BAB II
LANDASAN TEORI

2.1. *Internet*

Internet merupakan singkatan dari kata *interconnection-networking*, bila dijabarkan secara 1itera global maka *internet* merupakan *network* 1iteratu diseluruh penjuru dunia yang saling terhubung satu sama lain dengan menggunakan standar *protocol* yang dikenal dengan *Transmision Control Protocol/Internet Protocol* (*TCP/IP*). Hal tersebut memungkinkan antar 1iteratu dapat saling mengakses informasi dan bertukar informasi. *Internet* mencangkup segala sesuatu secara luas baik itu dalam bidang komputerisasi maupun telekomunikasi.

TCP/IP adalah gabungan dari *protocol TCP* dan *IP* sebagai sekelompok *protocol* yang mengatur komunikasi data dalam proses pertukaran data dari satu 1iteratu ke 1iteratu lain di dalam jaringan internet yang akan memastikan pengiriman data sampai ke alamat yang dituju. *Protocol* ini menggunakan skema pengalamatan yang sederhana yang disebut sebagai alamat IP (*Ip address*) yang mengizinkan hingga ratusan juta 1iteratu untuk dapat saling berhubungan satu sama lainnya di Internet. *Protocol* ini juga bersifat *routable* yang berarti cocok untuk menghubungkan 1itera-sistem berbeda seperti Microsoft Windows dan keluarga UNIX untuk membentuk jaringan yang heterogen.

2.2. Jaringan Komputer

Jaringan 1iteratu merupakan salah satu komponen yang membentuk *internet*. Jarigan 1iteratu adalah Jaringan dari komunikasi data yang melibatkan lebih dari sebuah 1iteratu yang dihubungkan dengan jalur transmisi dan alat komunikasi yang membentuk system (M, 1992). Jaringan 1iteratu terbagi 3 yakni *LAN* (*Local Area Network*), *MAN* (*Metropolitan Area Network*) dan *WAN* (*Wide Area Network*).

2.2.1. *LAN* (*Local Area Network*)

LAN adalah sejumlah 2iteratu yang dihubungkan bersama di dalam satu areal tertentu yang tidak begitu luas, seperti di dalam satu kantor atau gedung (Debyo Hendry Santoso, 2012). Saat ini, kebanyakan *LAN* berbasis pada teknologi *IEEE* 802.3 *Ethernet* menggunakan perangkat *switch*, yang mempunyai kecepatan *transfer* data 10, 100, atau 1000 *Mbit*/*s*. Selain teknologi *Ethernet*, saat ini teknologi 802.11b atau biasa disebut *Wi-fi* juga sering digunakan untuk membentuk *LAN*.

Pada sebuah *LAN*, setiap *node* atau 2iteratu mempunyai daya komputasi sendiri. Setiap 2iteratu juga dapat mengakses sumber daya yang ada di *LAN* sesuai dengan hak akses yang telah diatur. Sumber daya tersebut dapat berupa informasi atau perangkat seperti *printer*. Pada *LAN*, seorang pengguna juga dapat berkomunikasi dengan pengguna yang lain dengan menggunakan aplikasi yang sesuai.

2.2.2. *MAN* (*Metropolitan Area Network*)

MAN dalah jaringan yang menghubungkan banyak *LAN* dalam suatu kota menggunakan kecepatan transfer tinggi. Biasanya *MAN* memiliki jangkauan sekitar 10 sampai 50 km. *MAN* merupakan perkembangan dari *LAN* yang biasanya sudah terdapat fasilitas *server*. *MAN* mungkin sekali dijalankan oleh sebuah organisasi, namun digunakan oleh banyak individu dan kebanyakan telah menyediakan layanan internet bagi jaringan 2iter atau *LAN*.

2.2.3. *WAN* (*Wide Area Network*)

Wide Area Network merupakan *network* 2iteratu yang mencakup area yang besar sebagai contoh yaitu *network* 2iteratu antar wilayah, kota bahkan Negara. *WAN* dapat didefinisikan juga sebagai *network* 2iteratu yang membutuhkan *router* dan saluran komunikasi 2itera. *WAN* digunakan untuk menghubungkan *network* 2iter yang satu dengan *network* 2iter yang lain, sehingga dapat saling bertukar informasi.

