BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini, penulis mengambil beberapa referensi studi kepustakaan yang bersumber pada penelitian-penelitian sebelumnya. Hal ini berguna sebagai perbandingan bahan referensi dalam menyelesaikan penelitian ini.

Penelitian yang dilakukan oleh Imam Sutanto (2016), dalam penelitian ini dibuat Optimalisasi Jaringan Komputer Dengan Metode *Hierarchical Token Bucket Per Connection Queue* (PCQ) *Queue tree* Untuk Mendukung Kegiatan Perkuliahan PJJ Di Sekolah Tinggi Ilmu Kepolisian PTIK, menjelaskan tentang manajemen *bandwidth* dilakukan secara spesifik berdasarkan skala prioritas penggunaan akses jaringan komputer dan internet di lokasi penelitian. Sedangkan penelitian yang lakukan oleh peneliti menggunakan metode *queue tree* dengan yang diprioritaskan oleh kampus Universitas Islam Riau (UIR)

Penelitian yang dilakukan Imam Riadi (2010), dalam penelitian ini berjudul manajemen *bandwidth* dengan menggunakan *traffic shapping* menjelaskan tentang bagaimana memanajemen *bandwidth* menurut *traffic* yaitu *traffic* berat dan *traffic* ringan. Dan juga memanajemen *bandwidth* menurut prioritas berdasarkan tingkatan level yang lebih tinggi ke yang lebih rendah. Dan juga penelitian ini menggunakan VLAN untuk mempemudah pengelompokan dan memanajemen jaringannya. Sedangkan penelitian yang saya angkat yaitu

memanajemen berdasarkan prioritas yang telah di tentukan oleh kampus Universitas Islam Riau

Selanjutnya pada peneliti yang ketiga dilakukan oleh Burhanudin (2014) menjelaskan bagaimana mengkonfigurasi *queue tree* pada pc router mikrotik untuk melakukan manajemen *bandwidth* di PT.Tumbuh Selaras Alam, yang implementasinya hanya memanajemen *bandwidth* pada jaringan lokal area *network*. Sedangkan penelitian saya menggunakan metode *queue tree* dengan menambahkan berdasarkan prioritas menurut waktu, tempat, dan level *client* yang ada di kampus Universitas Islam Riau.

Berdasarkan penelitian yang dilakukan oleh peneliti di atas dapat disimpulkan bahwa terdapat perbedaan penelitian yang saya lakukan dengan penelitian sebelum yang telah dilakukan oleh beberapa peneliti.

2.2 Dasar Teori

2.2.1 Jaringan Komputer

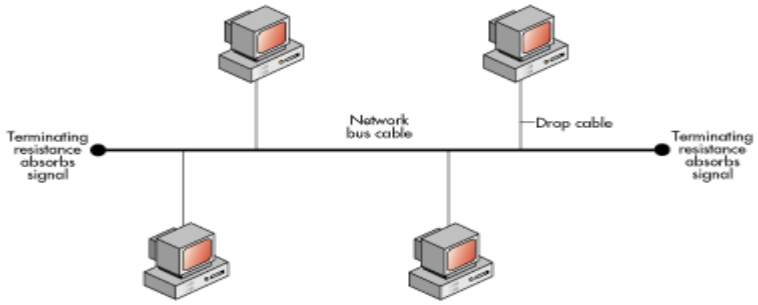
Jaringan komputer merupakan sekumpulan komputer *network* yang saling terhubung dan dapat saling berkomunikasi dengan media tertentu sebagai penghubungnya, yaitu dapat menggunakan kabel atau wireless. (Rendra dan Farhan, 2015) Ada tiga tipe yang membagi sebuah jaringan komputer berdasarkan besarnya area jangkauannya, yaitu:

2.2.1.1 Local Area *Network* (LAN)

Merupakan Jaringan Komputer yang memiliki jangkauan kecil, hanya sebatas dalam beberapa *network* saja tanpa harus ada koneksi internet dari luar, contohnya: laboratorium, Sekolah atau koneksi dalam satu gedung. Bentuk

minimal dari sebuah jaringan LAN adalah *peer to peer* yaitu jenis koneksi komputer yang hanya sebatas dua unit komputer saja.

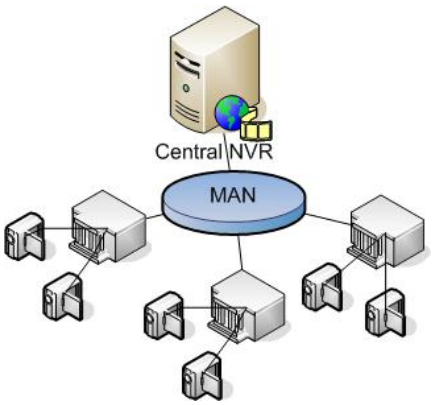
Dalam beberapa konfigurasi LAN tertentu ada komputer yang bertugas sebagai *server*, yang berfungsi untuk mengendalikan jaringan, serta dapat juga digunakan sebagai *gateway* penghubung ke internet jika jaringan lokal tersebut akan dikoneksikan dengan internet. Untuk komunikasi datanya dapat menggunakan kabel atau *wireless*. (Asiyah, 2013).



Gambar 2.1 *Local Area Network*

2.2.1.2 *Metropolitan Area Network* (MAN)

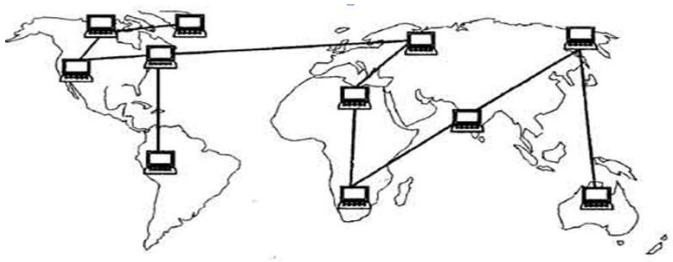
Merupakan bentuk topologi jaringan yang mirip dengan LAN akan tetapi memiliki jangkauan yang lebih luas daripada LAN, dalam hal ini jaringan LAN meliputi wilayah nasional atau daerah dalam satu cangkupan regional, jika pada LAN menggunakan kabel, maka pada MAN komunikasi dapat dilakukan dengan menggunakan kabel telepon atau nirkabel dengan menggunakan tower. (Asiyah, 2013).



Gambar 2.2 *Metropolitan Area Network*

2.2.1.3 *Wide Area Network* (WAN)

Suatu WAN meliputi area geografis yang lebih luas lagi, yang meliputi suatu negara atau dunia. Umumnya jaringan diletakkan pada banyak tempat yang berbeda (*server*), hal ini difungsikan untuk menghubungkan antar LAN yang lebih banyak lagi, dengan dihubungkan dengan dial up atau satelit. (Asiyah, 2013).



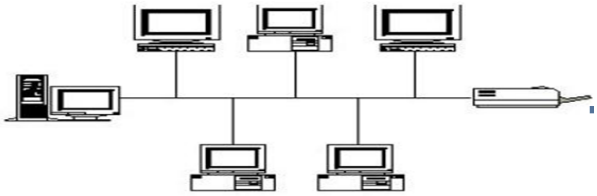
Gambar 2.3 *Wide Area Network*

2.2.2 Topologi jaringan

Topologi Jaringan adalah gambaran secara fisik dari pola hubungan antara komponen-komponen jaringan, yang meliputi *server*, *workstation*, *hub* dan pengkabelannnya. Terdapat tiga macam topologi jaringan umum digunakan, yaitu topologi *star*, Topologi *mesh*, Topologi *bus*. (Asiyah, 2013)

2.2.2.1 Topologi Bus

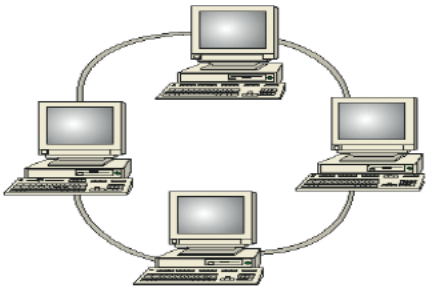
Topologi bus adalah arsitektur LAN linier di mana transmisi dari suatu peralatan jaringan dipropagasikan ke seluruh media dan diterima oleh seluruh *node* pada jaringan. Biasanya topologi ini dimanfaatkan pada implementasi jaringan Ethernet/IEEE 802.3, termasuk 100BaseT. Gambar 2.4 menunjukkan bentuk jaringan komputer dengan topologi *bus*. (Asiyah, 2013)



Gambar 2.4 Bentuk Jaringan Dengan Topologi *Bus*

2.2.2.2 Topologi *Ring*

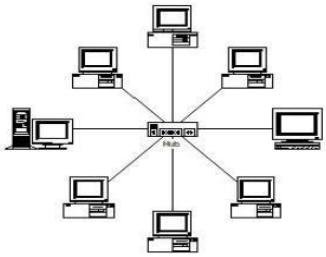
Topologi *Ring* adalah arsitektur LAN yang terdiri dari beberapa peralatan komputer yang terkoneksi melalui transmisi *unidirectional* membentuk suatu *closed-loop*. Gambar 2.5 menunjukkan bentuk jaringan komputer dengan topologi *ring*. (Asiyah, 2013)



Gambar 2.5 Bentuk Jaringan Dengan Topologi *Ring*

2.2.2.3 Topologi *Star*

Topologi *Star* adalah arsitektur LAN di mana *end points* dari jaringan terkoneksi ke sentral melalui *Hub* atau LAN *Switch* dengan *dedicated link*. Gambar 2.6 menunjukkan bentuk jaringan dengan topologi *star*. (Asiyah, 2013).



