APLIKASI MONITORING STATUS SERVICE MENGGUNAKAN REQUEST REPLY

S K R I P S I

Diajukan untuk memenuhi salah satu syarat
Memperoleh Gelar Sarjana Teknik
Jurusan Teknik Informatika

Oleh :

Nama : Nyoman Gede Wisnu Pramana

NIM : 005314014

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2005

STATUS SERVICE MONITORING APPLICATION USING REQUEST REPLY

A Thesis

Presented as Partial Fulfillment of the Requirements
To Obtain the Sarjana Teknik Degree
In Departement of Informatics Tecnology

Oleh :

Nama : Nyoman Gede Wisnu Pramana

NIM : 005314014

DEPARTEMENT OF INFORMATICS TECNOLOGY

FACULTY OF ENGINEERING

SANATA DHARMA UNIVERSITY

YOGYAKARTA

2005

Halaman Persetujuan

S K R I P S I

APLIKASI MONITORING STATUS SERVICE MENGGUNAKAN REQUEST REPLY

Oleh :

Nama : Nyoman Gede Wisnu Pramana

NIM : 005314014

Telah disetujui oleh :

Pembimbing I ,

(Drs. Harris Sriwindono, M.Kom.) tanggal : September 2005

Pembimbing II

(H. Agung Hernawan, S.T) tanggal : September 2005

Halaman Pengesahan

S K R I P S I

APLIKASI MONITORING STATUS SERVICE MENGGUNAKAN REQUEST REPLY

Disusun oleh :

Nama : Nyoman Gede Wisnu Pramana

NIM : 005314014

Telah dipertahankan di depan panitia penguji

Pada tanggal 30 Juli 2005

Dan dinyatakan memenuhi syarat

Susunan Panitia Penguji

	Nama lengkap	Tanda tangan
Ketua	: Drs. Harris Sriwindono, M.Kom.	.....
Sekretaris	: H. Agung Hernawan, S.T.	.....
Anggota	: Ridowati Gunawan, S.Kom., M.T.	.....
Anggota	: Albertus Agung Hadiatma, S.T.	.....

Yogyakarta, September 2005

Fakultas Teknik
Universitas Sanata Dharma
Dekan

(Ir. Greg. Herliarko, SJ., SS., B.ST., MA., M.Sc.)

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian karya orang lain kecuali yang telah disebutkan dalam kutipan dan daftar pustaka, sebagaimana layaknya karya ilmiah.

Demikian pernyataan ini saya buat dengan sebenar – benarnya.

Yogyakarta, 20 September 2005

Penulis

Nyoman Gede Wisnu Pramana

Halaman Persembahan

Ida Sang Hyang Widi WaÇa

My family
(Papa,Mama,Asri,Bayu,Anya)
it's nice to be part of astika family

Sari
for being the one
who always stand beside me
you light up my day
can't do it without you

All my friends in bali or jogja
I have no words for all of you guys and girls
but I had a really... really good times
thank's for every single thing
and every single time

myself
I made it !!!
I'm free at last...

Motto

i live in my own world
so don't bother to join in
just mind your own live...

(sleepers_design, 2005)

ABSTRAKSI

Meningkatnya kebutuhan akan teknologi informasi yang cepat dan akurat, telah mendorong berkembangnya dunia internet dewasa ini. Teknologi internet yang semakin canggih, telah memberi banyak kemudahan-kemudahan di berbagai bidang. Untuk komunikasi data dalam jaringan internet biasanya digunakan beragam protokol service. Terkadang suatu komputer mampu menyediakan lebih dari satu jenis service, dan diperlukan cara yang unik untuk mengidentifikasi service-service tersebut.

Dalam komunikasi data antar *node* dapat terjadi adanya kemacetan, kemacetan ini dapat terjadi pada level aplikasi maupun level network. Untuk itulah diperlukan suatu program aplikasi yang mampu memantau keadaan node dalam jaringan secara terus menerus tanpa harus membebani jalur jaringan itu sendiri.

ABSTRACT

Increasing of fast and accurate information technology have develope the internet these days. The internet technology has become more complicated, its gives many help. To communicate in internet network usually use many kind of service protocol. Sometime a computer provide more than one service, and it need an unique way to identified those services.

In data communication between node is possible to have a data traffic, this traffics can happen in application level or network level. Therefore an application is need to monitor the service in a node continuely.

Kata Pengantar

Puji syukur kepada Tuhan Yang Maha Esa yang telah memberikan hikmat, rahmat serta karunia-Nya, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik. Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik pada jurusan Teknik Informatika Fakultas Teknik Universitas Sanata Dharma.

Tugas akhir dengan judul "APLIKASI MONITORING STATUS SERVICE MENGGUNAKAN REQUEST REPLY" ini diharapkan dapat bermanfaat bagi semua user yang berhubungan dengan bidang jaringan komputer.

Dalam proses pengerjaan tugas akhir ini, penulis mendapatkan bantuan dan bimbingan dari berbagai pihak. Maka dari itu pada kesempata ini penulis hendak mengucapkan terima kasih dengan segala hormat kepada :

1. Bapak Ir. Greg. Herliarko, SJ., SS., B.ST., MA., M.Sc selaku Dekan Fakultas Teknik Universitas Sanata Dharma
2. Ibu A.M Pollina, S.Kom., M.Sc. selaku Ketua Jurusan Teknik Informatika Universitas Sanata Dharma
3. Drs. Harris Sriwindono, M.Kom, selaku dosen pembimbing I. Terima kasih atas masukannya untuk program ini.
4. H. Agung Hernawan, S.T, selaku dosen pembimbing II yang telah menemani penulis dari awal sampai tugas akhir ini terselesaikan. Terima kasih atas fasilitas laboratorium jaringannya dan solusi – solusinya.

5. Buat keluargaku, papa, mama, mbak asri, ade, anya. Terima kasih buat semua dorongannya dan fasilitasnya.
6. Terima kasih banyak buat Sari. Terima kasih telah berada disuatu tempat 600 kilometer dari Jogja dan tetap sabar menunggu.
7. Anak – anak TI'00, mas Widi yang membantu mencarikan software untuk program ini, mas Andi yang telah mengenalkan band - band indie jogja dan menemani menonton konser-konsernya, Mas Adi atas penerangannya (hehe...), mas Lana yang selalu memberikan senyum cerianya, mas Warih mas Natius, mas Aryo dan mas Slamet. Terima kasih untuk semuanya yang telah membagi ilmunya dan terima kasih telah mengisi hari – hari sepiku di jogja.
8. Anak – anak Mesin'00, mas Agus, mas Dixon, mas Ham, dan mas Herry buat kata – kata mutiaranya.
9. Mas Hatma, yang telah dengan sabar mengajarkan penulis bahasa pemrograman Java.
10. Buat anak – anak Mundu, karma, yudi, abdi, lolet dan blangkorn, terimakasih telah memberikan tumpangan.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari sempurna dan banyak kekurangannya, untuk itu penulis berterima kasih atas saran dan kritikannya. Bagi yang ingin mengirimkan saran tentang tulisan ini silahkan mengirimkan email kestreet2sleepers@yahoo.com . Sekian, terima kasih.

Yogyakarta, 20 Agustus 2005

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Semenjak teknologi jaringan komputer muncul dan berkembang seperti sekarang ini, berbagai pihak mencoba terus mendayagunakan teknologi yang sangat baik ini. Banyak orang yang melihat bahwa teknologi ini pendukung penting dalam otomatisasi dan efisiensi proses bisnis di suatu organisasi dan salah satunya adalah sebagai komponen pendukung penting munculnya teknologi internet.

Meningkatnya kebutuhan akan teknologi informasi yang cepat dan akurat, telah mendorong berkembangnya dunia internet dewasa ini. Teknologi internet yang semakin canggih, telah memberi banyak kemudahan-kemudahan di berbagai bidang. Dengan adanya fasilitas internet, seseorang dapat berkomunikasi secara luas dengan orang lain di berbagai penjuru dunia dengan biaya yang murah, cepat dan akurat.

Internet terdiri atas berjuta-juta komputer, bertempat diseluruh dunia, berkomunikasi dan mengirimkan informasi melalui berbagai sistem, platform, dan peralatan networking. Masing-masing komputer ini akan mempunyai suatu alamat IP.

Alamat IP adalah 32-bit angka-angka, berisi empat bilangan oktaf (8 bit biner) yang dipisahkan oleh suatu titik(William Stallings,2000). Masing-masing komputer dengan suatu koneksi internet langsung akan mempunyai suatu IP.

Beberapa komputer mempunyai alamat sementara, ini terjadi bila *host* tersebut terhubung ke ISP melalui suatu modem. Terdapat juga host yang mempunyai alamat permanen, dan beberapa bahkan mempunyai nama domain sendiri.

