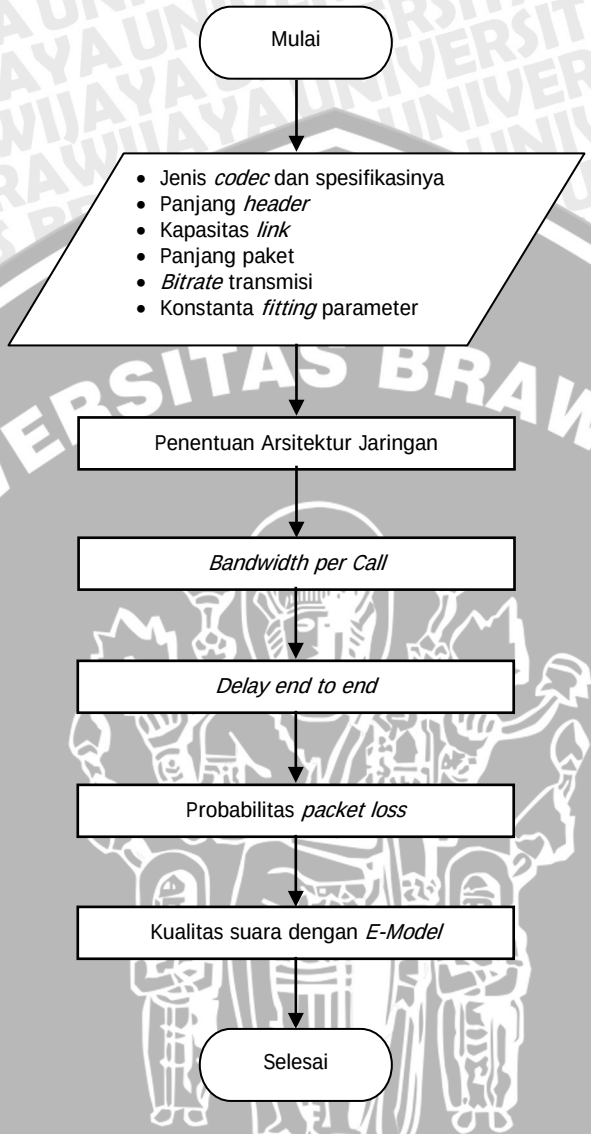


Diagram Alir Perhitungan *Bandwidth per Call*

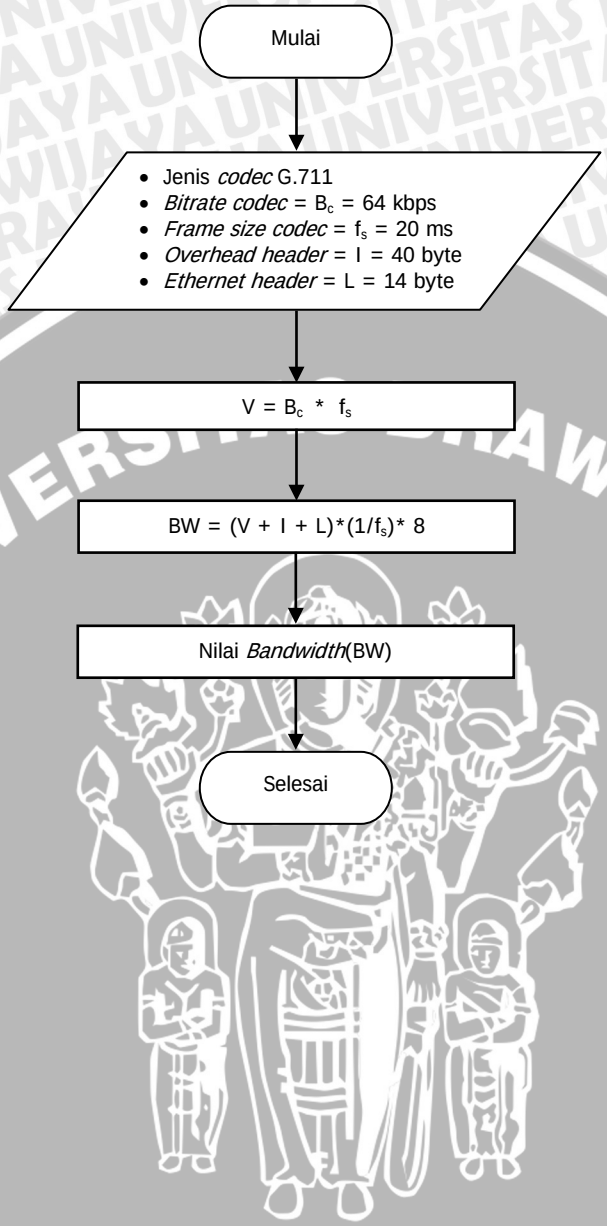


Diagram Alir Perhitungan *Delay End to end*

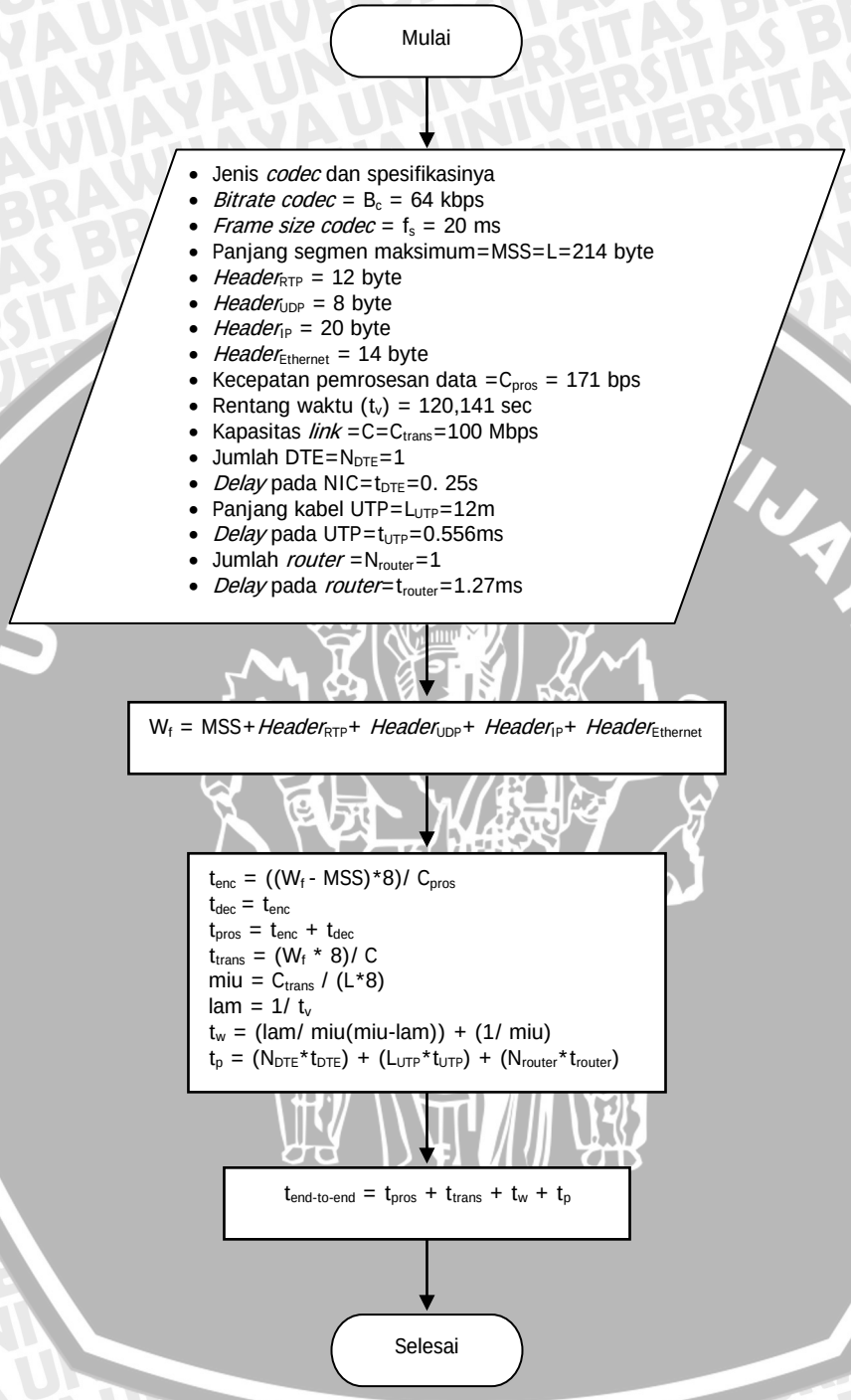
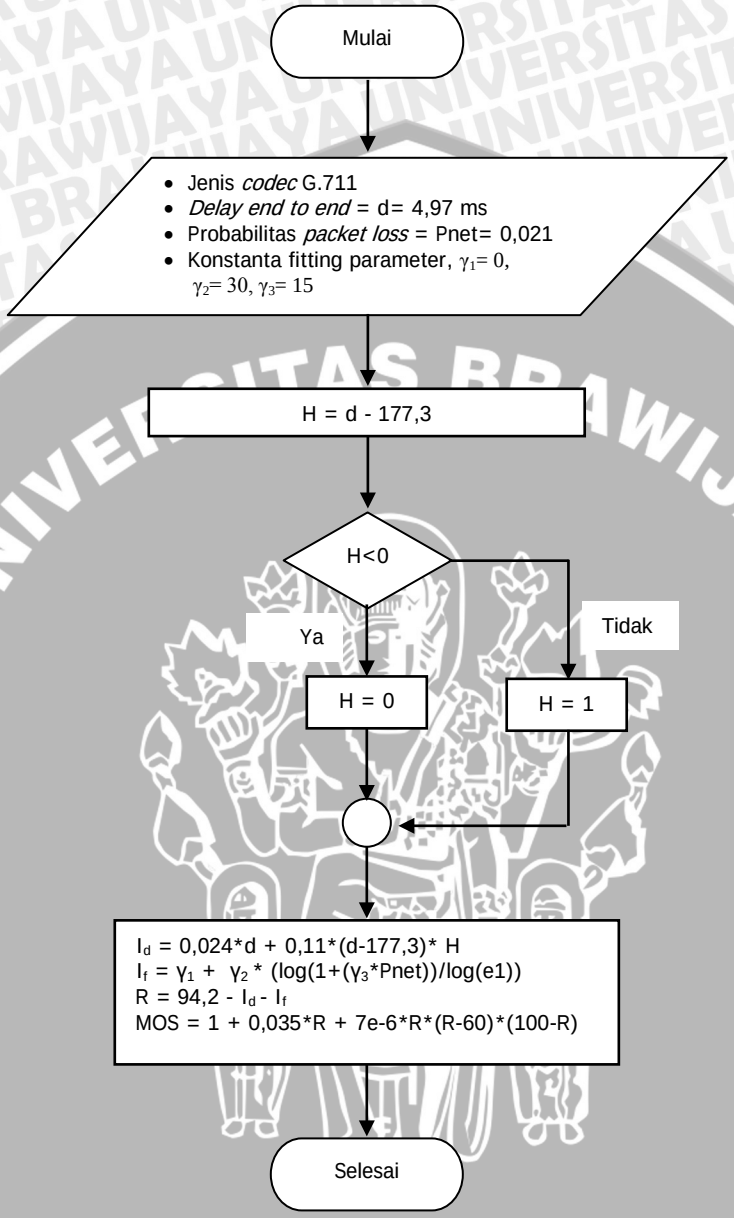


Diagram Alir Perhitungan Kualitas Suara VoIP
dengan menggunakan *E-Model*



RIWAYAT HIDUP

Yuyun Trihandini dilahirkan di Kediri pada tanggal 14 Juni 1981, merupakan putri ketiga dari tiga bersaudara dari pasangan Djoko Moentoro dan Suwarti.

Penulis menyelesaikan pendidikan taman kanak-kanak di TK Brawijaya tahun 1988, lulus SDK Frateran II Kediri tahun 1994, lulus SLTP Negeri 4 Kediri tahun 1997, dan menyelesaikan studi di SMU Negeri 2 pada kota yang sama tahun 2000. Setelah lulus SMU, penulis melanjutkan studi di D3 Computer Control Teknik Elektro ITS Surabaya dan lulus tahun 2003. Pada tahun 2003-2005 penulis bekerja di PT. Eterna Express Surabaya.