Gambar 2.6 Bentuk Jaringan Dengan Topologi *Star*

2.2.3 Router

Router berfungsi untuk mengatur aliran data dari satu jaringan ke jaringan yang yang lain. Dengan adanya *Router* maka arus data dari satu LAN (*Local Area Network*) dapat diisolasi dari arus LAN yang lain. Dengan demikian, arus data tidak bercampur-baur dengan arus data dari lan yang lain. Ada dua jenis *Router* yang biasa digunakan, *Router dedicated* yang merupakan keluaran dari pabrik dan router PC (Personal Komputer) *router* PC adalah komputer yang dibuat menjadi router. (Rendra, Farhan. 2015)

2.2.4 Sejarah Mikrotik

MikroTik adalah sebuah perusahaan kecil berkantor pusat di Latvia, bersebelahan dengan Rusia. Pembentukannya diprakarsai oleh John Trully dan Arnis Riekstins. John Trully adalah seorang berkewarganegaraan Amerika yang berimigrasi ke Latvia. Di Latvia ia bejumpa dengan Arnis, seorang sarjana

Fisika dan Mekanik sekitar tahun 1995. John dan Arnis mulai me-*routing* dunia pada tahun 1996 (misi *MikroTik* adalah me-*routing* seluruh dunia). Mulai dengan sistem *Linux* dan *MS-DOS* yang dikombinasikan dengan teknologi *Wireless-LAN (WLAN) Aeronet* berkecepatan 2Mbps di Moldova, negara tetangga Latvia, baru kemudian melayani lima pelanggannya di Latvia.

Prinsip dasar mereka bukan membuat *Wireless* ISP (W-ISP), tetapi membuat program *Router* yang handal dan dapat dijalankan di seluruh dunia. Latvia hanya merupakan tempat eksperimen John dan Arnis, saat itu mereka sudah membantu negara-negara lain termasuk Srilanka yang melayani sekitar 400 pengguna.

Linux yang pertama kali digunakan adalah Kernel 2.2 yang dikembangkan secara bersama-sama dengan bantuan 5-15 orang *staff Research and Development (R&D)* MikroTik yang sekarang menguasai dunia *routing* di negara-negara berkembang. Menurut Arnis, selain staf di lingkungan *MikroTik*, mereka juga merekrut tenaga-tenaga lepas dan pihak ketiga yang dengan intensif mengembangkan MikroTik secara *marathon* (https://id.wikipedia.org).

2.2.4.1 Jenis-Jenis Mikrotik

Mikrotik terdiri dari 2 (dua) jenis yaitu Mikrotik Router OS (software) dan Mikrotik Router Board (Hardware).

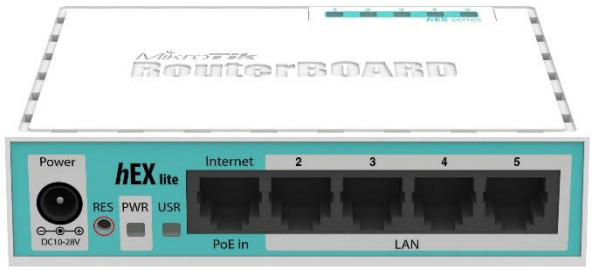
2.2.4.1.1 Mikrotik Router OS

MikroTik Router OS merupakan sistem operasi dan perangkat lunak yang dapat digunakan untuk menjadikan komputer manjadi router *network* yang

handal, mencakup berbagai fitur yang dibuat untuk IP *network* dan jaringan wireless, cocok digunakan oleh ISP dan provider hotspot (Muhammad Syarif Pagala. 2017).

2.2.4.1.2 *Mikrotik Routerboard*

Mikrotik *routerboard* merupakan sebuah perangkat jaringan komputer yang menggunakan Mikrotik RouterOS yang berbasis Linux dan diperuntukkan bagi *network* router. Mikrotik *routerboard* memiliki beberapa fasilitas seperti *bandwith management, stateful firewall, hotspot for plug and play access, remote* Winbox GUI *admin*, dan *routing*. Administrasi Mikrotik *routerboard* bisa dilakukan melalui *Windows application* (Winbox). Pada saat ini, Winbox telah di tampilkan secara *graphical*, sehingga *user* dengan mudah dapat mengakses dan mengkonfigurasi router sesuai kebutuhan dengan mudah efektif dan efisien. Memperkecil kesalahan pada waktu setup konfigurasi, mudah dipahami dan customable sesuai yang diinginkan. (Cangging Anjika Pamungkas. 2016)



Gambar 2.7 Mikrotik *Routerboard*

2.2.4.1.3 Fitur-Fitur *Mikrotik*

Adapun fitur-fitur dari MikroTik Router OS adalah sebagai berikut (https://id.wikipedia.org):

1. *Address List*: Pengelompokan IP Address berdasarkan nama.
2. Asynchronous: Mendukung *serial* PPP *dial-in/dial-out*, dengan otentikasi CHAP, PAP, MSCHAPv1 dan MSCHAPv2, Radius, *dial on demand*, modem *pool* hingga 128 ports.
3. Bonding: Mendukung dalam pengkombinasian beberapa antar muka Ethernet ke dalam 1 pipa pada koneksi cepat.
4. ISDN: Mendukung *dial-in/dial-out*. Dengan otentikasi PAP, CHAP, MSCHAPv1 dan MSCHAPv2, Radius. Mendukung 182K bundle, Cisco HDLC, x751, x75ui, 75bui *line* protokol.
5. M3P: MikroTik *Protocol Paket Packer* untuk *wireless links* dan ethernet.
6. MNDP: MikroTik *Discovery Neighbour Protokol*, juga mendukung Cisco Discovery Protokol (CDP).
7. *Monitoring / Accounting*: Laporan *Traffic* IP, log, statistik *graph* yang dapat

a. di aksen melalui HTTP.

8. NTP: *Network Time Protocol* untuk *server* dan *clients*: *sinkronisasi* menggunakan sistem GPS.
9. *Point to Point* Tunneling Protocol: PPTP, PPPoE, dan L2TP AccessConsetrator, protocol otentikasi menggunakan PAP, CHAP, MSCHAPv1, MSCHAPv2; otentikasi dan laporan Radius; enkripsi MPPE; kompresi untuk PpoE; limitrate.
10. *Proxy* : Cache untuk FTP dan HTTP *proxy* server, HTTPS *proxy* , transparent*proxy* untuk DNS dan HTTP; mendukung protocol SOCKS; mendukung parent*proxy* ; static DNS.

11. Routing: Routingstatic dan dinamik; RIPv1/v2, OSP v2, BGP v4.
12. Simple Tunnel: Tunnel IPIP dan EoIP (Ethernetover IP)
13. SDSL: Mendukung *Single Line* DSL; mode pemutusan jalur koneksi dan jaringan.
14. SNMP: *Simple Network Monitoring Protocol* mode akses read-only.
15. Synchronous : V.35, V.24, E1/T1, X21, DS3 (T3) media types; sync-PPP, Cisco HDLC; Frame Relay line protokol; ANSI-617d (ANDI atau annex D) dan Q933a (CCITT atau annex A); Frame Relay jenis LMI.
16. Tool: Ping, Traceroute; *bandwidth test; pingflood; telnet; SSH; packetsniffer;*
17. *Dinamik DNS updat*e.
18. UPnP: Mendukung antarmuka *Universal Plug and Play*.
19. VLAN: Mendukung Virtual LAN IEEE 802.1q untuk jaringan ethernet dan *wireless*; multiple VLAN; VLAN *bridging.*
20. VoIP: Mendukung aplikasi *voiceover* IP.
21. VRRP: Mendukung Virtual Router Redudant Protocol.
22. Winbox: Aplikasi mode GUI untuk meremote dan mengkonfigurasi MikroTik Router OS.

2.2.5 *Protocol*

Menurut Zaenal Arifin (2005) *protocol* merupakan aturan-aturan dan prosedur untuk melakukan komunikasi. Ketika beberapa komputer dalam sebuah jaringan hendak melakukan komunikasi dengan komputer lain, aturan-aturan atau prosedur komunikasi harus dilakukan terlebih dahulu. Atuan-aturan tersebut dikenal dengan istilah protocol.

Beberapa hal yang perlu kita pahami tentang *protocol* dalam sebuah lingkungan jaringan *computer* adalah sebagai berikut:

1. Dalam sistem jaringan komputer terdapat beberapa jenis *protocol*, masing-masing memiliki tujuan dan tugas yang berbeda. Setiap *protocol* memiliki kelebihan dan kekurangan.
2. Beberapa *protocol* dapat saling bekerjasama. Hal ini dikenal dengan istilah *protocol stack* atau *protocol suite*.

2.2.5.1 Transmission Control Protocol (TCP/IP)

Menurut Zaenal Arifin (2005) TCP/IP (*Transport Control Protocol/Internet Protocol*) merupakan sebuah *protocol suite standard* yang menyediakan komunikasi dalam sebuah lingkungan (sistem oprerasi) yang beragam. TCP/IP bersifat *routeable* dan merupakan *protocol* yang biasa di dalam jaringan global (*enterprise*). Terdapat beberapa *protocol* yang berjalan menggunakan *protocol* TCP/IP, diantaranya:

1. FTP (*File Transfer Protocol*) – digunakan untuk melakukan mekanisme pengiriman file.
2. SMTP (*Simple Mail Transfer Protocol*) – digunakan untuk pengiriman *e-mail*, dan lain-lain.

2.2.5.2 Internet Protocol (IP)

IP *address* (IP) adalah alamat yang diberikan pada jaringan komputer dan peralatan jaringan yang menggunakan *protokol* TCP/IP. IP *address* terdiri atas 32

bit angka biner yang dapat dituliskan sebagai empat kelompok angka desimal yang dipisahkan oleh tanda titik seperti 192.168.0.1. (Andi Micro, 2012)

Ada beberapa kelas dalam IP *address*:

1. *Class* A

Dalam jaringan *class* A *byte* pertama digunakan untuk menujukkan alamat *network*, dan tiga *byte* sisanya digunakan untuk alamat *host. Class*A ini, *bit* pertama harus selalu *off* atau bernilai 0. Ini berarti alamat class A adalah semua nilai antara 0 dan 127. Formatnya adalah *network.host.host.host*, atau jika digantikan dengan binary akan menjadi: 0XXXXXXX.*host.host.host*. Jika pada *byte* pertama tanda ’X’ di ganti dengan 0 maka akan menjadi: 00000000=0 Dan jika tanda ’X’ diganti dengan 1 maka akan menjadi: 01111111=127.