Suatu IP mampu mengidentifikasi secara unik suatu alat atau sistem yang terhubung ke internet. Bila *host* tersebut ingin berhubungan dan mengirimkan suatu pesan maka hal itu dapat dilakukan bila *host* tujuannya memiliki alamat IP spesifik. Tanpa suatu alamat IP, pesan yang dikirim tidak akan mencapai tujuannya.

Saat ini banyak sekali muncul website-website baru yang memberikan berbagai macam fasilitas yang berfungsi untuk menarik perhatian dari penggunanya, seperti misalnya *email, mailing list, free sms* dan lain-lain. Semua alamat website ini memiliki alamat IP tersendiri.

Dan untuk membuka halaman-halaman website ini disajikan oleh service suatu web server. Terkadang suatu komputer mampu menyediakan lebih dari satu jenis service, dan diperlukan cara yang unik untuk mengidentifikasi service-service tersebut. Seperti suatu alamat IP, kita menggunakan suatu nomor yang disebut nomor *port*. Nomor *port* merupakan layanan service yang menjadi jembatan komunikasi antara level transport dan level aplikasi. Tiap nomor port mewakilkan proses pada tiap service pada level aplikasi. Nomor port ini diletakkan pada header TCP.

Service umum (seperti HTTP, FTP, Telnet, SMTP) mempunyai angka-angka port terkenal. Sebagai contoh, kebanyakan web server menggunakan *port*

80, tetapi hal ini bukanlah sesuatu yang baku, nomor port ditentukan oleh administrator jaringan tersebut.

Ada beberapa mekanisme komunikasi yang dapat menggunakan untuk menyediakan jasa jaringan. Kita bisa menggunakan UDP (*unreliable datagram protocol*), atau TCP (*transfer-control protocol*). Kebanyakan protokol service (seperti HTTP, FTP, SMTP) menggunakan TCP. Dalam komunikasi antar *node* dapat terjadi adanya kemacetan, kemacetan ini dapat terjadi pada level aplikasi maupun level network.

Dengan melihat keadaan seperti diatas, maka penulis tertarik membuat suatu penelitian berjudul APLIKASI MONITORING STATUS SERVICE MENGGUNAKAN REQUEST REPLY. Program ini digunakan untuk memonitor service pada *node* yang terhubung ke jaringan. *Node* disini dapat berupa *host, router* atau komponen lainnya yang digunakan dalam membangun suatu jaringan. Program ini memanfaatkan proses *request- reply* saat koneksi dilakukan. Dalam proses ini akan diperoleh input atau masukan yang dapat digunakan untuk mengetahui kondisi service pada *node* yang diinginkan. Program ini akan dibuat menggunakan bahasa pemrograman Java. Pemilihan bahasa pemrograman Java mengingat Java telah dilengkapi kelas –kelas yang diperlukan dalam pembuatan program ini nantinya.

1.2 Batasan Masalah

Batasan masalah pada program *network view* ini adalah sebagai berikut:

1. Monitoring yang dilakukan program ini hanya berdasarkan proses *request reply.*

2. Bagian instalasi Sistem Operasi dan service yang digunakan tidak dibahas dalam tulisan ini.
3. Service yang dapat dipantau melalui program ini adalah ICMP, HTTP, FTP dan DNS
4. Bahasa pemrograman yang dipakai untuk pembutan program ini adalah Java J2SDK.
5. Program ini menggunakan user interface grafis.
6. Jumlah host yang dapat dipantau pada satu peta dibatasi.

1.3 Rumusan Masalah

1. Bagaimana mengetahui status/kondisi servis pada TCP/IP yang diperlukan dalam jaringan, dalam keadan aktif atau tidak berdasarkan *reply* yang diterima?

1.4 Tujuan dan Manfaat Penulisan

Tujuan dari pembuatan program ini adalah :

1. Mengimplementasikan proses request reply untuk memonitoring keadaan servis pada jaringan komputer.
2. Mengimplementasikan pemrograman java dalam pembuatan program monitoring jaringan

Dengan adanya program ini diharapkan dapat membantu network administrator dalam menjalankan tugasnya dalam mengawasi jaringan sehingga dapat menentukan tindakan yang harus dilakukan jika terdapat kerusakan.

1.5 Metodologi Penelitian

1. Survey kebutuhan program

   Mempelajari cara kerja service yang akan digunakan pada program ini dalam melakukan proses *request reply*.

2. Studi Literatur

   Mencari bahan-bahan yang mendukung program yang akan dibuat, termasuk mencari informasi (literatur seperti internet dan buku) dari luar pemrograman mengenai inti program yang akan dibuat

3. Perancangan antarmuka program

   Memulai merancang bagaimana desain antarmuka untuk program yang akan dibuat, agar mudah digunakan dan mudah dipahami.

4. Perancangan program

   Merancang program sesuai dengan desain antarmuka yang telah dirancang sebelumnya.

5. Pembuatan program

   Membuat program yang telah dirancang dengan menggunakan bahasa pemrograman

6. Pengujian program

   Menguji program yang telah dibuat pada jaringan komputer.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Berisi latar belakang, batasan masalah, tujuan dan manfaat, metodologi penelitian, serta sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Bab ini menjelaskan tentang TCP/IP, protokol service yang digunakan dalam program ini seperti DNS, FTP, ICMP dan HTTP.

BAB III PERANCANGAN SISTEM

Bab ini menjelaskan tentang analisis sistem dan identifikasi masalah, perancangan sistem. Perancangan meliputi perancangan *interface*, perancangan perangkat lunak dan alur proses.

BAB IV IMPLEMENTASI SISTEM

Merupakan bab tentang pengimplementasian perancangan sistem ke dalam bahasa pemrograman dan pengujian dalam jaringan

BAB V ANALISA HASIL

Berisi tentang analisa terhadap implementasi program yang telah dibuat

BAB IV KESIMPULAN

Berisi tentang kesimpulan dari keseluruhan sistem yang dibangun dan saran-saran yang diajukan.

DAFTAR PUSTAKA

LAMPIRAN

BAB II

LANDASAN TEORI

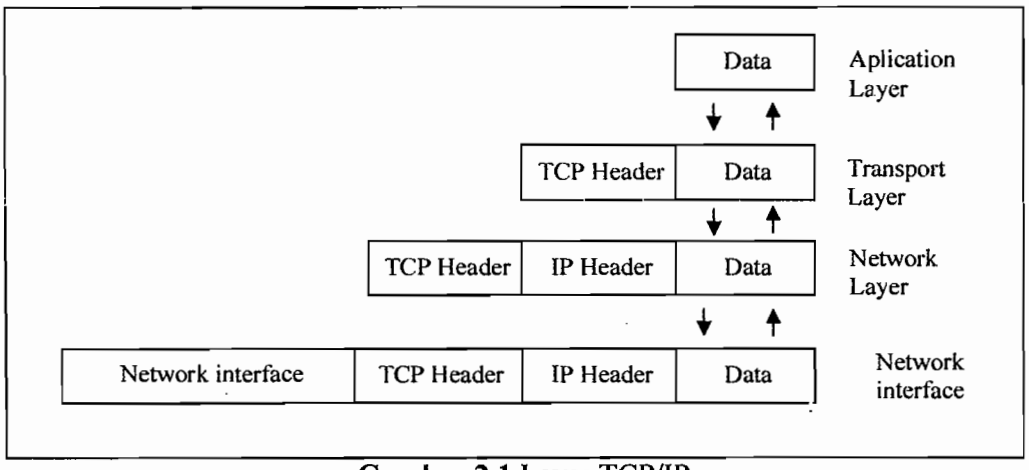
Program ini dibuat dengan memanfaatkan beberapa protokol service yang ada pada suatu jaringan internet. Protokol - protokol tersebut adalah sebagai berikut :

2.1 TCP / IP

TCP/IP (*Transmision Control Protocol/Internet Protocol*) menjadi standar *defacto* jaringan komputer berkaitan dengan ciri-ciri yang terdapat pada protokol itu sendiri meliputi:

- Protokol TCP/IP menggunakan standar protokol yang terbuka.
- Standar protokol TCP/IP dalam bentuk *Request For Comment* (RFC) dapat diambil oleh siapapun tanpa biaya.
- Pengembangan tidak bergantung pada sistem operasi atau perangkat keras tertentu.
- *Independen* terhadap perangkat keras jaringan dan dapat dijalankan pada jaringan Ethernet, Token Ring, jalur telepon *dial-up*, dan praktis pada jenis media transmisi apapun.
- Pengalamatan TCP/IP bersifat unik dalam skala global.
- TCP/IP memiliki fasilitas routing yang memungkinkan dapat diterapkan pada *internetwork*.

Gambar 2.1 memperlihatkan sekumpulan protokol TCP/IP yang dimodelkan dalam empat layer TCP/IP.