2.3. *Router*

Router merupakan perangkat jaringan yang memiliki tugas untuk mengatur lalu lintas data dalam jaringan melalui sebuah proses *routing*. Proses *routing* diatur oleh *routing protocol* yang telah diimplementasikan pada sebuah *network* (Gede Saindra S, 2012). Proses *routing* terjadi pada lapisan 3 yakni *network layer* dari *stack protocol* tujuh lapis *OSI*. *Router* sangat banyak digunakan dalam *network* berbasis teknologi *protocol TCP/IP*, dan *router* jenis itu disebut juga dengan *IP Router*. *Internet* merupakan contoh utama dari sebuah *network* yang memiliki banyak *router IP*.

2.3.1. *Routing*

Dalam *interconnection networking* terdapat beberapa hal penting salah satunya adalah proses *routing*. *Routing* itu sendiri adalah suatu mekanisme memindahkan informasi dari sumber ke tujuan melalui jaringan. Proses *routing* terjadi pada *network layer* standar *OSI* yang diimplementasikan pada *router* (Ferrianto Gozali, 2003).

Routing merupakan fungsi yang bertanggung jawab membawa data melewati sekumpulan jaringan dengan cara memilih jalur terbaik untuk dilewati oleh data. Jalur yang baik tergantung pada beban *network*, panjang datagram, *type of service requested* dan pola trafik. Pada umumnya skema *routing* hanya mempertimbangkan jalur terpendek *(the shortest path)*. *Router* menggunakan *routing table* untuk melewatkan/*forward traffic* yang diterima dari *router* lain atau *hosts*. Nassar dan Daniel (2000, dalam Gozali & Juniman, 2003) menyatakan bahwa terdapat 3 jenis *routing* yaitu *default routing*, *static routing*, dan *dynamic routing*.

2.3.1.1. *Default Routing*

Pada umumnya *default route* dibangun secara manual oleh *network administrator*. *Default routing* itu sendiri hanya digunakan ketika pada *table routing* tidak terdapat informasi atas jalur yang dapat digunakan untuk mengirim informasi ke tujuan (secara eksplisit).

2.3.1.2. *Routing Static*

Routing static merupakan pengaturan *routing* paling sederhana yang dapat dilakukan pada jaringan 4iteratu. *Routing static* adalah rute-rute ke *host* atau jaringan tujuan yang dimasukkan secara manual oleh *network administrator* dalam *routing table* suatu *router* (Edi, 2006).

Routing static tidak akan merubah informasi yang ada pada *table routing* secara otomatis, sehingga *administrator* harus melakukan perubahan secara manual apabila terjadi perubahan terhadap *topology network*. Penggunaan *routing static* dalam sebuah jaringan kecil tentu bukanlah suatu masalah, namun tentu dapat dibayangkan bagaimana jika harus melengkapi *routing table* setiap *router* yang jumlahnya tidak sedikit dalam jaringan besar maupun internet. Oleh karena itu, penggunaan *routing static* dalam *network* bersekala besar cenderung membutuhkan kerja dan waktu esktra untuk melengkapi *table routing* seluruh *router* yang ada.

2.3.1.3. *Routing Dynamic*

Bila pada *routing static, table routing* dibangun secara manual oleh *network administrator*, berbeda dangan *routing dynamic*. *Routing dynamic* adalah suatu mekanisme untuk mengisi informasi pada *table routing* secara otomatis. Dalam hal ini *routing protocol* sangat diperlukan untuk melakukan mekanisme tersebut. *Routing protocol* mengatur *router-router* dapat berkomunikasi satu dengan yang lainnya dan saling memberikan informasi *routing* yang dapat mengubah isi *forwarding table*, tergantung keadaan jaringannya.

Routing dynamic yang popular mengacu pada dua tipe algoritma oleh Bellman-Ford dengan algoritma *distance vector*-nya dan oleh Djikstra dengan algoritma *link state*-nya. Pada *network* besar yang menggunakan banyak *router, routing dynamic* merupakan metode yang paling umum. Dengan cara ini, apabila terjadi perubahan terhadap *topology network* maka *network administrator* tidak perlu melakukan perubahan *table routing* karena perubahan tersebut akan terjadi secara otomatis berkat peran *routing protocol*.