2. *Class* B

Pada jaringan *class* B, dua *byte* pertama menunjukan alamat *network* dan dua *byte* selebihnya digunakan untuk alamat *host.*Pada *class*B, bit pertama harus selalu dalam kondisi *on*, tapi bit kedua harus selalu dalam kondisi *off*. Ini berarti alamat *class* B adalah semua nilai antara 128 dan 191.

Formatnya adalah *network.network.host.host*, atau jika digantikan dengan binari akan menjadi : 10XXXXXX.XXXXXXXX.host.host, Jika pada *byte* pertama tanda ’X’ diganti dengan 0 maka akan menjadi : 10000000=128 Dan jika tanda ’X’ diganti dengan 1 maka akan menjadi : 10111111=191.

3. *Class* C

Tiga byte pertama dari pengalamatan jaringan *class* C digunakan untuk alamat *network*, dengan hanya menyisakan satu byte kecil untuk alamat *host.*Pada class ini, 2 bit pertama dari *byte* pertama harus selalu dalam kondisi *on*, tapi *bit* ketiga harus selalu dalam kondisi *off*. Ini berarti alamat *class* C adalah semua nilai antara 192 dan 223.Formatnya adalah *network.network.network.host*, atau jika diganti-kan dengan binari akan menjadi : 110XXXXX.XXXXXXXX.XXXXXXXX.host Jika pada byte pertama tanda ’X’ diganti dengan 0 maka akan menjadi : 11000000=192 Dan jika tanda ’X’ diganti dengan 1 maka akan menjadi : 11011111=223.

2.2.6 Routing

Routing adalah proses pemilihan jalur di jaringan yang digunakan untuk mengirimkan paket data ke alamat tujuan. *Router* membuat keputusan *routing* berdasarkan IP *address* tujuan dari paket. Istilah *routing* digunakan untuk pemilihan jalur sebuah paket dari sebuah jaringan ke jaringan lain yang saling terhubung melalui *router*. *Router* hanya memperhatikan *network* tujuan dan jalur terbaik untuk menuju ke *network* tujuan. (Iwan Sofana, 2012)

Routing ada 2 jenis, yaitu:

2.2.6.1 Routing Statis

Static routing (Routing Statis) adalah sebuah *router* yang memiliki tabel routing statik yang di setting secara manual oleh para administrator jaringan. Routing static pengaturan *routing* paling sederhana yang dapat dilakukan pada

jaringan komputer. Menggunakan *routing* statik murni dalam sebuah jaringan berarti mengisi setiap entri dalam *forwarding table* disetiap *router* yang berada di jaringan tersebut.

2.2.6.2 Routing Dinamis

Dynamic Routing (Router Dinamis) adalah sebuah *router* yang memiliki dan membuat tabel *routing* secara otomatis, dengan mendengarkan lalu lintas jaringan dan juga dengan saling berhubungan antara *router* lainnya. *Protokol routing* mengatur *router-router* sehingga dapat berkomunikasi satu dengan yang lain dan saling memberikan informasi satu dengan yang lain dan saling memberikan informasi routing yang dapat mengubah isi *forwarding table*, tergantung keadaan jaringannya.

Dengan cara ini, *router-router* mengetahui keadaan jaringan yang terakhir dan mampu meneruskan data ke arah yang benar. Dengan kata lain, *routing* dinamik adalah proses pengisian data *routing* di *table routing* secara otomatis.

2.2.6.3 Subnetting

Menurut Gin-Gin Yugianto (2012) teknik *subnet* merupakan cara untuk membagi jaringan menjadi jaringan-jaringan yang lebih kecil dengan cara mengubah *subnet mask*, yaitu dengan cara meminjam *bit host*menjadi *bit network*. Teknik *subnet* menjadi penting bila sebuah jaringan memiliki alokasi IP yang terbatas.

Untuk melakukan *subnetting* terdiri dari beberapa proses, yaitu:.

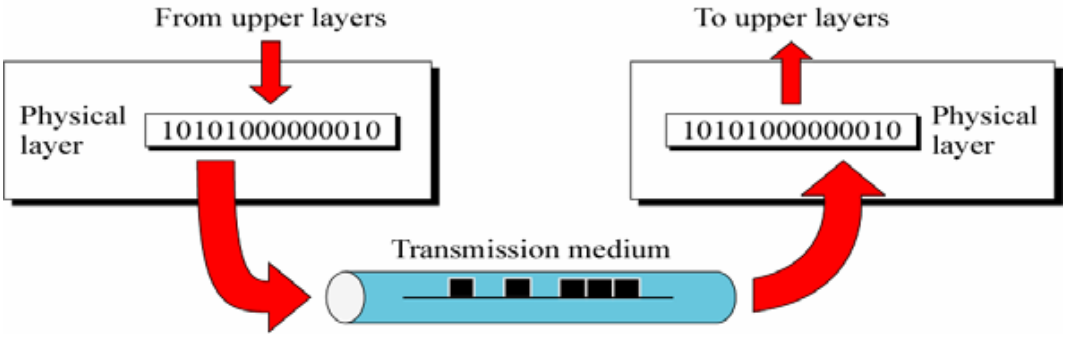
1. Menentukan jumlah *subnet* yang dihasilkan oleh subnet mask Yaitu dengan cara memakai rumus 2x – 2 = jumlah *subnet*, x adalah *bit* 1 pada *subnet mask*. Misalnya 11000000, maka x adalah 2 (dilihat dari jumlah angka 1 yang ada di situ), 22 – 2 = 2 *subnet*.
2. Menentukan jumlah *host* per subnetYaitu dengan memakai rumus 2y – 2 = jumlah *host* per *subnet*, y adalah jumlah *bit* dibagian *host* atau yang bernilai nol '0' Misalnya 11000000,maka y adalah 6 (dilihat dari angka 0 yang ada disitu), 26 – 2 = 62 host.
3. Menentukan *subnet* yang valid Yaitu dengan mengurangi 256 dengan angka yang ada dibelakang *subnet mask*, misalnya 255.255.255.224/27, maka untuk menentukan *subnet* yang valid 256 – 224 = 32. Hasil dari pengurangan ditambahkan dengan bilangan itu sendiri sampai berjumlah sama dengan angka belakang *subnet mask*. 32 + 32 =64, 64 + 32 = 96, 96 + 32 = 128, 128 + 32 = 160, 160 + 32 = 192, 192 + 32 = 224. Maka jumlah *host* yang valid adalah 32, 64, 96, 128, 160, 192.
4. Menentukan alamat *broadcast* untuk tiap *subne*t Yaitu mengambil alamat ip address yang terletak paling akhir.
5. Menetukan *host-host* yang valid untuk tiap *subnet* Yaitu mengambil nomor diantara *subnet-subnet* dengan menghilangkan angka 0.

2.2.7 OSI (Open System Interconnection) Layer

2.2.7.1 Phisycal Layer

Menurut Melwin Syafrizal, S.Kom. M.Eng lapisan ini bertanggung jawab untuk mengaktifkan dan mengatur physical interface jaringan komputer. Pada

lapisan ini, hubungan antara interface-interface dari perangkat keras diatur seperti hubungan antara DTE dan DCE. Interface yang didefenisikan pada lapisan ini antara lain: 10BaseT, 100BaseTX, V35, X.21 dan High Speed Serial *Interface* (HSSI).



Gambar 2.8 *Transmission* Data Pada *Physical Layer*

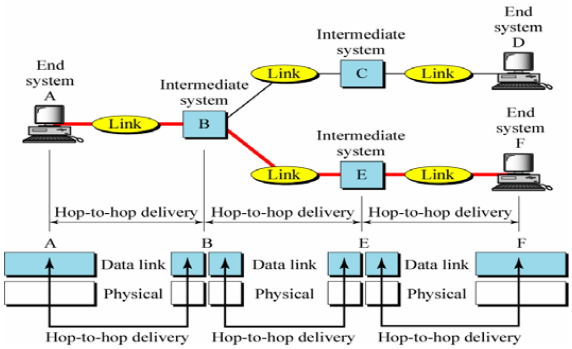
Physical Layer berfungsi dalam pengiriman *raw bit* ke *channel* komunikasi. Masalah desain yang harus diperhatikan disini adalah memastikan bahwa bila satu sisi mengirim data 1 *bit*, data tersebut harus diterima oleh sisi lainnya sebagai 1 *bit* pula, dan bukan 0 *bit*. Secara umum masalah-masalah desain yang ditemukan disini berhubungan secara mekanik, elektrik dan *interface* prosedural, juga media fisik yang berada di bawah *physical layer*.

2.2.7.2 Data Link Layer

Lapisan ini mengatur topologi jaringan, *error notification* dan *flow control*. Tugas utama data *link layer* adalah sebagai fasilitas transmisi *raw* data dan mentransformasi data tersebut ke saluran yang bebas dari kesalahan transmisi.

Sebelum diteruskan ke *network layer*, data *link layer* melaksanakan tugas ini dengan memungkinkan pengirim memecah-mecah data *input* menjadi

sejumlah data *frame* (biasanya berjumlah ratusan atau ribuan byte). Kemudian data *link layer* mentransmisikan *frame* tersebut secara berurutan, dan memproses *acknowledgement frame* yang dikirim kembali oleh penerima.