Gambar 2.1 Layer TCP/IP

Keempat layer tersebut adalah:

1. *Networks Interface Layer*, bertanggung jawab mengirim dan menerima data ke dan dari media transmisi.
2. *Internet Layer*, protokol yang berada pada layer ini bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat.
3. *Transport layer*, protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua host / komputer.
4. *Application Layer*, pada layer ini terletak semua aplikasi yang menggunakan protokol TCP/IP.

Seperti halnya protokol komunikasi lainnya, TCP/IP juga dibagi menjadi beberapa lapisan :

a. Internet (IP)

IP bertanggung jawab untuk memindahkan paket data dari satu titik ke titik lainnya. IP meneruskan paket data berdasarkan empat *byte* alamat tujuan (*IP number / IP address*). Otoritas *Internet* membagi alamat IP

menjadi beberapa organisasi yang berbeda. Setiap organisasi akan mengelompokkan alamat itu menjadi beberapa departemen. IP bekerja sebagai pintu gerbang dan alamat mesin dalam memindahkan paket data dari satu departemen ke satu organisasi dan diteruskan ke suatu wilayah regional dan kemudian seluruh dunia ini.

b. Transport (TCP dan UDP)

TCP bertanggung jawab untuk pengecekan pengiriman paket data dari *client* ke *server*. Dalam pengirimannya, data dapat hilang ditengah jalan dalam jaringan. Oleh karena itu, TCP menambahkan suatu paket tambahan untuk mengecek kesalahan data atau kehilangan data dan memerintahkan untuk mengirim ulang data sampai data diterima dalam keadaan benar dan lengkap.

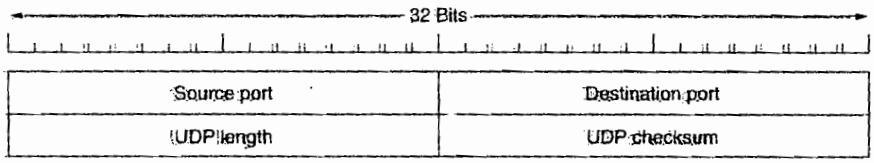
2.2 UDP (User Datagram Protocol)

Koneksi UDP bersifat *connectionless*, tidak didahului proses *handshaking* antara pengirim UDP untuk menetapkan jalur koneksi. Setiap paket UDP ditangani secara independen dan tidak terhubung antara satu dengan yang lainnya. Jadi kemungkinan untuk kehilangan paket data atau urutan datangnya data tidak teratur sangat mungkin terjadi.

Paket UDP disebut segment atau *user datagram,* terdiri dari dua bagian yaitu header dan data. Pada header terdapat beberapa bagian yaitu :

1. *Source port number* : nomor *port* yang digunakan oleh proses pada *host* sumber
2. *Destinaton port number* : nomor *port* yang digunakan oleh proses pada *host* tujuan

3. *Length* : panjangnya paket (header ditambah data)
4. *Checksum* : untuk mendeteksi *error* dari paket.



Gambar 2.2 Header UDP

UDP hanya menggunakan tiga elemen dasar dari service transport yaitu:

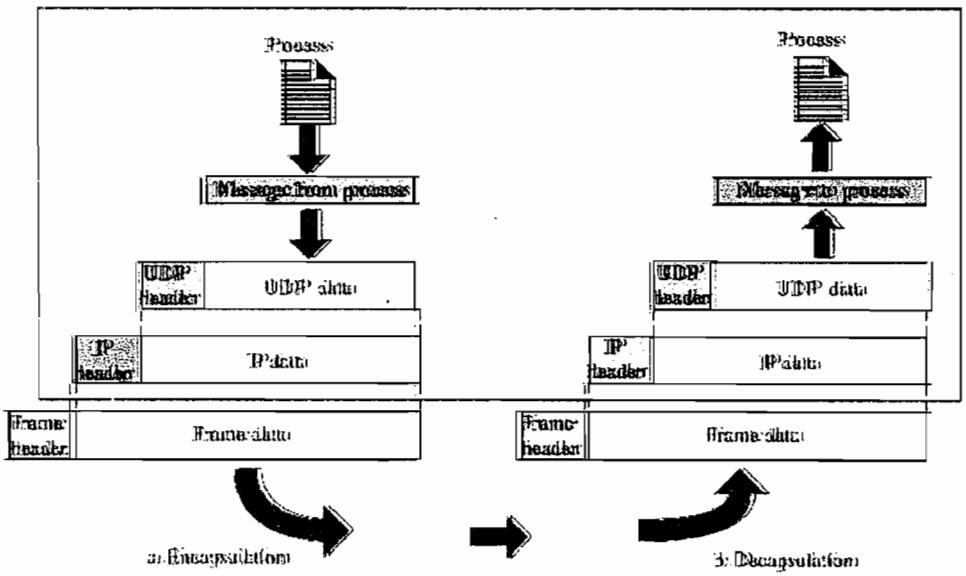
1. *Encapsulation/ decapsulation*

   Encapsulation pada pengirim :

   - Proses melewatkan pesan UDP bersama dengan alamat *socket* dan panjang data.
   - UDP menambahkan *header* UDP pada data
   - UDP melewatkan segment ke IP dengan alamat socket
   - IP menambahkan *header*-nya, menggunakan nilai 17 pada *field* protokol untuk menandakan bahwa data berasal dari UDP

   Decapsulation pada penerima :

   - UDP menerima data dari IP
   - UDP menggunakan *checksum* untuk mendeteksi *error*
   - Jika tidak terdapat *error* maka UDP akan meneruskan pesan bersama dengan pengirim dan alamat *socket* pada proses

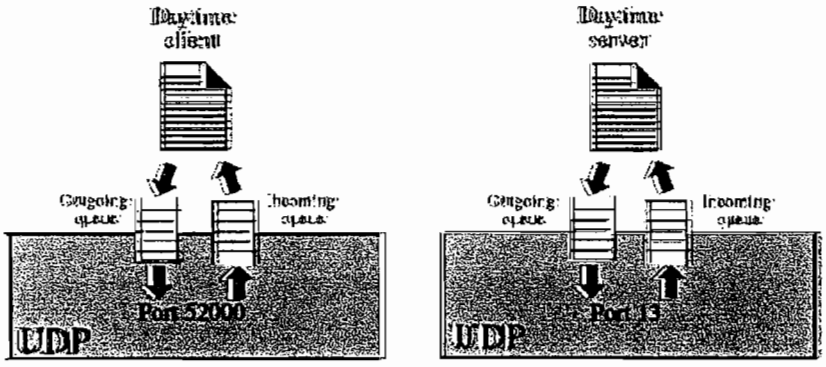


Gambar 2.3 Encapsulation/Decapsulation

2. *Queuing*

Queuing dibuat oleh sistem operasi saat proses meminta sebuah nomor *port* dan menghancurkannya saat proses dihentikan. Ada dua tipe *queuing* yaitu :

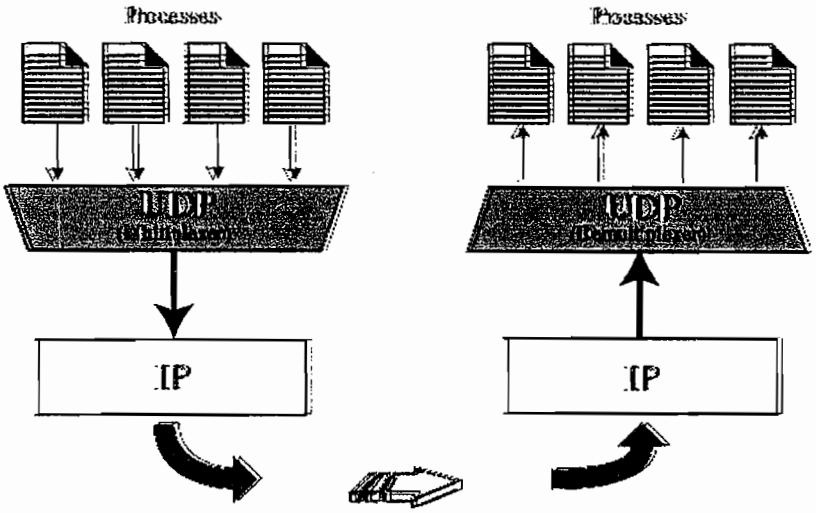
- *Outgoing queue* : penyimpanan data sementara selama transmisi data
- *Incoming queue* : tempat penyimapan data sementara untuk aplikasi



Gambar 2.4 Proses Queuing

3. *Multiplexing/demultiplexing*

   Dibutuhkan oleh beberapa proses untuk mengirimkan atau menerima segment bila hanya ada satu UDP tiap *host*



Gambar 2.5 Multiplexing/Demultiplexing

2.3 TCP (Transfer Control Protocol)

Berbeda dengan UDP, koneksi TCP bersifat *connection-oriented.* Proses *handshaking* dilakukan untuk mengenali pengirim, penerima sebelum pengiriman data dilakukan. Koneksi yang dilakukan secara *point to point,* yaitu satu pengirim untuk satu penerima. *Header* TCP terdiri dari :

1. *Source port number* : nomor port yang digunakan oleh proses pada host sumber
2. *Destinaton port number* : nomor *port* yang digunakan oleh proses pada host tujuan
3. *Sequence number* : nomor yang digunakan untuk *byte* pertama dari data pada segment.