2.4. *Routing Protocol*

Routing protocol merupakan aturan yang melakukan pertukaran informasi *routing* dari *router* satu ke *router* lainnya dimana informasi yang diperoleh akan digunakan untuk membentuk dan memperbaiki *table routing* (Lady Silk M, 2011). *Routing protocol* bertujuan mencari rute terbaik untuk mencapai tujuan. Setiap *protocol routing* memiliki cara dan metodenya sendiri-sendiri. Secara garis besar, *routing protocol* dibagi menjadi dua yakni *EGP* (*Exterior Routing Protocol*) dan *IGP* (*Interior Routing Protocol*).

2.4.1. *Exterior Gateway Protocol* (*EGP*)

Exterior Gateway Protocol merupakan jenis *protocol routing* yang digunakan untuk menghubungkan *Autonomous System* (*AS*) yang berbeda di dalam *network* berskala besar. *Routing protocol* yang termasuk kedalam *EGP* adalah *Border Gateway Protocol* (*BGP*). Saat ini *BGP* yang digunakan adalah *BGP* versi 4 sedangkan *BGP* versi terdahulu sudah tidak digunakan lagi. *BGP* merupakan *backbone* dari *interconnection network* dunia.

Routing protocol BGP baru dapat dikatakan bekerja pada sebuah *router* jika sudah terbentuk sesi komunikasi dengan *router* tetangganya yang juga menjalankan *BGP*. Sesi komunikasi ini adalah berupa komunikasi dengan *protocol TCP*. Setelah terjalin komunikasi ini, maka kedua buah *router BGP* dapat saling bertukar informasi rute.

2.4.2. *Interior Gateway Protocol* (*IGP*)

IGP merupakan *routing protocol* yang bekerja di dalam satu *autonomous system* (*AS*) yang sama. *Protocol* ini menerapkan bahwa *router-router* saling berhubungan dengan *system* mereka dan secara bebas saling menukarkan informasi *routing* dengan beberapa *router* yang berada pada satu *autonomous system* (*AS*). *IGP* digunakan untuk menentukan rute terbaik di dalam suatu *autonomous system*. Banyak *protocol routing* yang termasuk kedalam *IGP* diantaranya adalah *Ripv2*, *EIGRP* dan lain-lain.

2.4.2.1. *Routing Information Protocol Version* 2 (*RIPv2*)

Routing Information Protocol (*RIP*) adalah *protocol* yang memanfaatkan algoritma Bellman-Ford (kelompok *protocol distance-vector*) dalam pemilihan rute terbaiknya. *RIP* memiliki tingkat kompleksitas komputasional yang lebih rendah, sehingga konsumsi sumber daya *memory* juga lebih rendah. Akan tetapi, konsekuensi yang ditimbulkan dari hal tersebut adalah bahwa penggunaan *RIP* hanya terbatas pada jaringan menengah ke bawah dengan jumlah *host* yang tidak terlalu besar.

Perlu diketahui bahwa *RIP* tidak mengadopsi *protocol distance-vector* begitu saja, melainkan dengan melakukan beberapa penambahan pada algoritmanya agar perutean dapat diminimalkan. *Split horizon* digunakan *RIP* untuk meminimalkan efek *bouncing*.

Untuk mencegah kasus menghitung sampai tak hingga, *RIP* menggunakan metode *triggered update*. *RIP* memiliki penghitung waktu (*timer*) untuk mengetahui kapan perute harus kembali memberikan informasi perutean. Jika terjadi perubahan pada jaringan, sementara *timer* belum habis, perute tetap harus mengirimkan informasi perutean karena dipicu oleh *triggered update*. Dengan demikian, perute dalam jaringan dapat dengan cepat mengetahui perubahan yang terjadi dan meminimalkan kemungkinan *routing loop* terjadi.

Untuk jaringan 6iteratu yang sangat kecil, terbatas untuk jaringan dengan pencarian jalur ke tujuan maksimum lompatan sebanyak 15 kali lompatan. (Edward & Bramante, 2009) . Dalam proses *update* pada *RIP* dikenal istilah *RIP Timer*s. Terdapat 4 macam *RIP timer*s yakni :

a. *Route Update Timer* , Interval antar *update* biasanya 30 detik secara 6iteratu dimana *router* mengirimkan sebuah copy-an lengkap dari *routing table*-nya ke semua *router* terdekat.
b. *Route Invalid Timer*, *Timer* ini menentukan jangka waktu yanga harus dilewati (180 detik) sebelum sebuah *router* menentukan bahwa sebuah rute menjadi tidak *valid*.

c. *Holddown Timer*, *Timer* ini men-set interval waktu di mana informasi *routing* ditahan (*holddown state*) , *default*nya adalah 180 detik.

d. *Route Flush Time*, *Timer* ini men-set waktu antara sebuah *route* menjadi tidak *valid* dan penghapusannya dari *routing table* (240 detik).