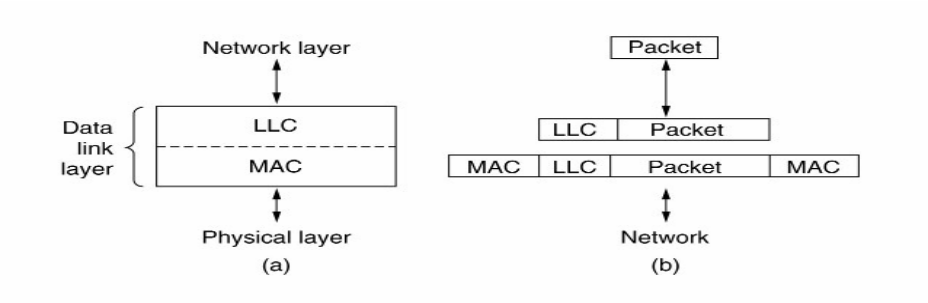


Gambar 2.9 *Transmission* Data Pada *Data Link Layer*

Karena *physical layer* menerima dan mengirim aliran *bit* tanpa mengindahkan arti atau *arsitektur frame*, maka tergantung pada *data link* layerlah untuk membuat dan mengenali batas-batas *frame* itu. Hal ini bisa dilakukan dengan cara membubuhkan *bit* khusus ke awal dan akhir *frame*. Bila secara insidental pola-pola *bit* ini bisa ditemui pada data, maka diperlukan perhatian khusus untuk meyakinkan bahwa pola tersebut tidak secara salah dianggap sebagai batas-batas *frame.*

Terjadinya *noise* pada saluran dapat merusak *frame*. Dalam hal ini, perangkat lunak data *link layer* pada mesin sumber dapat mengirim kembali *frame* yang rusak tersebut. Akan tetapi transmisi *frame* sama secara berulang-ulang bisa menimbulkan duplikasi *frame*. *Frame* duplikat perlu dikirim apabila *acknowledgement frame* dari penerima yang dikembalikan ke pengirim telah hilang. Tergantung pada *layer* inilah untuk mengatasi masalah-masalah yang

disebabkan rusaknya, hilangnya dan duplikasi *frame*. Data *link layer* menyediakan beberapa kelas layanan bagi *network layer*. Kelas layanan ini dapat dibedakan dalam hal kualitas dan harganya.



Gambar 2.10 Interaksi Antar *Data Link Layer* Dengan *Layer* Diatas Dan Dibawahnya

Masalah-masalah lainnya yang timbul pada data *link layer* dan juga sebagian besar *layer-layer* diatasnya adalah mengusahakan kelancaran proses pengiriman data dari pengirim yang cepat ke penerima yang lambat.

Mekanisme pengaturan lalu-lintas data harus memungkinkan pengirim mengetahui jumlah ruang *buffer* yang dimiliki penerima pada suatu saat tertentu. Seringkali pengaturan aliran dan penanganan *error* ini dilakukan secara terintegrasi.

Saluran yang dapat mengirim data pada kedua arahnya juga bisa menimbulkan masalah. Sehingga dengan demikian perlu dijadikan bahan pertimbangan bagi *software* data *link layer*. Masalah yang dapat timbul di sini adalah bahwa *frame-frame acknowledgement* yang mengalir dari A ke B bersaing saling mendahului dengan aliran dari B ke A. Penyelesaian yang terbaik (*piggy backing)* telah bisa digunakan. *Switch* dan *bridge* bekerja dilapisan data link ini.

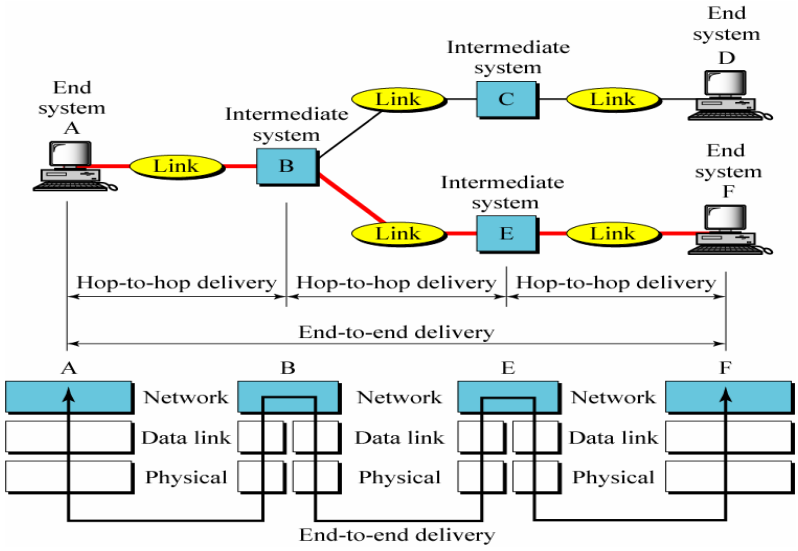
Jaringan *broadcast* memiliki masalah tambahan pada data *link layer*. Masalah tersebut adalah dalam hal mengontrol akses ke saluran yang dipakai bersama. Untuk mengatasinya dapat digunakan *sublayer* khusus data *link layer*, yang disebut *medium access sublayer*.

2.2.7.3 *Network* Layer

Network layer berfungsi untuk pengendalian operasi *subnet* dengan meneruskan paket-paket dari satu *node* ke *node* lain dalam jaringan. Masalah desain yang penting adalah bagaimana caranya menentukan *route* pengiriman paket dari sumber ke tujuannya.

Route dapat didasarkan pada *table statik* yang dihubungkan ke *network*. *Route* juga dapat ditentukan pada saat awal percakapan misalnya session terminal. *Route* juga sangat dinamik, dapat berbeda bagi setiap paketnya, dan karena itu, *route* pengiriman sebuah paket tergantung beban jaringan saat itu.

Bila pada saat yang sama dalam sebuah *subnet* terdapat terlalu banyak paket maka ada kemungkinan paket-paket tersebut tiba pada saat yang bersamaan. Hal ini dapat menyebabkan terjadinya *bottleneck* (penyempitan di bagian ujung, seperti leher botol).Pengendalian kemacetan seperti itu juga merupakan tugas *network layer.*



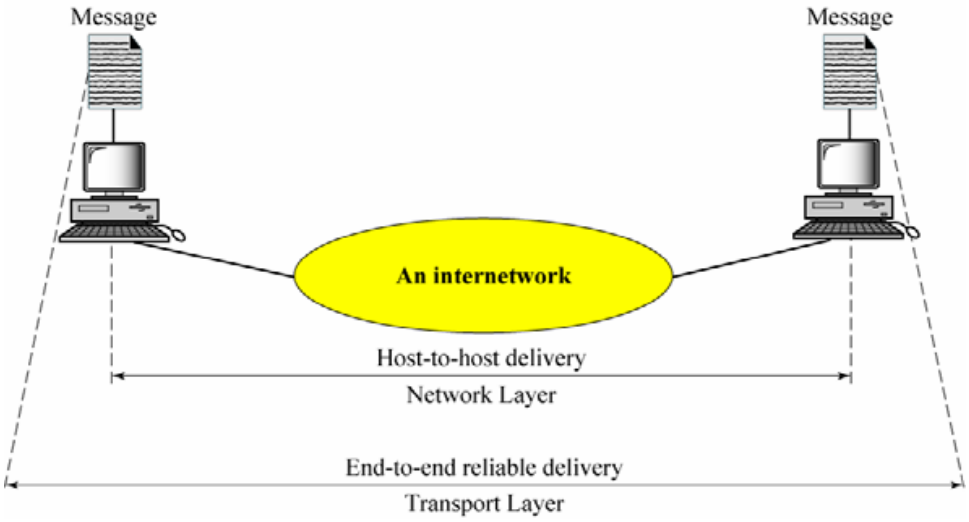
Gambar 2.11 *Transmission* Data Pada *Network Layer*

Karena operator *subnet* mengharap bayaran yang baik atas tugas pekerjaannya. Sering kali terdapat beberapa fungsi *accounting* yang dibuat pada *network layer*. Untuk membuat informasi tagihan, setidaknya *software* mesti menghitung jumlah paket atau karakter atau bit yang dikirimkan oleh setiap pelanggannya. *Accounting* menjadi lebih rumit, bilamana sebuah paket melintasi batas negara yang memiliki tarif yang berbeda.

Perpindahan paket dari satu jaringan ke jaringan lainnya juga dapat menimbulkan masalah yang tidak sedikit. Cara pengalamatan yang digunakan oleh sebuah jaringan dapat berbeda dengan cara yang dipakai oleh jaringan lainnya. Suatu jaringan mungkin tidak dapat menerima paket sama sekali karena ukuran paket yang terlalu besar. Protokolnyapun bisa berbeda pula, demikian juga dengan yang lainnya. *Network layer* telah mendapat tugas untuk mengatasi semua masalah seperti ini, sehingga memungkinkan jaringan-jaringan yang berbeda untuk saling terinterkoneksi.

2.2.7.4 Transport Layer

Fungsi dasar *transport layer* adalah menerima data dari *session layer*, memecah data menjadi bagian-bagian yang lebih kecil bila perlu, meneruskan data ke *network layer*, dan menjamin bahwa semua potongan data tersebut 26ias tiba di sisi lainnya dengan benar. Selain itu, semua hal tersebut harus dilaksanakan secara efisien, dan bertujuan dapat melindungi *layer-layer* bagian atas dari perubahan teknologi *hardware* yang tidak dapat dihindari.



Gambar 2.12 *Transmission* Data Pada *Transport Layer*

Dalam keadaan normal, *transport layer* membuat koneksi jaringan yang berbeda bagi setiap koneksi *transport* yang diperlukan oleh *session layer*. Bila koneksi *transport* memerlukan *throughput* yang tinggi, maka *transport layer* dapat membuat koneksi jaringan yang banyak. *Transport layer* membagi-bagi pengiriman data ke sejumlah jaringan untuk meningkatkan *throughput*.

Dilain pihak, bila pembuatan atau pemeliharaan koneksi jaringan cukup mahal, *transport layer* dapat menggabungkan beberapa koneksi *transport* ke

koneksi jaringan yang sama. Hal tersebut dilakukan untuk membuat penggabungan ini tidak terlihat oleh *session layer*.

2.2.7.5 Session Layer

Session layer mengijinkan para pengguna untuk menetapkan *session* dengan pengguna lainnya. *Layer* ini membuka, mengatur dan menutup suatu *session* antara aplikasi-aplikasi.

Sebuah *session* selain memungkinkan *transport* data biasa, seperti yang dilakukan oleh *transport layer*, juga menyediakan layanan yang istimewa untuk aplikasi-aplikasi tertentu. Sebuah *session* digunakan untuk memungkinkan seseorang pengguna log ke *remote timesharing system* atau untuk memindahkan file dari satu mesin ke mesin lainnya.