4. *Acknowledgement number* : menentukan nomor *byte* data selanjutnya.
5. *Header length* : menampilkan nomor dari empat *byte* kata pada *header.*
6. *Reserved* : field ini digunakan untuk kepentingan yang belum ditentukan.
7. *Control* : menentukan enam bit atau *flag* control yang berbeda
   - URG : urgent data
   - ACK : acknowledgement valid
   - PSH : push data
   - RST : reset connection
   - SYN : synchronize sequence number
   - FIN : terminate connection
8. *Window size* : menentukan ukuran dari jendela pada penerima
9. *Checksum* : berisikan *checksum*
10. *Urgent pointer* : digunakan bila segment yang dikirim terdapat data *urgent.*
11. *Option* : tempat untuk informasi tambahan yang ingin disampaikan pada penerima.

TCP menggunakan element yang sama seperti yang digunakan oleh UDP ditambah beberapa element lainnya yaitu :

1. *Connection management :*
   - *Connection establishment* : menentukan jalur virtual antara pengirim dan penerima.
   - *Data transmission* : semua pesan dikuirim melalui jalur virtual yang telah dibuat.

- *Connection termination* : penutupan jalur saat semua data telah dikirimkan.

2. *Error control* : digunakan untuk memperbaiki data yang hilang atau rusak
3. *Flow control*: digunakan agar pengirim tidak membanjiri penerima.
4. *Congestion control*: digunakan agar pengirim tidak membanjiri jaringan

2.4 IP (Intenet Protocol)

Protokol IP merupakan inti dari protokol TCP/IP. Seluruh data yang berasal dari protokol pada layer di atas IP harus dilewatkan, diolah oleh protokol IP, dan dipancarkan sebagai paket IP, agar sampai ke tujuan. IP terletak di layer ketiga dalam referensi model OSI yaitu layer *network*. IP adalah sebuah alamat logika yang digunakan untuk memberikan alamat kepada komputer yang terhubung ke jaringan yang menggunakan protokol TCP/IP. Alamat logika ini sering disebut sebagai alamat IP atau *IP Address*. Setiap komputer yang terhubung dengan internet harus mempunyai alamat IP yang unik dan tidak boleh sama.

<table>
<tr><td>Version</td><td>Header Length</td><td>Type of Service</td><td colspan="2">Total Length of Datagram</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="5">Options Strict Source Routing, Loose Source Routing</td></tr>
<tr><td colspan="5">DATA</td></tr>
</table>

Gambar 2.6 Format datagram IP

Format datagram IP terdiri atas:

1. *Version*, versi dan protokol IP yang dipakai. Pada saat ini versi IP yang dipakai adalah IP versi 4.
2. *Header Length*, berisi panjang dan header paket IP ini dalam hitungan 32 bit word.
3. *Type Length of Service*, berisi kualitas *service* yang dapat mempengaruhi cara penanganan paket IP ini.
4. *Total Length of Datagram*, panjang IP datagram total dalam ukuran byte.
5. *Identification, Flags*, dan *Fragment Offset*, berisi beberapa data yang berhubungan dengan *fragmentasi* (dipecah menjadi beberapa paket yang lebih kecil).
6. *Time to Live*, berisi jumlah router atau *loop* maksimal yang boleh dilewati paket IP.
7. Protokol, mengandung angka yang mengidentifikasikan protokol layer atas pengguna isi data dan paket IP ini.
8. *Header Checksum*, berisi nilai *checksum* yang dihitung dan seluruh *field* dan *header* paket IP.
9. *IP address* pengirim dan penerima data, berisi alamat pengirim paket dan penerima paket.
10. *Byte Option*, berisi *Strict Source Route* (SSR) dan *Loose Source Route* (LSR). SSR berisi daftar lengkap *IP address* dan router yang harus dilalui oleh paket ini dalam perjalanan ke *host* tujuan. SSR paket yang

dikirim diharuskan singgah di beberapa router.

Format *IP Address* bisa dinyatakan dalam bentuk antara lain :

a. Bentuk Biner

IP address merupakan bilangan biner 32 bit yang dipisahkan oleh tanda pemisah berupa tanda titik setiap 8 bit. Bentuk *IP address* adalah sebagai berikut:

xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx

Simbol "x" dapat digantikan oleh angka 0 dan 1, misalnya:

10000100.1011 100.1111001.00000001

b. Bentuk *dotted decimal*

Format penulisan "*dotted-decimal notation*" (notasi desimal bertitik). Setiap bilangan desimal tersebut merupakan nilai dari satu oktet (8 bit) IP address. Tiap oktet mewakili bilangan desimal dan 0 sampai *255.*

bit 0 31

10000100 01011100 01111001 00000001

132 92 121 1

132.92.121.1

2.5 ICMP (Internet Control Message Protocol)

ICMP (*Internet Control Message Protocol*) adalah protokol yang bertugas mengirimkan pesan-pesan kesalahan dan kondisi lain yang memerlukan perhatian khusus. Pesan / paket ICMP dikirim jika terjadi masalah pada layer IP dan layer diatasnya (TCP/UDP).

Karakteristik dari ICMP adalah :

- ICMP merupakan bagian internal dari IP dan diimplementasikan disetiap module IP
- ICMP digunakan untuk menyediakan *feedback* tentang beberapa *error* pada sebuah proses datagram.
- Tidak mendukung kehandalan pengiriman paket IP. Datagram / paket bisa tidak terkirim dan tidak ada *report* pemberitahuan tentang kehilangan datagram. Jika diperlukan adanya kehandalan maka harus diimplementasikan pada layer transport (pada arsitektur TCP /IP).
- Tidak ada respon ICMP yang dikirimkan untuk menghindari adanya perulangan tak terbatas, kecuali respon dari *query message* (ICMP type 0, 8-10, 13-18).
- ICMP *error message* tidak pernah dikirimkan sebagai respon sebuah datagram untuk tujuan broadcast atau multicast.

Setiap ICMP *error message* terdiri dari *Internet Header* dan sekurangnya 8 data octet *(byte)* pada datagram. Pada field *Type* menspesifikasikan tipe dari *message*, sementara *error code* pada datagram dilaporkan oleh *ICMP message* pada field *Code*. Pengertian setiap *code* bergantung dari tipe *message*.

Ada dua tipe pesan yang dapat dihasilkan oleh ICMP yaitu *ICMP Error Message* dan *ICMP Query Message*. *ICMP Error Message* sesuai namanya dihasilkan jika terjadi kesalahan pada jaringan. Sedangkan *ICMP Query Message* ialah jenis pesan yang dihasilkan oleh protocol ICMP jika pengirim paket menginginkan informasi tertentu yang berkaitan dengan kondisi jaringan.

ICMP Error Message dibagi menjadi beberapa jenis diantaranya :

1. *Destination Unreachable*, pesan ini dihasilkan oleh router jika pengiriman paket mengalami kegagalan akibat masalah putusnya jalur, baik secara fisik maupun secara logic. *Destination Unreachable* ini dibagi menjadi beberapa tipe. Beberapa tipe yang penting adalah:
    - *Network Unreachable*, jika jaringan tujuan tak dapat dihubungi
    - *Host Unreachable*, jika host tujuan tak bisa dihubungi
    - *Protocol at Destination is Unreachable*, jika ditujuan tak tersedia protokol tersebut
    - *Port is Unreachable*, jika tidak ada *port* yang dimaksud pada tujuan
    - *Destination Network is Unknown*, jika *network* tujuan tak diketahui
    - *Destination Host is Unknown*, jika host tujuan tidak diketahui
2. *Time exceeded*, Paket ICMP jenis ini dikirimkan jika isi field TTL dalam paket IP sudah habis dan paket belum juga sampai ke tujuannya. Sebagaimana telah diterangkan pada bagian IP diatas, tiap kali sebuah paket IP melewati satu router, nilai TTL dalam paket tersebut dikurangi satu. TTL ini diterapkan untuk mencegah timbulnya paket IP yang terus menerus berputar di network karena suatu kesalahan tertentu, sehingga menghabiskan sumberdaya jaringan yang ada.
3. *Parameter Problem*, paket ini dikirimkan jika terdapat kesalahan parameter pada *header* paket IP

4. *Source Quench*, Paket ICMP ini dikirimkan jika router atau tujuan mengalami *kongesti*. Sebagai respons pada paket ini, pihak pengirim paket harus memperlambat pengiriman paketnya.
5. *Redirect*, paket ini dikirimkan jika router merasa *host* mengirimkan paket IP melalui router yang salah. Paket ini seharusnya dikirimkan melalui router lain.