Update timer dilakukan secara priodik setiap 30 detik. Secara logika tentu saja hal tersebut akan memakan *bandwidth* yang cukup besar. Ketika terjadi *event triggered update* yaitu kondisi dimana terjadi perubahan *topology* secara tiba-tiba yang disebabkan adanya jalur yang putus atau mati maka penghitungan *invalid time* dan *hold down time* dimulai. Dalam selang waktu 180 detik, jalur yang putus akan ditandai sebagai *unreachable*/*invalid route* didaam *table routing*. Jika dalam selang waktu 180 detik jalur tersebut tidak memberikan respon, maka penghitungan *flush time* akan dilakukan selama 60 detik kedepan. Apabila dalam selang waktu 60 detik jalur tersebut tidak kunjung memberikan respon maka dilakukan penghapusan pada *table routing* terhadap jalur tersebut.

$HDT = 6 * UT$	$FT = HDT + (2 * UT)$	ket :
$HDT = 6 * 30$	$FT = 180 + (2 * 30)$	HDT = Hold Down Timer (detik)
$HDT = 180$	$FT = 180 + 60$	IT = Invalid Timer (detik)
$IT = HDT$	$FT = 240$	UT = Update Timer (detik)
$IT = 180$		FT = Flush Timer (detik)

Untuk menentukan jalur terbaik dalam pengiriman paket data, *Ripv2* melakukan perhitungan *metric* berdasarkan jumlah *hop*. Jalur yang memiliki jumlah *hop* terkecil yang akan dipilih sebagai jalur terbaik pada *protocol Ripv2*.

2.4.2.2. *Enhanced Interior Gateway Routing Protocol* (*EIGRP*)

EIGRP mengkombinasikan kelebihan-kelebihan yang dimiliki oleh *protocol routing link-state* dan *distance vector*. Tetapi pada dasarnya *eigrp* adalah *protocol distance vector* karena *router-router* yang menjalankan *eigrp* tidak mengetahui *road map*/*topology* jaringan secara menyeluruh seperti pada *protocol link-state* (Safitri, 2010).

EIGRP mudah dikonfigurasi seperti pendahulunya dan dapat diadaptasikan dengan variasi *topology network*. Penambahan fitur-fitur *protocol link-state* seperti *neighbor discovery* membuat *eigrp* menjadi *protocol distance vector* tingkat lanjut.

Protocol EIGRP melakukan *maintenance* 3 *table* pada proses *routing* yakni *routing table*, *topology table* dan *neighbor table*. Dalam perhitungan *metric*, *EIGRP* menggunakan 5 parameter yakni *bandwidth* ($k1$), *reliability* ($k2$), *delay* ($k3$), loading ($k4$) dan MTU ($k5$) yang disebut parameter *K value*. Secara *default* $k1 = 1, k2 = 0, k3 = 1, k4 = 0$, dan $k5 = 0$ yang menyebabkan perhitungan *metric* pada *EIGRP* secara *default* hanya menggunakan parameter $k1$ dan $k3$. Apbila $k2, k4$, dan $k5$ juga di set 1 maka *protocol EIGRP* akan melakukan kalkulasi tambahan untuk menghitung *metric* yang menyebabkan penghitungan ulang *metric* lebih sering dilakukan dan akan mengurangi keefisienan waktu penentuan jalur terbaik.

***Default Formula*:**

$M = BC + DC$	ket : M = Metric BC = Bandwidth Calculation DC = Delay Calculation

***Complete Formula* :**

$$M = \left[k1 * BC + \left(\frac{k2 * }{256} - load\right) + k3 * DC\right] * \left[\frac{}{rel + k4}\right]$$

ket:
$k1, k2, k3, k4, dan\ k5\ adalah\ konstanta$
$L = load\ dalam\ bps$
$rel = reliability\ dalam\ bps$

***Bandwidth Calculation* :**

missal pada *router* 1 *interface serial* 0/0/0 nya memiliki *bandwidth* 64 *Kbit* dan *router* 2 memiliki *bandwidth* 1024 *Kbit* pada *interface fastethernet* 0/0 maka untuk menghitung *slowest bandwith* dengan menggunakan rumus :

$$BC = \left(\frac{10.000.000}{SB}\right) * 256$$

$$BC = \left(\frac{10.000.000}{64}\right) * 256$$

$$BC = 40.000.000$$

ket :

10.000.000 merupakan referensi untuk bandwidth.