Sebuah layanan *session layer* yang lain adalah untuk melaksanakan pengendalian dialog. *Session* dapat memungkinkan lalu lintas bergerak dalam bentuk dua arah pada suatu saat, atau hanya satu arah saja. Jika pada satu saat lalu lintas hanya satu arah saja (analog dengan rel kereta api tunggal), *session layer* membantu untuk menentukan giliran yang berhak menggunakan saluran pada suatu saat.

Layanan *session* di atas disebut manajemen token. Untuk sebagian protokol, adalah penting untuk memastikan bahwa kedua pihak yang bersangkutan tidak melakukan operasi pada saat yang sama. Untuk mengatur aktivitas ini, *session layer* menyediakan token-token yang dapat digilirkan. Hanya pihak yang memegang token yang diijinkan melakukan operasi kritis.

Layanan *session* lainnya adalah sinkronisasi. Ambil contoh yang dapat terjadi ketika mencoba *transfer file* yang berdurasi 2 jam dari mesin yang satu ke mesin lainnya dengan kemungkinan mempunyai selang waktu 1 jam antara dua *crash* yang dapat terjadi. Setelah masing-masing *transfer* dibatalkan, seluruh *transfer* mungkin perlu diulangi lagi dari awal, dan mungkin saja mengalami kegagalan lain.

Untuk mengurangi kemungkinan terjadinya masalah ini, *session layer* dapat menyisipkan tanda tertentu ke aliran data. Karena itu bila terjadi *crash*, hanya data yang berada sesudah tanda tersebut yang akan ditransfer ulang.

Protokol yang berfungsi pada lapis *session* ini antara lain adalah: NFS, NETBEUI, RPC, SQL, X *Windows System, Apple Talk Session Protocol* (ASP), dan *Digital Network Architecture Session Control Program* (DNASCP).

2.2.7.6 Presesntation Layer

Presentation layer melakukan fungsi-fungsi tertentu yang diminta untuk menjamin penemuan sebuah penyelesaian umum bagi masalah tertentu. Selain memberikan sarana-sarana pelayanan untuk konversi, format dan enkripsi data, presentation layer juga bekerja dengan file berformat ASCII, EBCDIC, JPEG, MPEG, TIFF, PICT, MIDI, dan Quick Time.

Presentation layer tidak mengizinkan pengguna untuk menyelesaikan sendiri suatu masalah. Tidak seperti *layer-layer* di bawahnya yang hanya melakukan pemindahan *bit* dari satu tempat ke tempat lainnya, *presentation layer* memperhatikan *syntax* dan semantik informasi yang dikirimkan.

Satu contoh layanan *presentation* adalah *encoding data*. Kebanyakan pengguna tidak memindahkan *string* bit biner yang random. Para pengguna saling bertukar data sperti nama orang, tanggal, jumlah uang, dan tagihan. *Item-item* tersebut dinyatakan dalam bentuk *string* karakter, bilangan *interger*, bilangan *floating point*, struktur data yang dibentuk dari beberapa *item* yang lebih sederhana. Terdapat perbedaan antara satu komputer dengan komputer lainnya dalam memberi kode untuk menyatakan *string* karakter (misalnya, ASCII dan Unicode), integer (misalnya komplemen satu dan komplemen dua), dsb.

Untuk memungkinkan dua buah komputer yang memiliki *presentation* yang berbeda untuk dapat berkomunikasi, struktur data yang akan dipertukarkan dapat dinyatakan dengan cara abstrak, sesuai dengan ”*encoding standard*” yang akan digunakan “pada saluran”. *Presentation layer* mengatur data struktur abstrak ini dan mengkonversi dari representation yang digunakan pada sebuah komputer menjadi ”*representation standard*” jaringan, dan sebaliknya

2.2.7.7 Application Layer

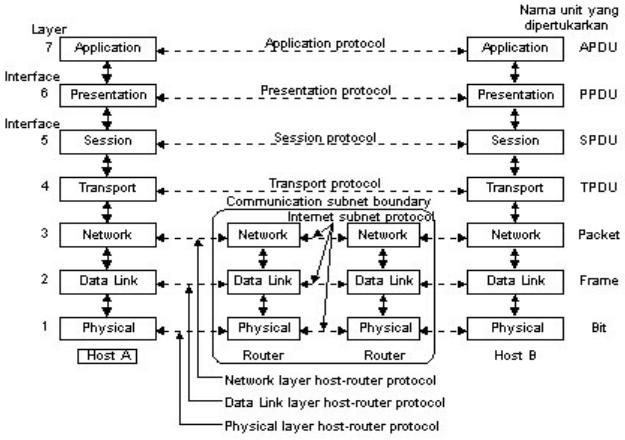
Lapisan ini bertugas memberikan sarana pelayanan langsung ke user, yang berupa aplikasi-aplikasi dan mengadakan komunikasi dari program ke program. Jika kita mencari suatu *file* dari *file server* untuk digunakan sebagai aplikasi pengolah kata, maka proses ini bekerja melalui layer ini. Demikian pula jika kita mengirimkan *e-mail, browse* ke internet, *chatting*, membuka telnet *session*, atau menjalankan FTP, maka semua proses tersebut dilaksanakan di *layer* ini.

Application layer terdiri dari bermacam-macam protokol. Misalnya terdapat ratusan jenis terminal yang tidak kompatibel di seluruh dunia, kemudian

kita memerlukan aplikasi yang diharapkan bekerja pada jaringan dengan bermacam-macam terminal, yang masing-masing memiliki *layout* layar yang berlainan, mempunyai cara urutan penekanan tombol yang berbeda untuk penyisipan dan penghapusan teks, memindahkan sensor dan sebagainya.

Suatu cara untuk mengatasi masalah seperti di atas, adalah dengan menentukan terminal virtual jaringan abstrak, sehingga editor dan program-program lainnya dapat ditulis agar saling bersesuaian. Untuk menangani setiap jenis terminal, satu bagian *software* harus ditulis untuk memetakan fungsi terminal virtual jaringan ke terminal sebenarnya.

Fungsi lainnya adalah pemindahan *file*. Sistem *file* yang satu dengan lainnya memiliki konvensi penamaan yang berbeda, cara menyatakan baris-baris teks yang berbeda, dan sebagainya. Perpindahan *file* dari sebuah sistem ke sistem lainnya yang berbeda memerlukan penanganan untuk mengatasi adanya ketidak-kompatibel-an ini. Tugas tersebut juga merupakan pekerjaan *application layer*, seperti pada surat elektronik, *remote job entry*, *directory lookup*, dan berbagai fasilitas bertujuan umum dan fasilitas bertujuan khusus lainnya.

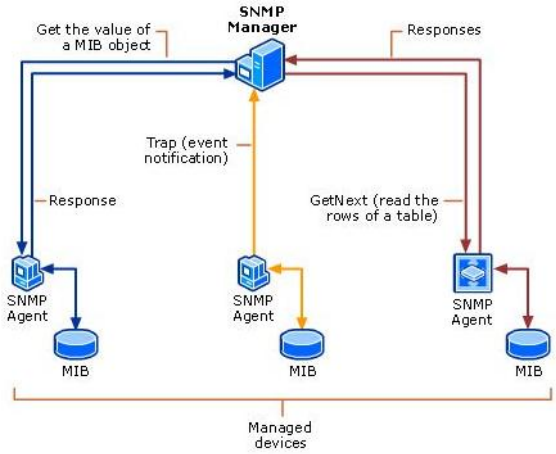


Gambar 2.13 Contoh Tentang Bagaimana Model OSI Digunakan

Layer ini layer aplikasi dimana aplikasi pemakai diletakan. Biasanya aplikasi layer bekerjama sama dengan presentasi untuk diterapkan pada sistem komunikasi data. (Jonathan Lukas. 2006)

2.2.7.7.1 *Simple Network Management Protokol* (SNMP)

Simple Network Management Protocol (SNMP) adalah sebuah protokol yang dirancang untuk memberikan kemampuan kepada pengguna untuk memonitor dan mengatur suatu jaringan komputer dari jarak jauh (secara *remote*) atau dalam satu pusat kontrol saja. Dengan menggunakan protokol ini bisa didapatkan informasi tentang status dan keadaan dari suatu jaringan. Protokol ini menggunakan *transport* UDP pada port 161. Pengolahan ini dijalankan dengan mengumpulkan data dan melakukan penetapan terhadap variabel-variabel dalam elemen jaringan yang dikelola Dalam aplikasinya, Elemen SNMP terdiri dari tiga bagian, yaitu *manager, agent*, dan MIB. *Manager* merupakan *software* yang berjalan di sebuah *host* di jaringan, yang merupakan suatu proses atau lebih yang berkomunikasi dengan *agent* dalam jaringan. *Agent* merupakan perangkat lunak yang dijalankan disetiap elemen jaringan yang dikelola. *Agent* terdapat pada, *workstation, repeater*, router, switch, dan personal *computer*, bertugas untuk merespon dan memberikan informasi sesuai permintaan *manager* SNMP. Manager Information Base (MIB) merupakan struktur database variabel dari elemen jaringan yang dikelola . Pendefinisian MIB dalam SNMP menggunakan diagram pohon, dan menempatkan setiap Object Identifier (OID) pada suatu lokasi unik pada pohon (Muazam Nugroho, Achmad Affandi, dan Djoko Suprajitno Rahardjo**.** 2014)



Gambar 2.14 Simple *Network* Management Protokol

2.2.8 *Bandwidth*

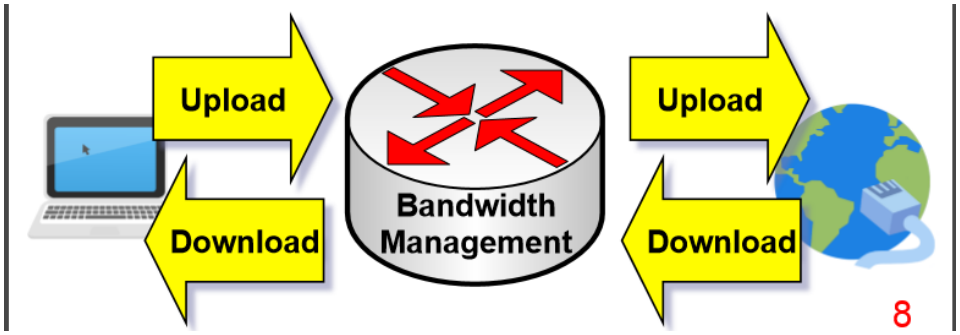
2.2.8.1 Pengertian *Bandwidth*

Di dalam jaringan komputer, *Bandwidth* sering digunakan sebagai suatu sinonim untuk *data transfer rate* yaitu jumlah data yang dapat dibawa dari sebuah titik ke titik lain dalam jangka waktu tertentu (pada umumnya dalam detik). Jenis *Bandwidth* ini biasanya diukur dalam *bits per second (bps)*. Adakalanya juga dinyatakan dalam *bytes persecond (Bps).* Secara umum, koneksi dengan *Bandwidth* yang besar/tinggi memungkinkan pengiriman informasi yang besar seperti pengiriman gambar/*images* dalam *video presentation*. (www.mikrotik.co.id 2017)

2.2.8.2 Manajemen *Bandwidth*

Manajemen *Bandwidth* adalah suatu cara yang dapat digunakan untuk *management* dan mengoptimalkan berbagai jenis jaringan dengan menerapkan layanan *Quality Of Service (QoS)* untuk menetapkan tipe-tipe lalulintas jaringan.