Sedangkan *ICMP Query Messages* terdiri atas:

1. *Echo* dan *Echo Reply*. Bertujuan untuk memeriksa apakah system tujuan dalam keadaan aktif. Program *ping* merupakan program pengiriman paket ini. Responder harus mengembalikan data yang sama dengan data yang dikirimkan.
2. *Timestamp* dan *Timestamp Reply*. Menghasilkan informasi waktu yang diperlukan system tujuan untuk memproses suatu paket.
3. *Address mask*, Untuk mengetahui berapa *netmask* yang harus digunakan oleh suatu *host* dalam suatu network.

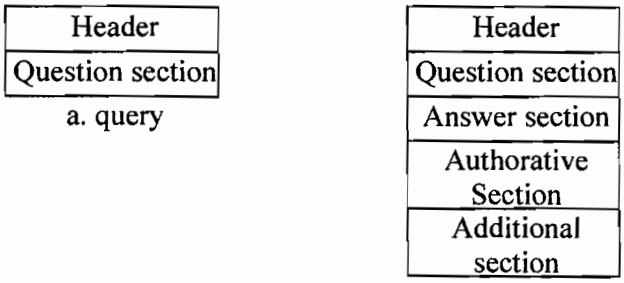
2.6 DNS (Domain Name System)

DNS merupakan sebuah protokol yang memetakan sebuah alamat IP ke dalam bentuk nama *host*. Nama *host* haruslah unik seperti halnya alamat IP. Dalam pemberian nama *host,* DNS menggunakan metode *hierarchical*. Nama *host* merupakan gabungan dari beberapa bagian seperti bagian awal merupakan asal dari organisasi *host,* lalu bagian kedua merupakan nama organisasi *host* tersebut dan bagian terakhir merupakan sifat organisasi tersebut. Berikut contoh beberapa nama domain :

Label	Description
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
int	International organizations
mil	Military groups
net	Network support centers
org	Nonprofit organizations

Gambar 2.7 Contoh Beberapa Nama Domain

Protokol ini menggunakan port 53. Bentuk pesan dari sevice ini ada 2 yaitu *query* dan *response.* Format lengkap dari pesan DNS dapat dilihat pada gambar 2.8



Gambar 2.8 Format Pesan DNS

2.7 FTP (File Transfer Protocol)

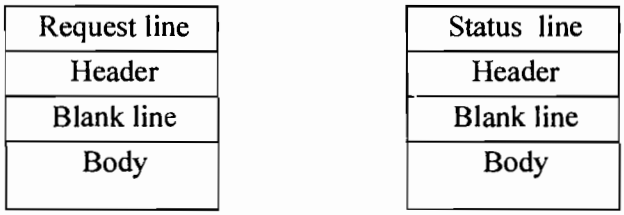
FTP merupakan protokol standar untuk mengirim suatu file dari satu *host* ke *host* yang lain. protokol ini digunakan menghadapi sistem yang heterogen dalam jaringan internet. Keheterogenan sistem ini meliputi perbedaan struktur, perbedaan cara merepresentasikan data dan konversi nama file. FTP menggunakan 2 *port* dalam melakukan koneksi yaitu *port* 21 untuk kontrol koneksi dan *port* 20 untuk koneksi data. Kontrol koneksi tetap terhubung selama

service dijalankan sedangkan koneksi data dibuka dan ditutup untuk setiap file yang dikirim. Berikut contoh *response code* dari FTP :

Response	Description
200	Command OK
331	Username OK Password is Needed
425	Cannot open data connection
530	Cannot login anonymous

Gambar 2.9 Response Code FTP

Format pesan FTP ada 2 yaitu *request* dan *response*, dapat dilihat pada gambar 2.10



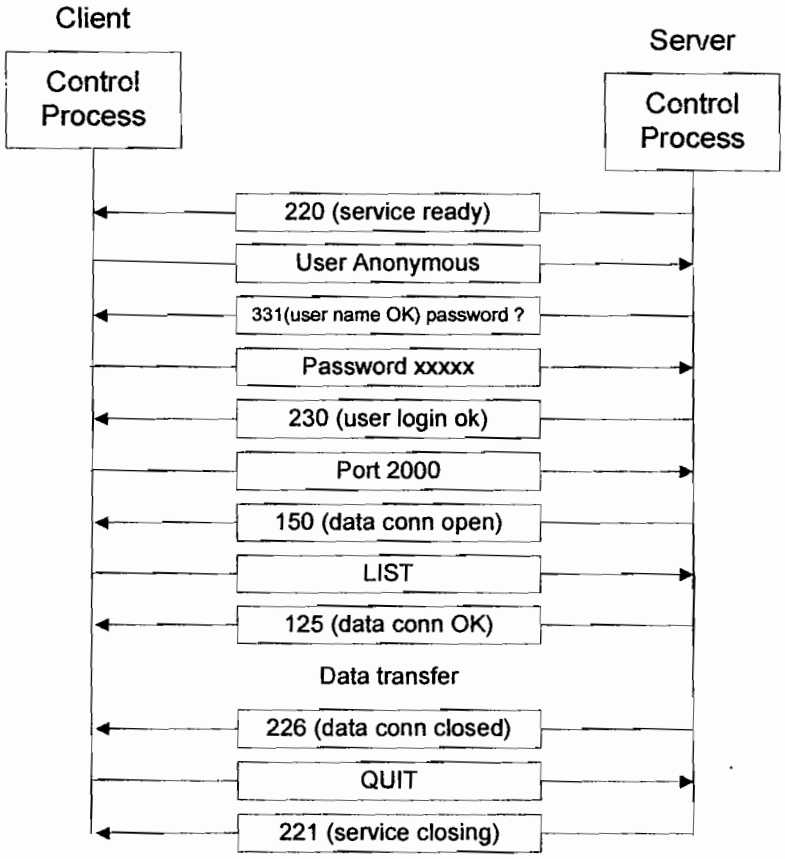
a. request a. response

Gambar 2.10 Format Pesan FTP

Perintah yang digunakan dalam komunikasi FTP :

1. USER = username
2. PASS = password
3. LIST = meminta daftar file yang terdapat pada suatu direktori
4. RETR = mengambil file
5. STOR = menyimpan file

Dalam memulai koneksi, FTP memerlukan user name beserta *password.* Berikut proses *requst reply* pada FTP :



Gambar 2.11 Proses Request Reply Pada FTP

2.8 HTTP (Hypertext Transfer Protocol)

HTTP merupakan protokol yang digunakan untuk mengirimkan data dalam jaringan WAN. Fungsi dari protokol ini merupakan kombinasi dari FTP dan SMTP. HTTP mengirimkan data dan menggunakan service TCP seperti FTP tapi hanya menggunakan satu koneksi. Data yang dikirim mirip dengan pesan SMTP tapi pengirimannya lebih cepat. Protokol ini menggunakan *port* 80 dalam

melakukan servicenya. Format pesan HTTP ada 2 yaitu *request* dan *response*, dapat dilihat pada gambar 2.12

Request line
Header
Blank line
Body

a. request

Status line
Header
Blank line
Body

a. response

Gambar 2.12 Format Pesan HTTP

Berikut beberapa *response code* dari protokol HTTP :

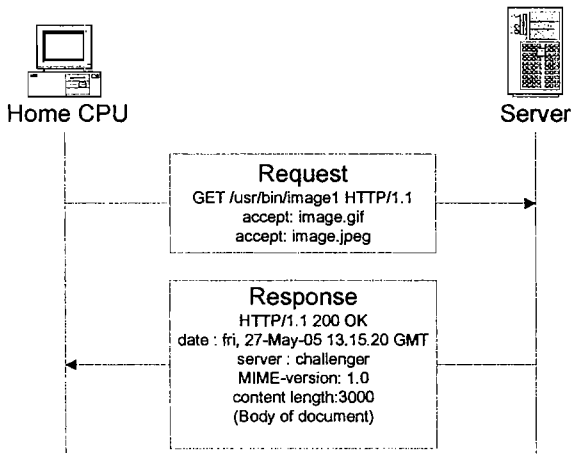
Response	Description
200	OK Request succesed
301	Moved permanently Request object moved
400	Bad Request Request message not understood by server
404	Not Found Requested document not found in this server
505	HTTP version not supported

Gambar 2.13 Response Code HTTP

Dan beberapa perintah yang biasanya digunakan untuk mengakses protokol HTTP adalah :

1. GET : digunakan untuk mengambil dokumen dari server
2. HEAD : untuk mendapatkan informasi tentang dokumen dari server tetapi bukan dokumen itu sendiri.
3. POST : digunakan untuk memberikan informasi ke server
4. PUT : digunakan untuk meletakkan dokumen baru pada server
5. DELETE : digunakan untuk menghapus dokumen pada server.