***Delay Calculation* :**

Misalnya pada *router* 1 *interface* serial 0/0/0 nya memiliki *delay* 20000 *ms* sedangkan *router* 2 memiliki *delay* 100 *ms* pada *interface fastethernet* 0/0, maka untuk menghitung *delay* :

maka untuk m

$$DC = \left(\frac{BD}{10}\right) * 256$$

$$DC = \left(\frac{20100}{10}\right) * 256$$

$$DC = 514560$$

ket :

20100 *merupakan hasil penjumlahan dari* 20000 *ms dan* 100 *ms.*

Sehingga di dapat *metric EIGRP* :

M = (40.000.000 + 514560)

M = 40514560

2.5. *QoS* (*Quality of Service*)

Saat pertama kali mendengar istilah *QoS* (*Qulity of Service*), sebagian besar orang mengartikan istilah terserbut sebagai kualitas dari suatu layanan. Istilah *QoS* sangat popular dan memiliki banyak arti dalam dunia teknologi bila dilihat dari perspektif yang berbeda misalnya dari segi *networking, application development,* dan lain sebagainya. Jika dilihat dari segi *networking, QoS* mengacu kepada kemampuan memberikan pelayanan berbeda kepada lalulintas jaringan dengan kelas-kelas yang berbeda. Tujuan akhir dari *QoS* adalah memberikan *network service* yang lebih baik dan terencana dengan *dedicated bandwidth, jitter* dan *latency* yang terkontrol dan meningkatkan *loss* karakteristik.

Flannagan dkk (2003) mendefinisikan bahwa *QoS* adalah teknik untuk mengelola *bandwidth*, *delay*, *jitter*, dan *packet loss* untuk aliran dalam jaringan. Tujuan dari mekanisme *QoS* adalah mempengaruhi setidaknya satu diantara empat parameter dasar *QoS* yang telah ditentukan. *Quality of Service* suatu *network* merujuk ke tingkat kecepatan dan keandalan penyampaian berbagai jenis beban data di dalam suatu komunikasi. Terdapat beberapa parameter *QoS*, yaitu:

1. *Delay*, merupakan waktu tunda ketika sebuah data menempuh jarak dari asal ke tujuan. *Delay* dapat dihitung dengan persamaan berikut :

$$Delay = request\ times - response\ time$$

$$Delay_{avrg} = \frac{\sum delay}{\sum packet\ received}$$

2. *Packt Loss*, merupakan kegagal transmisi *packet IP* mencapai tujuannya yang dapat terjadi akibat *overload* trafik, *congestion*, *error* pada media fisik, dan *overflow* yang terjadi pada *buffer*. Untuk menghitung nilai dari *packet loss* dapat digunakan persamaan berikut :

$$Packet\ Loss = \frac{Packet\ Sent - Packet\ received}{Packet\ Sent} \times 10^6$$

3. *Throughput* adalah kemampuan sebenarnya suatu *network* saat mengiriman data. *Throughput* dapat dihitung dengan rumus :

$$Throughput = \frac{Jumlah\ Data\ yang\ Dikirim\ (byte)}{Waktu\ untuk\ Mengirim\ Data\ (sec)}\ Bps$$

4. *Jitter* adalah *variant delay* antar paket yang terjadi pada jaringan *IP*. Semakin besar nilai *jitter* akan mengakibatkan nilai *QoS* semakin turun. *Jitter* dapat dihitung dengan rumus berikut :

$$Jitter = \frac{delay\ variant}{\sum packet\ received - 1}$$

$$\sum delay\ variant = (delay_2 - delay_1) + (delay_3 - delay_2) + \cdots (delay_n - delay\ (n-1))$$

Dalam implementasi jaringan *IP*, ke empat parameter *QoS* tersebut dapat mengalami 4 kategori penurunan performasi jaringan bila mengacu pada standarisasi yang dibuat oleh ETSI (European Telecommunications Standards Institute) pada *project TIPHON* (*Telecommunications and Internet Protocol Harmonization Over Network*). Keempat kategori tersebut dapat dilihat pada *table* berikut :