QoS adalah kemampuan untuk menggambarkan suatu tingkatan pencapaian didalam suatu sistem komunikasi data. Peningkatan pertumbuhan penggunaan internet ditambah dengan bertambahnya jumlah aplikasi-aplikasi berbasis *Web*, telah mengakibatkan adanya permintaan ketersediaan sistem komunikasi yang sulit diprediksi. Dalam rangka mencapai suatu tingkat layanan yang dapat diterima dan mengatasi masalah *bandwidth bottleneck*, maka para manajer jaringan memerlukan kemampuan untuk mengendalikan lalu lintas jaringan dan mengembangkan prioritas kebijakan yang sesuai dengan *bandwidth* yang tersedia (Jonathan Lukas, 2006). Untuk proses manajemen *bandwidth* dapat dilakukan dengan beberapa *tipe queue*, yaitu *simple queue* dan *queue tree*.



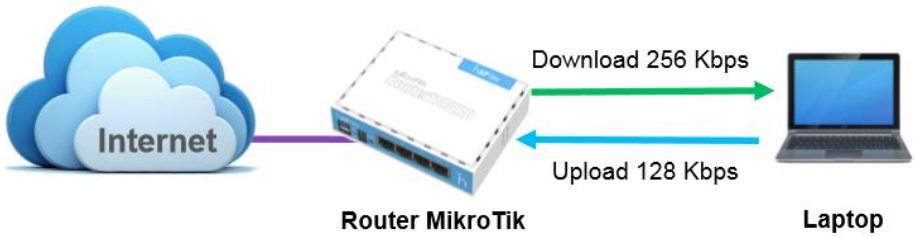
Gambar 2.15 Manajemen *bandwidth*

2.2.8.2.1 *Simple queue*

Merupakan metode *bandwidth management* termudah yang ada di Mikrotik. Menu dan konfigurasi yang dilakukan untuk menerapkan *simple queue* cukup sederhana dan mudah dipahami. Walaupun namanya *simple queue* sebenarnya parameter yang ada pada *simple queue* sangat banyak, bisa disesuaikan dengan kebutuhan yang ingin diterapkan pada jaringan.

Parameter dasar dari *simple queue* adalah Target dan *Max-limit*. Target dapat berupa IP *address*, *network address*, dan bisa juga *interface* yang akan diatur *bandwidth* nya. *Max-limit Upload / Download* digunakan untuk memberikan batas maksimal *bandwidth* untuk si target.

Simple Queue mampu *melimit Upload*, *download* secara terpisah atau Total(*Upload+download*) sekaligus dalam satu rule menggunakan tab Total. Setiap *rule* pada *Simple Queue* dapat berdiri sendiri ataupun dapat juga disusun dalam sebuah hierarki dengan mengarahkan *Parent* ke *rule* lain. Parameter-parameter lain juga bisa dimanfaatkan untuk membuat *rule* semakin spesifik seperti Dst, *Priority*, *Packete Mark* dan sebagainya. Salah satu contoh bisa di lihat di artikel Manajemen *Bandwidth* Menggunakan *Simple Queu*



Gambar 2.16 Cara Kerja *Simple Queue*

2.2.8.2.2 *Queue tree*

Merupakan fitur *bandwidth management* di Mikrotik yang sangat fleksibel dan cukup kompleks. Pendefinisian target yang akan dilimit pada *Queue tree* tidak dilakukan langsung saat penambahan *rule Queue* namun dilakukan dengan melakukan *marking* paket data menggunakan *Firewall Mangle*.

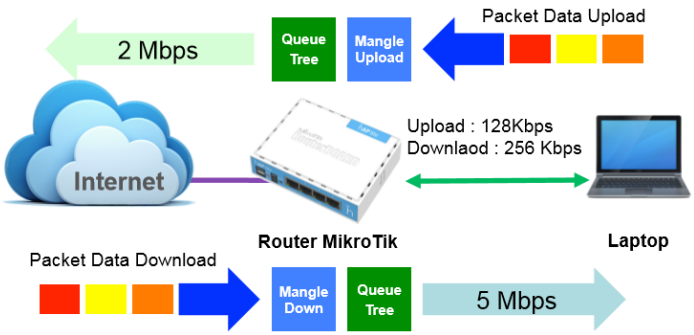
Inilah yang menjadikan penerapan *Queue tree* menjadi lebih kompleks. Langkah ini menjadi tantangan tersendiri, sebab jika salah pembuatan Mangle bisa berakibat *Queue tree* tidak berjalan.

Namun disisi lain penggunaan *Mangle* Packet-Mark ini juga menguntungkan, sebab akan lebih fleksible dalam menentukan *traffic* apa yang akan dilimit, bisa berdasar IP *Address, Protocol,* Port dan sebagainya. Setiap *service* pada jaringan dapat diberikan kecepatan yang berbeda. Sebagai contoh, bisa dilihat pada artikel penerapan *Queue tree* untuk memberikan limit kecepatan yang berbeda antara *traffic, games, online,* dan *browsing*. (www.mikrotik.co.id 2017).

Queue tree adalah konfigurasi *queue* yang bersifat *one way* (satu arah), ini berarti sebuah konfigurasi *queue* hanya akan mampu melakukan *queue* terhadap satu arah jenis *traffic*. Jika sebuah konfigurasi *queue* pada *Queue tree* ditujukan untuk melakukan *queue* terhadap *bandwidth download*, maka konfigurasi tersebut tidak akan melakukan *queue* untuk *bandwidth upload*, demikian pula sebaliknya. Sehingga untuk melakukan *queue* terhadap *traffic upload* dan *download* dari sebuah komputer *client*, kita harus membuat 2 (dua) konfigurasi *queue*. Menurut Towidjojo (2014), Pada saat akan menerapkan *queue* pada jaringan, dikenal dua rate atau alokasi *bandwidth* yang akan didapat oleh setiap user, yaitu :

1. *Committed Information Rate (*CIR), merupakan alokasi *bandwidth* terendah yang bisa didapatkan oleh sebuah *user* jika *traffic* jaringan sangat sibuk. Seburuk apapun keadaan dari jaringan tersebut, komputer *user* tidak akan mendapatkan alokasi *bandwidth* di bawah dari CIR.

2. *Maximum Information Rate* (MIR), merupakan alokasi *bandwidth* maksimum yang bisa didapatkan komputer user. MIR biasanya akan didapatkan seorang user jika ada alokasi *bandwidth* yang tidak digunakan lagi oleh *user* lain (Sukri, Jumiati 2017).



Gambar 2.17 Cara Kerja *Queue Tree*

Penjelasan beberapa argumen di *Queue tree* (Bagus Akhmad Gunawan 2015):

1 *Parent*: berguna untuk menentukan apakah *queue* yang dipilih bertugas sebagai *child queue* Ada beberapa pilihan default di *parent queue tree* yang biasanya digunakan untuk induk *queue*:

2 *Global-in*: Mewakili semua *input* interface pada umumnya. Maksudnya disini interface yang menerima *input* data/*trafik* sebelum difilter seperti *trafik upload*

3 *Global-out*: Mewakili semua *output* interface pada umumnya. Maksudnya disini interface yang mengeluarkan *output* data/*trafik* yang sudah difilter seperti *trafik download*

4 *Global-total*: Mewakili semua *input* dan *output* interface secara bersama, dengan kata lain merupakan penyatuan dari global-in dan global-out.

5 Interface name: ex: lan atau wan : Mewakili salah satu interface keluar. Maksudnya disini hanya *trafik* yang keluar dari interface ini yang akan di*queue*.

6 Packet Mark: Digunakan untuk menandai paket yang sudah ditandai di /ip firewall mangle.Priority (1 s/d 8) : Digunakan untuk memprioritaskan child *queue* dari child *queue* lainnya. Priority tidak bekerja pada induk *queue*. Child *Queue* yang mempunyai priority satu akan mencapai limit-at lebih dulu dari pada child *queue* yang berpriority

7 *Queue* Type: Digunakan untuk memilih type *queue* yang bisa dibuat secara khusus dibagian *queue* types

8 Limit At: *Bandwidth* minimal yang diperoleh oleh target/ip yang di*queue*

9 Max Limit: *Bandwidth* maksimal yang bisa dicapai oleh target/ip yang di*queue*.

10 Burst limit: *Bandwidth* maksimal yang bisa dicapai oleh target/ip yang di*queue* ketika burst sedang aktif

11 Burst time: Periode waktu dalam detik, dimana data *Rate* rata2 dikalkulasikan. Burst

12 Threshold: Digunakan ketika data *Rate* dibawah nilai burst threshold maka burst diperbolehkan.Ketika data *Rate* sama dengan nilai burst threshold burst dilarang. Untuk mengoptimalkan burst nilai burst threshold harus diatas nilai Limit At dan dibawah nilai Max Limit

2.2.9 *Traffic* Shapping

Traffic Shapping merupakan proses yang dapat digunakan untuk melakukan manajemen dan mengoptimalkan berbagai jenis jaringan dengan menerapkan layanan *Quality Of Service* (QOS) untuk menetapkan tipe-tipe lalu-lintas jaringan QOS adalah kemampuan untuk menggambarkan suatu tingkatan pencapaian dalam suatu sistem komunikasi data (Imam Riadi. 2010)

Traffic shapping menyediakan suatu mekanisme sejumlah *traffic* yang akan dikirim ke jaringan. Oleh karena itu, *traffic shapping* perlu untuk di implementasikan untuk mengontrol *trffic* yang masuk ke dalam jaringan. Tujuan *traffic shapping* adalah memonitor *traffic* secara aktif, menangani kondisi kongesti dan memberikan prioritas diantar aliran *traffic* sesuai dengan yang diatur oleh administrator jaringan.