Dalam program ini nantinya akan digunakan perintah HEAD untuk mendapatkan dokumen dari server HTTP. Dari dokumen tersebut nantinya bagian yang digunakan sebagai masukan dalam program ini hanya bagian header. Dalam bagian header tersebut terdapat *response code* dari service HTTP yang memberitahukan kondisi dari service tersebut. Proses *request reply* HTTP dapat dilihat dalam gambar berikut ini :



Gambar 2.14 Proses Request Reply Pada HTTP

2.9 Java

Java dibuat dan diperkenalkan pertama kali oleh sebuah tim *Sun Microsystem* yang dipimpin oleh Patrick Noughton dan James Gosling pada tahun 1991 dengan *code name* ***Oak***. Tahun 1995 Sun mengubah nama *Oak* tersebut menjadi *Java*. Ide pertama kali kenapa Java dibuat adalah karena adanya motivasi untuk membuat sebuah bahasa pemrograman yang bersifat *portable* dan *platform independent* (tidak tergantung mesin dan sistem operasi) yang dapat digunakan untuk membuat peranti lunak yang dapat ditanamkan (*embendded*) pada berbagai macam peralatan elektronik konsumer biasa, seperti: *remote control*, telepon, *card reader* dan sebagainya. Peralatan-peralatan elektronik konsumer ini dapat menggunakan berbagai CPU (*central processing unit*) yang berbeda yang digunakan sebagai *controller* (Indrajani dan Martin, 2004).

Ini mengakibatkan pembuatan peranti lunak untuk setiap peralatan elektronik ini menjadi sulit dan mahal. Karena alasan inilah maka muncul suatu ide yang kemudian mendasari dan melahirkan Java. Kenyataan ini mungkin sedikit mengejutkan kerena banyak yang mengira alasan munculnya Java adalah karena adanya internet. Internet memang membantu menjadikan Java sangat terkenal seperti sekarang ini. Java didesain dengan ide utama adalah portabilitas dengan konsep *write once run everywhere*.

Untuk memenuhi konsep ini maka hasil kompilasi Java bukanlah merupakan *native code* sebagaimana yang dihasilkan bahasa pemrograman lainnya seperti C/C++ ataupunVisual Basic melainkan berupa *bytecode*. Teknologi Java diadopsi oleh Netscape tahun 1996, JDK 1.1 , kemudian JDK 1.2,

berikutnya J2EE (Java 2 Enterprise Edition) yang berbasis J2SE yaitu servlet, EJB dan JSP, dan yang terakhir adalah J2ME (Java 2 Micro Edition).

2.9.1 Java.net

Java menyediakan suatu paket yang mendukung tipe koneksi *Connection oriented* dan *Connectionless*, yaitu *java.net*. Secara umum pada paket java.net berisi kelas-kelas dan interface yang menyediakan API (*Application Programming Interface*) level rendah (TCP dan UDP) dan level tinggi (*URL connection*). Berikut contoh sederhana API pada level rendah :

```
import java.net.*;
public class Tes {
        public static void main(String[] args) {
        try {
                InetAddress ip = InetAddress.getByName("localhost");
                System.out.println(ip.getHostAddress());
                System.out.println(ip.getHostName());
        }
        catch(UnknownHostExeption e) {
                System.out.println(e);
        }
    }
}
```

2.9.2 Multithreading

Multithreading tersusun dari dua kata yaitu *multi* dan *thread. Thread* lebih ditujukan untuk sebuah aliran kontrol dalam sebuah program. Dalam suatu aplikasi dapat berjalan lebih dari satu dan semua *thread* menggunakan *space address* memori yang sama, sehingga satu *thread* dengan *thread* lain dalam satu program dapat berbagi *variabel public*. Oleh karena banyak *thread* dapat mengakses *variabel public*, antara satu *thread* dengan *thread* lainnya harus ada sinkronisasi. Bahasa pemrograman Java sejak awal mendukung karakteristik multithreading (Budi Susanto, 2003).

Untuk mendukung multithreading pada Java, sebuah *thread* mulai dibuat ketika dilakukan pembuatan objek baru., dimana objek tersebut merupakan *instance* dari sebuah kelas yang diterapkan dari antarmuka *java.lang.Runnable* atau diturunkan dari *java.lang.Thread*. Sebuah objek *thread* akan mewakili sebuah *thread* sesungguhnya dan juga memberikan penanganan kontrol dan sinkronisasi eksekusinya. Sebuah objek yang diperlakukan sebagai sebuah *thread,* harus menerapkan dari sebuah interface *java.lang.Runnable*. Interface Runnable menyediakan satu method kunci yaitu run(). Berikut gambaran interface Runnable:

```
Public interface Runnable{
        Abstract public void run()
}
```

Dengan memanggil method run() dari sebuah objek *thread,* sebuah *java.lang.Runnable* akan dimulai siklus hidupnya.

Untuk menerapkan pemrograman multithreading *java.lang.Runnable* dengan Java, dapat dilakukan dengan dua teknik umum, yaitu dengan mendefinisikan sebuah kelas yang menerapkan dari *java.lang.Runnable* atau mendefinisikan sebuah kelas yang diturunkan dari kelas *java.lang.Thread* (Budi Susanto, 2003)

2.9.3 JRE (Java Runtime Environment)

Program ini dibuat dalam bentuk aplikasi dengan user interface grafis. Untuk itu diperlukan JRE. JRE merupakan bentuk implementasi dari JVM (*Java Virtual Machine*). JRE menyediakan *library*, JVM dan komponen lainnya untuk menjalankan aplikasi yang dibuat menggunakan bahasa pemrograman Java. JVM adalah lingkungan tempat eksekusi program Java berlangsung dimana para objek saling berinteraksi satu dengan yang lainnya. *Virtual Machine* inilah yang menyebabkan Java mempunyai kemampuan penanganan memori yang lebih baik, keamanan yang lebih tinggi serta portabilitas yang besar. Oleh Sun, developer Java, disediakan berbagai JRE sesuai dengan Sistem Operasi yang digunakan.

BAB III

ANALISA DAN DESAIN SISTEM

3.1 Analisa Sistem

Setiap koneksi jaringan yang dilakukan dengan menggunakan alamat IP akan dimulai dengan proses *handshaking*. Dalam proses ini sebuah *node* akan melakukan *connection establishment* kepada *node* yang dituju untuk mengetahui keadaan *node* tersebut. Proses ini juga bertujuan membentuk suatu jalur virtual yang nantinya akan dilewati oleh data yang akan dikirimkan (protokol TCP).

Dalam proses ini akan dikirimkan sebuah *request* kepada *node* yang dituju, bila ada *reply* maka koneksi akan dilanjutkan. Bentuk request paling sederhana yang biasa dilakukan adalah *ping*. Pesan *ping* ini disediakan oleh protokol service ICMP. Sedangkan untuk protokol service lainnya pesan harus dikirimkan ke alamat IP dan *service* yang aktif digunakan oleh *node* tersebut. Setiap service memiliki kode *reply* berbeda. Dan cara yang digunakan untuk mendapatkan *reply* juga berbeda. Untuk service HTTP yang biasanya merupakan service untuk *website, request* yang dikirimkan berupa pesan untuk mendapatkan halaman utama dari *website host* tersebut. Dan *reply* yang didapatkan berupa *header* berisikan kode status service tersebut. Sedangkan untuk service FTP, *request* yang dikirimkan untuk mendapatkan form *login* dari service tersebiut. Jadi *request* berupa alamat *host* dan *port* dari service FTP. *Reply* dari *request* inilah yang nantinya digunakan oleh program ini sebagai masukan. Selama ada kode status pada *reply* sesuai maka program akan mengasumsikan node tersebut dalam

keadaan aktif. Dan untuk service DNS, *request* yang dikirimkan berupa alamat *host* dari *node* yang diinginkan. Dan *reply* yang dikirimkan oleh service ini berupa alamat IP dari *node* tersebut (*forwarding*) ataupun sebaliknya (*reverse*).

3.2 Kebutuhan Sistem

Program *network view* yang akan dirancang merupakan sebuah perangkat lunak yang dapat digunakan untuk proses monitoring jaringan internet. Pada bagian ini akan dijelaskan beberapa kebutuhan dalam perancangan program bantu tersebut.