Table 2.1. Standarisasi nilai parameter *QoS project* TIPHON by ETSI

Delay	Jitter	Packet Loss	Throughput	Kualitas	Indeks
<150 ms	0 ms	0%	100%	Sangat Bagus	4
150-300 ms	0-75 ms	3%	75%	Bagus	3
300-450 ms	75-125 ms	15%	50%	Sedang	2
>450 ms	125-225	25%	<25%	Jelek	1

2.6. *GNS* 3 (*Graphical Network Simulator* 3)

GNS3 memungkinkan simulasi *network* yang kompleks. Untuk menyediakan simulasi lengkap dan akurat, GNS3 sangat terkait dengan:

- Dynamips , Cisco *IOS* emulator.
- Dynagen , berbasis teks *front end* untuk Dynamips.
- Qemu , sumber 11iterat dan *open* mesin *emulator* dan *virtualizer*.
- VirtualBox , sebuah *software* virtualisasi yang berjalan di GNS3.

GNS3 adalah alat pelengkap yang sangat baik untuk laboratorium nyata bagi *network engineer*, *administrator* dan orang-orang yang ingin belajar untuk sertifikasi seperti Cisco CCNA, CCNP, CCIP dan CCIE serta Juniper JNCIA, JNCIS dan JNCIE. Hal ini juga dapat digunakan untuk fitur eksperimen Cisco IOS, JUNOS Juniper atau untuk memeriksa konfigurasi yang perlu digunakan kemudian pada *router* nyata. *Software* ini bersifat *open source* yang dapat digunakan pada beberapa 11itera operasi yakni Windows, Linux, dan MacOS X.

Ada beberapa *router* simulator di pasaran, tetapi terbatas pada *command line*-nya. Banyak command atau parameternya yang tidak mendukung saat dijalankan pada *simulator*. Pada simulator tersebut kita hanya melihat representasi *output simulator router* disana, dimana ketepatan hasilnya dibuat oleh pembuat

software tersebut. Dengan *GNS3*, kita seperti menjalankan *router* sesungguhnya. Disini kita akan melihat secara nyata apa yang *IOS* Cisco hasilkan. Selain itu kita juga dapat mengakses beberapa command dan parameter yang didukung oleh *IOS* tersebut.

2.7. *VMWare* (*Virtual Machine Ware*)

VMWare merupakan *software* untuk mesin *virtual*. Fungsinya adalah untuk menjalankan banyak 12itera operasi dalam satu perangkat keras dan untuk menjalankan aplikasi yang ditujukan untuk 12itera operasi lainnya. Fungsi lainnya adalah untuk mempelajari suatu 12itera operasi baik ketika pada proses pembelajaran atau ketika proses pengembangan 12itera operasi dan lain sebagainya. *VMWare* memungkinkan bebarapa 12itera operasi dijalankan pada satu mesin *PC* tunggal secara bersamaan.

2.8. *Wireshark*

Wireshark merupakan salah satu dari sekian banyak tool *Network Analyzer* yang banyak digunakan oleh *Network Administrator* untuk menganalisa kinerja *network*-nya termasuk *protocol* didalamnya. *Wireshark* banyak disukai karena *interface*-nya yang menggunakan *Graphical User Interface* (*GUI*) atau tampilan grafis. Semua jenis paket informasi dalam berbagai format *protocol* akan dengan mudah ditangkap dan dianalisa. Karenanya tak jarang *tool* ini juga dapat dipakai untuk *sniffing* dengan menangkap paket-paket yang berjalan di dalam *network* dan menganalisanya.

Wireshark dapat membaca data secara langsung dari *Ethernet*, *Token-Ring*, *FDDI*, serial (PPP dan SLIP), 802.11 *wireless LAN*, dan koneksi ATM. Program ini juga sering digunakan oleh chatters untuk mengetahui ip korban maupun para *chatter* lainnya lewat *typingan room*. *Tool wireshark* dapat menganalisa transmisi paket data dalam *network*, proses koneksi dan transmisi data antar 12iteratu. Selama kita 12ite mendapatkan paket langsung dari *network*, dengan *tools* seperti *wireshark*, maka kita juga 12ite memanfaatkan *wireshark* untuk ‘menyadap’ pembicaraan *Voice Over IP*.