2.2.10 Quality Of Service (QoS)

Quality of Service (QoS) merupakan mekanisme jaringan yang memungkingkan aplikasi-aplikasi atau layanan dapat beroperasi sesuai dengan yang diterapkan. Tujuan dari *QoS* adalah untuk memenuhi kebutuhan-kebutuhan layanan yang berbeda, yang menggunakan infrastruktur yang sama. Performasi mengacu ke tingkat kecepatan dan keandalan penyampaian berbagai jenis beban data didalam suatu komunikasi (Arifin., 2012) Berikut ini merupakan beberapa parameter QoS yang akan digunakan dalam mengukur performasi jaringan, yaitu :

2.2.10.1 Throughput

Yaitu kecepatan (rate) *transfer* data yang efektif yang diukur dalam *bps*. *Throughtput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut.

Tabel 2.1 *Throughput*

Katregori *Througput*	*Througput*	Indeks
Sangat Bagus	100%	4
Bagus	75%	3
Sedang	50%	2
Jelek	<25%	1

Adapun persamaan yang digunakan untuk mengukur *throughput* adalah sebagai berikut: $Throughput = \frac{paket\ data\ diterima}{lama\ pengamatan}$

$$Throughput = \frac{Throughput}{Alokasi\ Banwidth\ User}\ x\ 100\%$$

2.2.10.2 Packet Loss

Packet Loss merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi Karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi Karena *retransmisi* akan mengurangi efisiensi jaringan secara

keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut. Jika terjadi kongesti yang cukup lama, bufferakan penuh, dan data baru tidak akan diterima. Nilai packet loss sesuai dengan versi TIPHON sebagai berikut:

Tabel 2.2 *Packet Loss*

Kategori Degrasi	*Packet los*	Indeks
Sangat Bagus	0%	4
Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

Adapun persamaan yang digunakan mengukur *packet loss* adalah

$$Packet\ Loss = \frac{Paket\ data\ dkirim - Paket\ data\ diterima}{Paket\ data\ dikirim}\ x\ 100\%$$

2.2.10.3 Jitter

Hal ini diakibatkan oleh variasi-variasi dalam panjang antrian,dalam waktu pengolahan data,dan juga dalam waktu penghimpunan ulang paket-paket diakhir perjalanan *jitter*. *Jitter* lazimnya disebut variasi *delay*, berhubungan erat dengan *latency*, yang menunjukkan banyaknya variasi *delay* pada transmisi data di jaringan. *Delay* antrian pada *router* dan *switch* dapat menyebabkan *jitter*. Terdapat empat kategori penurunan *performansi* jaringan berdasarkan nilai *peak jitter* sesuai dengan versi TIPHON yaitu:

Tabel 2.3 *Jitter*

Kategori Degrasi	*Peak Jitter*	Indeks
Sangat Bagus	0 ms	4
Bagus	0 s/d 75 ms	3
Sedang	75 s/d 125 ms	2
Jelek	125 s/d 225 ms	1

Adapun persamaan yang digunakan untuk mengukur *jitter* adalah

$$Jitter = \frac{Total\ variasi\ delay}{Total\ paket\ diterima}\ x\ 100\%$$

Total variasi delay = Delay – (Rata-Rata delay)

2.2.10.4 Delay

Adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ketujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama. Adapun komponen *delay* adalah sebagai berikut: Menurut versi TIPHON, besarnya *delay* dapat diklasifikasikan sebagai berikut.

Tabel 2.4 ***Delay***

Kategori *Latensi*	Besar *Delay*	Indeks
Sangat Bagus	<150 ms	4
Bagus	150 s/d 300 ms	3
Sedang	300 s/d 450 ms	2
Jelek	>450 ms	1

Untuk mengukur *delay* digunakan persamaan sebagai berikut.

$$Delay = \frac{Paket\ leght}{Link\ bandwidth}$$

2.2.11 *Proxy*

Proxy merupakan perangkat yang bersifat "middleman", yang bekerja diantara *client* & Server, bertugas menghadle transmisi request ataupun respon. Data yang melewati *proxy* , bisa diubah oleh *proxy* atau tidak diubah sama sekali, tergantung implementasi fitur dan kemampuan *proxy* . Adapun manfaat *proxy* sebagai berikut (www.mikrotik.co.id):

1. *Security* : *Proxy* mampu melakukan pengecekan terhadap incoming response & *outgoing request*, sehingga memungkinkan untuk melakukan *block* paket yang tidak diharapkan
2. *Caching* : Menyimpan sementara data dari internet di *storage*, sehingga jika ada *client* lain yang hendak mengakses data yang sama, cukup mengambil data di *cache*. Dengan begitu, kita bisa menghemat *bandwidth*.
3. *Perfomance* : Pada kondisi tertentu, *proxy* dapat menurunkan *latency* beberapa data telah di *cache* tidak perlu diakses langsung dari internet, cukup dari *storage proxy* yang masih dalam satu jaringan sehingga *latency* bisa lebih bagus.

2.2.12 Perangkat Lunak

2.2.12.1 *Winbox*

Winbox adalah sebuah *software* atau *utility* yang di gunakan untuk meremote sebuah server mikrotik kedalam mode GUI (Graphical User Interface) melalui *operating system* windows. Kebanyakan teknisi banyak mengkonfigurasi mikrotik os atau mikrotik routerboard menggunakan winbox di banding dengan yang mengkonfigurasi langsung lewat mode CLI (Command Line Interface). Hal ini karena menggunakan winbox dirasa lebih mudah dan simple dibanding melalui browser. Dan hasilnya pun juga lebih cepat. (Burhanudin. 2014)

Fungsi dari winbox ini banyak sekali. Winbox mudah di install mudah dipakai, ringan cepat dan tepat. Jika ingin diperinci bisa dilihat dibawah ini.

1. Setting mikrotik router dalam mode GUI
2. Setting bandwith jaringan internet
3. Membelokir sebuah website/situs
4. Mempercepat pekerjaan
5. Dan masih banyak yang lainnya.

2.2.12.2 Wireshark

Wireshark adalah penganalisa paket jaringan. Sebuah analisa paket jaringan akan mencoba untuk menangkap paket jaringan dan mencoba untuk menampilkan data paket serinci mungkin. Anda bisa memikirkan analisa paket jaringan sebagai alat pengukur yang digunakan untuk memeriksa apa yang terjadi di dalam kabel jaringan, seperti voltmeter digunakan oleh seorang teknisi listrik untuk memeriksa apa yang terjadi dalam sebuah kabel listrik (tetapi pada tingkat

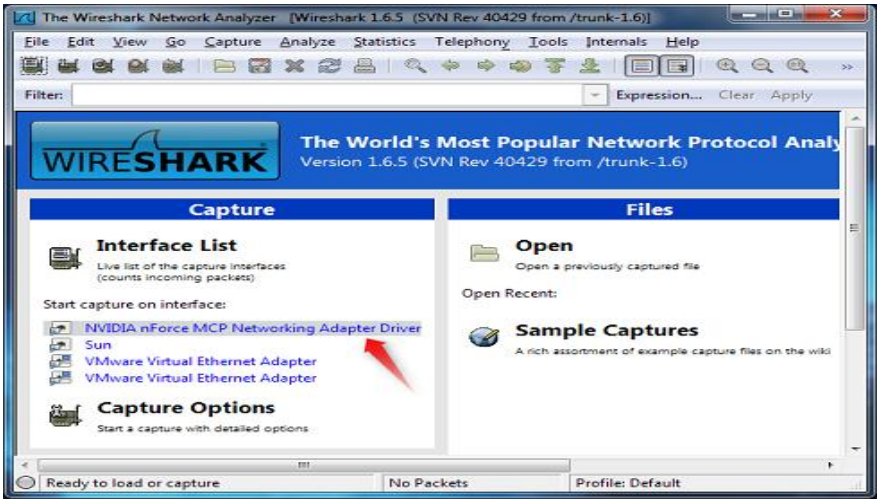
yang lebih tinggi, tentu saja). Di masa lalu, alat-alat seperti yang baik sangat mahal, eksklusif atau keduanya. Namun dengan munculnya wireshark semua itu telah berubah. Wireshark adalah salah satu yang terbaik *open source* analisa paket yang tersedia saat ini. Berikut adalah beberapa contoh orang menggunakan Wireshark untuk:

1. Administrator jaringan menggunakannya untuk memecahkan masalah jaringan
2. insinyur keamanan jaringan menggunakannya untuk memeriksa masalah keamanan
3. Pengembang menggunakannya untuk implementasi protokol men-debug
4. Orang-orang menggunakannya untuk belajar internal protokol jaringan

Disamping contoh-contoh ini wireshark dapat membantu dalam banyak situasi lain juga. Berikut ini adalah beberapa dari banyak fitur Wireshark tersedia, Lihat pada gambar 2.18.