3.2.1 Kebutuhan Perangkat Lunak

Adapun kebutuhan perangkat lunak yang digunakankan dalam pembuatan program antara lain :

- Sistem Operasi :
  - Microsoft Windows XP Profesional Edition
- Programming :
  - Java 1.4 SDK
  - JRE 1.5 for Windows
  - JRE 1.5 for Linux
  - FTP server
  - DNS server
  - Web server / HTTP server

Spesifikasi minimum untuk kebutuhan perangkat lunak adalah sebagai berikut :

- Microsoft Windows 98

3.2.2 Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras yang digunakan antara lain :

- PC dengan prosessor Intel Pentium IV 1,6 MHz.
- Memori DDRAM 512 MB.
- Harddisk dengan kapasitas 40 GB
- *Lan card*
- *VGA Gforce* IV 4000 8x 128Mb
- Monitor (Resolusi yang digunakan 1024)

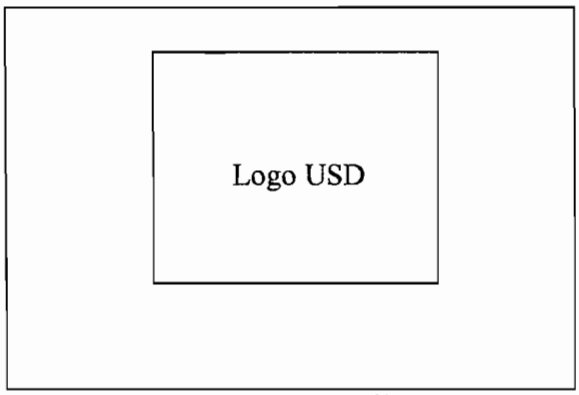
Spesifikasi minimum untuk kebutuhan perangkat keras adalah sebagai berikut :

- PC dengan prosessor Intel Pentium III
- Memori SDRAM 64 MB.
- Harddisk dengan kapasitas 10 GB
- *Lan card*
- *VGA 32Mb*
- Monitor

3.3 Desain Interface Program

Interface (antarmuka) program disini menggunakan grafis. Hal ini digunakan untuk mempermudah *user* dalam menjalankan program (*user friendly*). Fasilitas yang disediakan oleh program akan berbentuk menu – menu beserta sub menunya.

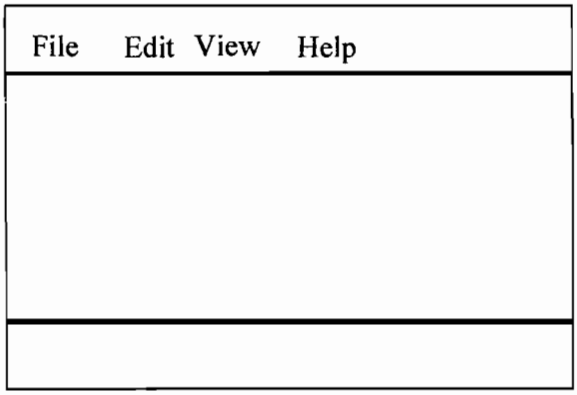
3.3.1 Tampilan Utama



Gambar 3.1 Tampilan Utama

Saat pertama dieksekusi maka akan muncul tampilan seperti diatas berisi animasi logo universitas.

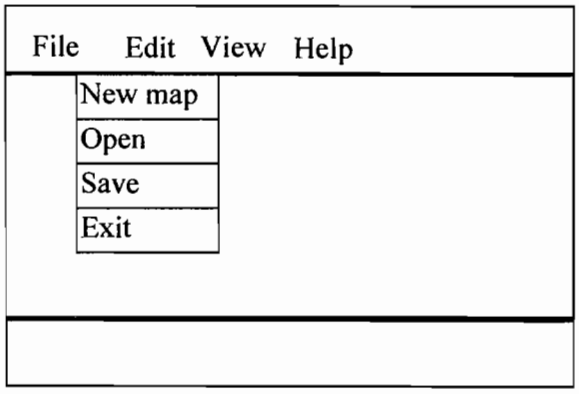
3.3.2 Tampilan Menu Utama



Gambar 3.2 Tampilan Menu Utama

Tampilan selanjutnya dapat dilihat seperti gambar 3.2. Disini terdapat beberapa pilihan pada *toolbar* yaitu menu file, edit, view dan help. Pada tiap tiap menu terdapat beberaba sub menu.

3.3.3 Tampilan Menu File

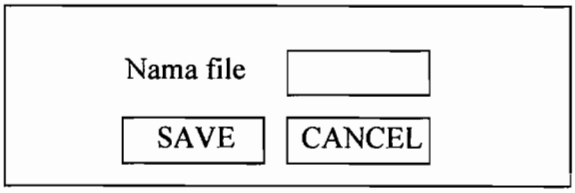


Gambar 3.3 Tampilan Menu File

Saat *user* memilih menu file maka *user* dapat memilih beberapa submenu yaitu :

1. New map : digunakan untuk membuat peta jaringan baru yang akan dimonitor.
2. Open : untuk melihat peta jaringan yang telah ada.
3. Save : digunakan untuk menyimpan peta jaringan yang telah dibuat
4. Exit : digunakan untuk keluar dari program.

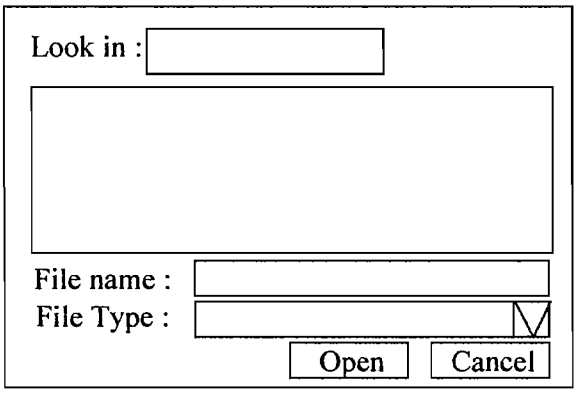
3.3.3.1 Tampilan Sub Menu New Map



Gambar 3.4 Tampilan Sub Menu New Map

Saat membuat peta baru *user* dapat memasukkan nama dari peta yang akan dibuat tersebut. Data node dari peta tersebut nantinya akan disimpan pada file berbentuk *text* dengan nama masukan dan disimpan pada folder *My Document*

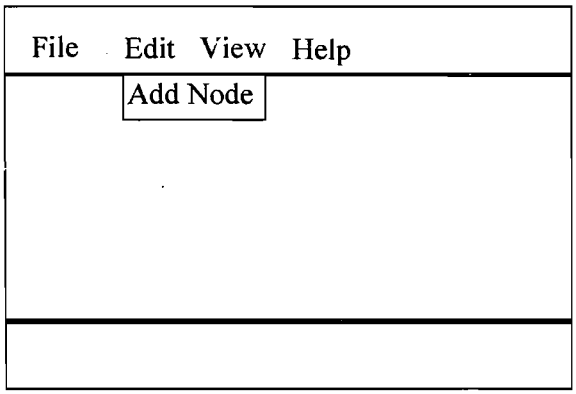
3.3.3.2 Tampilan Sub Menu Open



Gambar 3.5 Tampilan Sub Menu Open Map

Sub menu ini dapat digunakan untuk membuka file map yang telah dibuat sebelumnya. Browser akan langsung menuju ke folder *My Document* karena folder tersebut di setting sebagai *default* penyimpanan file dengan extensi *.map.*

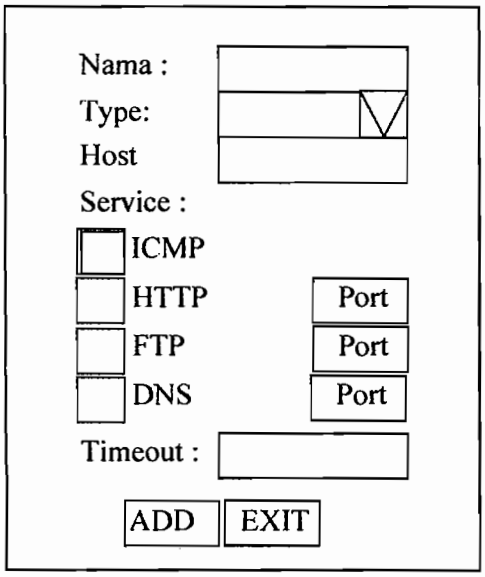
3.3.4 Tampilan Menu Edit



Gambar 3.6 Tampilan Menu Edit

Menu edit digunakan untuk membuat *node* pada peta jaringan yang akan dimonitor. Saat *user* memilih submenu add node maka akan ada form untuk memasukkan *node* tersebut. Dalam form itu terdapat bebarapa field yang harus diisi seperti dapat dilihat pada gambar 3.6.

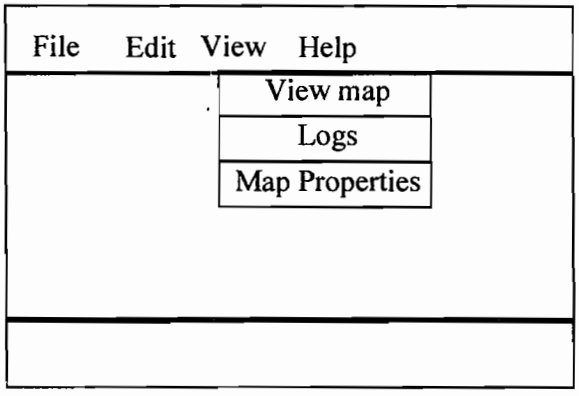
3.3.4.1 Tampilan Sub Menu Add Node



Gambar 3.7 Tampilan Sub Menu Add Node

Data- data node yang dimasukkan pada form ini akan disimpan pada file yang dibuat pada awal pada sub menu new map.