1. Tersedia untuk UNIX dan Windows.
2. Tangkap hidup paket data dari antarmuka jaringan.
3. Buka file yang berisi data paket yang diambil dengan tcpdump / WinDump, Wireshark dan sejumlah program capture paket lainnya.
4. Impor paket dari file teks yang berisi tempat pembuangan hex dari paket data.
5. Tampilan paket dengan informasi protokol yang sangat rinci.
6. Simpan data paket yang diambil.
7. Ekspor beberapa atau semua paket di sejumlah format capture file.
8. Filter paket pada banyak kriteria.

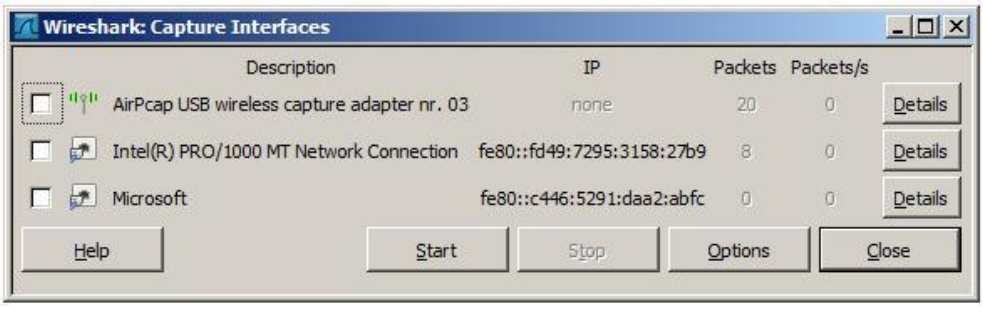
9. Cari untuk paket pada banyak kriteria.
10. Layar Colorize paket berdasarkan filter.
11. Buat berbagai statistik.



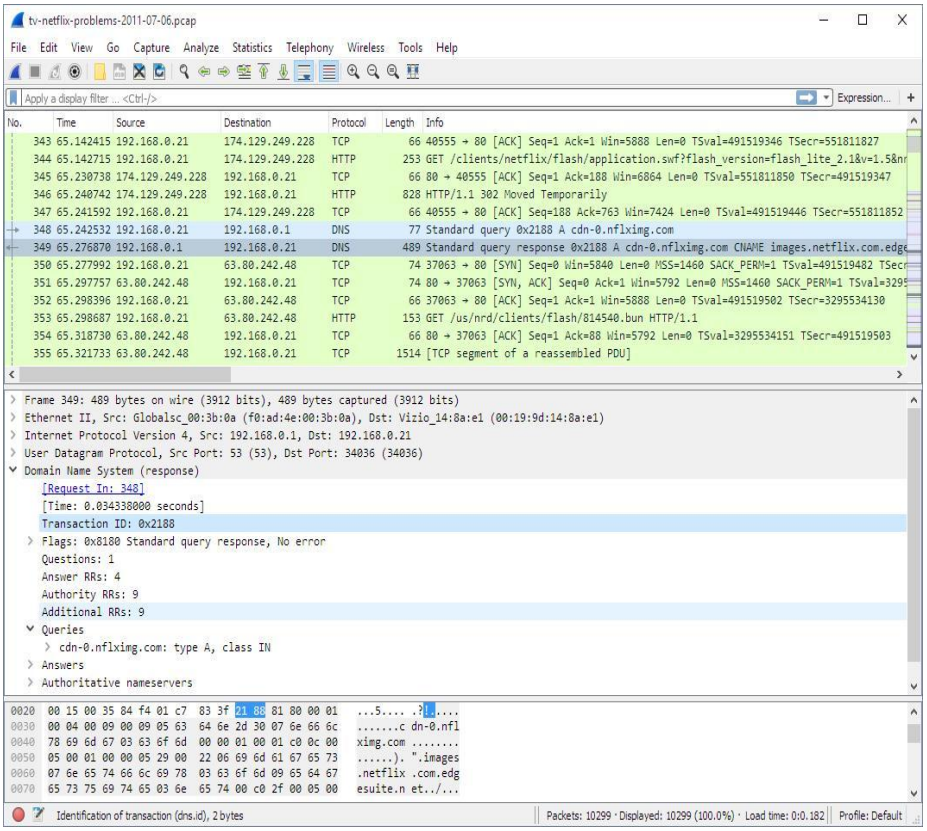
Gambar 2.18 Tampilan Awal Aplikasi Wireshark

Metode berikut dapat digunakan untuk memulai *capture* paket-paket dengan wireshark:

1. Anda dapat mengklik dua kali pada sebuah *interfaces* di *main windo.*
2. Anda bisa mendapatkan gambaran tentang *interfaces* yang tersedia menggunakan "*Capture Interfaces*" kotak dialog (Capture → Options). Lihat gambar 2.19 (a) dan gambar 2.20 (b).



Gambar 2.19 *Capture Interface*



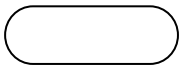
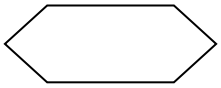
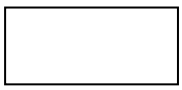
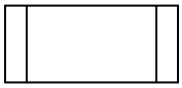
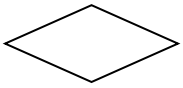
Gambar 2.20 *Capture* Wireshark (b)

2.2.13 Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong *analyst* dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut. *Flowchart* adalah bentuk gambar/diagram yang mempunyai aliran satu atau dua arah secara *sekuensial*. *Flowchart* digunakan untuk merepresentasikan maupun mendesain program. Oleh karena itu *flowchart* harus bisa merepresentasikan komponen-

komponen dalam bahasa pemrograman. (Algoritma dan Pemrograman Menggunakan C++, 2005)

Table 2.5 Simbol-Simbol *Flow chart*

SIMBOL	NAMA	FUNGSI
	**TERMINATOR**	Permulaan/akhir program
	**GARIS ALIR** **(FLOW LINE)**	Arah aliran program
	**PREPARATION**	Proses inisialisasi/pemberian harga awal
	**PROSES**	Proses perhitungan/proses pengolahan data
	**INPUT/OUTPUT DATA**	Proses input/output data, parameter, informasi
	**PREDEFINED PROCESS** **(SUB PROGRAM)**	Permulaan sub program/proses menjalankan sub program
	**DECISION**	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	**ON PAGE**	Penghubung bagian-bagian flowchart

SIMBOL	NAMA	FUNGSI
	CONNECTOR	yang berada pada satu halaman
	**OFF PAGE CONNECTOR**	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

2.2.14 *Address List*

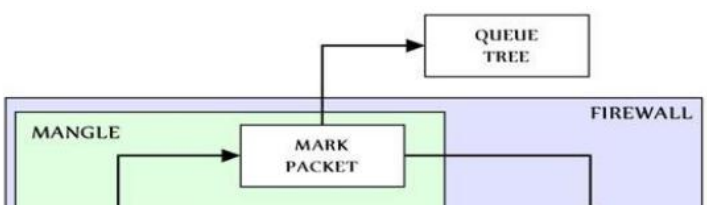
Address list, adalah salah satu fitur mikroTik yang fungsinya untuk memudahkan kita dalam menandai suatu konfigurasi *address*. Sehingga dengan *address list*, kita bisa membuat *list address* yang ingin di tandai tanpa harus mengganggu konfigurasi penting di fitur lainnya.

Fungsi lain *address list* adalah sebagai *action* pada *firewall* agar *admin* bisa menentukan *address* apa saja yang ingin ditandai dan dimasukkan kedalam *address list.* Jika pada lab sebelumnya, kita menggunakan fitur log untuk membuat catatan aktivitas si Router. Bisa dibilang sama, *address list* juga memiliki fungsi membuat catatan seperti penanda *address* paket agar dimasukkan kedalam *address list*. (wiski.mikrotik.com)

2.2.15 Mangle

Mangle sendiri memiliki fungsi untuk menandai sebuah koneksi atau paket data, yang melewati *route*, masuk ke *router*, ataupun yang keluar dari *router*. Pada implementasinya Mangle sering dikombinasikan dengan fitur lain seperti

Management Bandwidth, Routing policy, dll. Adapun fungsi dari masing-masing chain yang ada pada mangle adalah sebagai berikut:



Gambar 2.21 *Firewall Mangel*

1. *Forward, Input, Output* :

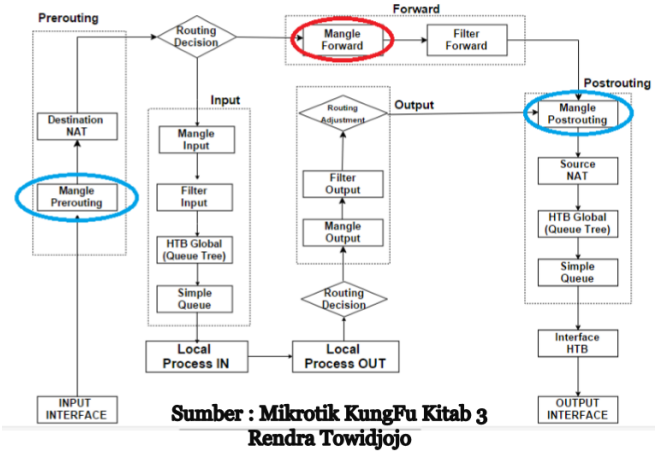
Untuk penjelasan mengenai *Forward, Input,* dan *Output* sebenarnya tidak jauh berbeda dengan apa yang telah diuraikan pada Filter rules diatas. Namun pada Mangle, semua jenis *trafik* paket data *forward, input,* dan *output* bisa ditandai berdasarkan koneksi atau paket atau paket data.

2. *Prerouting* :

Merupakan sebuah koneksi yang akan masuk kedalam router dan melewati *router*. Berbeda dengan *input* yang mana hanya akan menangkap *trafik* yang masuk ke router. *Trafik* yang melewat router dan *trafik* yang masuk kedalam router dapat ditangkap di chain prerouting.

3. Postrouting :

Kebalikan dari prerouting, postrouting merupakan koneksi yang akan keluar dari *router*, baik untuk *trafik* yang melewati *router* ataupun yang keluar dari *router*. (mikrotik.co.id) (Rendra Towowidjojo : 2013)



Gambar 2.22 Pemilihan Chain