3.3.5 Tampilan Menu View

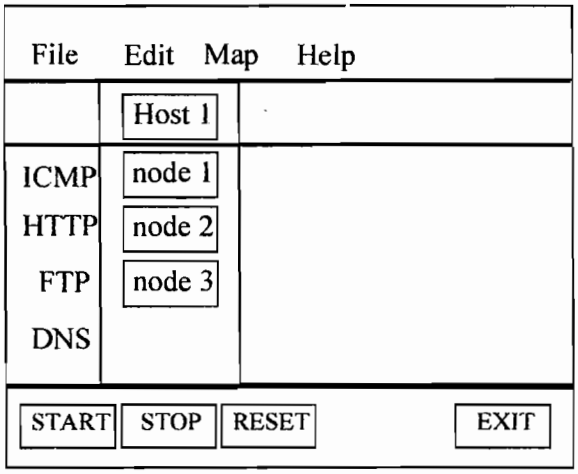


Gambar 3.8 Tampilan Menu Map

Saat tombol map dipilih maka akan diberikan pilihan dalam 3 sub menu yaitu view map, logs dan map properties. Sub menu view map digunakan untuk

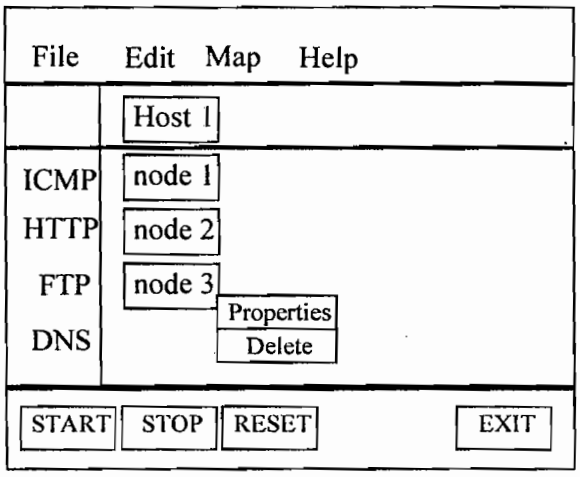
melihat peta jaringan yang sedang dimonitoring, sub menu logs digunakan untuk menampilkan daftar logs dari semua node dalam peta tersebut dalam sebuah laporan. Dan sub menu map properties digunakan untuk mengatur waktu *polling* peta jaringan tersebut. Untuk tampilan sub-sub menu diatas dapat dilihat seperti gambar 3.8.

3.3.5.1 Tampilan Sub Menu View Map



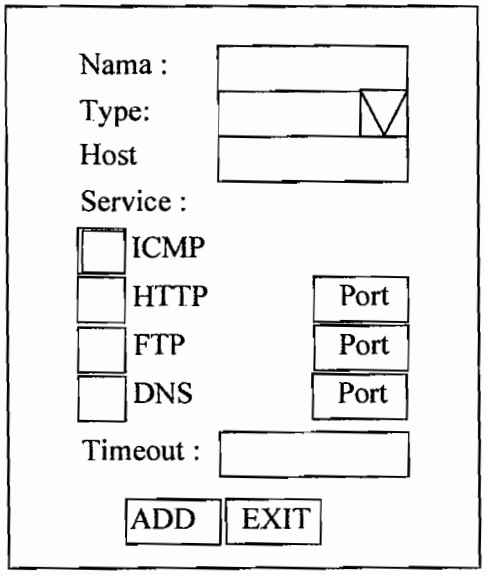
Gambar 3.9 Tampilan Sub Menu View Map

Pada sub menu ini terdapat tombol tambahan *start* yang digunakan untuk memulai mengirimkan request sesuai waktu polling. Tombol *stop* digunakan untuk menghentikan proses monitoring dengan menghentikan pengiriman request. Sedangkan tombol *reset* digunakan untuk mengeset node – node dalam peta seperti keadaan awal. Saat node pada peta di klik kanan maka akan muncul tampilan seperti gambar 3.10.



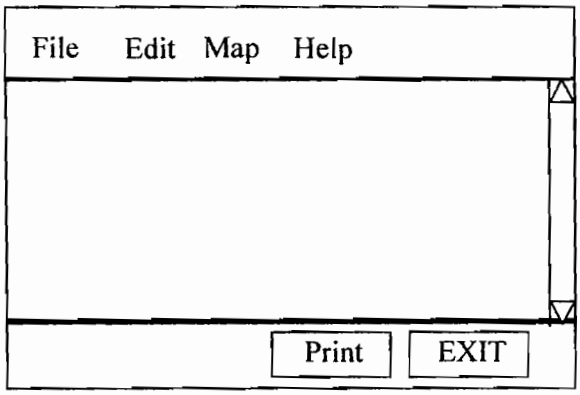
Gambar 3.10 Tampilan Right Click pada Sub Menu View Map

Sub menu *properties* dapat digunakan untuk melihat data dari node tersebut seperti nama node, tipe node, host, service, port dan timeout yang digunakan untuk monitoring. Sub menu ini juga dapat digunakan untuk mengubah data tersebut diatas. Sedangkan sub menu *delete* digunakan untuk menghapus node tersebut dari peta. Tampilan untuk sub menu properties dapat dilihat pada gambar 3.11.



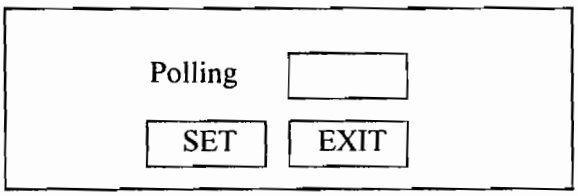
Gambar 3.11 Tampilan Sub Menu Properties

3.3.5.2 Tampilan Sub Menu Logs



Gambar 3.12 Tampilan Sub Menu Logs

3.3.5.3 Tampilan Sub Map Properties



Gambar 3.13 Tampilan Sub Menu Map Properties

Sub menu ini digunakan untuk mengubah waktu *polling* peta. Satuan waktu yang digunakan adalah detik. *Default* dari program adalah 60 detik

3.4. Desain Penyimpanan Data

Dalam program ini tidak digunakan tabel, semua data dalam program nantinya akan disimpan dalam bentuk teks. File berisi data dari node ini nantinya akan disimpan dengan ekstensi *.map*. Data dari peta saat peta itu disimpan akan disimpan dengan format sebagai berikut :

nonode<koma>namanode<koma>type<koma>host<koma>service<koma>port <koma>timaout

Keterangan variabel :

- no node : nomor dari node yang diamati (2 karakter)
- nama node : nama dari node yang diamati (15 karakter)
- type : tipe dari node (7 karakter)
- host : alamat host (30 karakter)
- service : service yang digunakan node untuk monitoring (4 karakter)
- port : port yang digunakan service(5 karakter)
- timeout : waktu timeout pesan (2 karakter)

Format penyimpanan data log untuk semua *node* dalam peta adalah sebagai berikut :

node<tabs>sevice<:>port<:> date <tabs> time

Keterangan variabel:

- nama node : nama node yang diamati (15 karakter)
- service : type service yang digunakan untuk monitoring (4 karakter)
- time : waktu saat kegagalan pengiriman pesan terjadi (8 karakter) dengan format hour:minute:secon
- date : tanggal saat kegagalan pengiriman pesan terjadi (10 karakter) dengan format dd-mm-yyyy

File berisikan data *log* dari peta ini nantinya akan disimpan dengan ekstensi *nama_peta.log.* dan disimpan pada folder dimana program dijalankan.

3.5. Desain Pemrograman

Bahasa pemrograman yang digunakan untuk membuat program ini adalah Java 2SDK. *Package* yang digunakan antaranya :

1. javax.swing.*

Untuk tampilan frame digunakan class Jframe yang merupakan bagian dari package javax.swing.*

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
```

Package ini juga menyediakan class JButton yang nantinya digunakan untuk membuat *node – node* yang akan dimonitoring. Penggunaan JButton ini untuk memudahkan melakukan *polling* terhadap *node* yang dilakukan diluar waktu *polling* dari peta tersebut.

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.AbstractButton
                  └javax.swing.JButton
```

2. java.io.*

Untuk operasi file data, program ini menggunakan *package* java.io.* yang menyediakan class yang diperlukan seperti FileInputStream(), class ini digunakan untuk membaca baris data yang tersimpan.

```
java.lang.Object
  └java.io.InputStream
      └java.io.FileInputStream
```

3. java.net.*

Untuk melakukan proses *request reply* digunakan beberapa kelas yang berbeda sesuai dengan service yang dimonitoring. untuk service HTTP proses *request reply* dilakukan dengan membuka suatu *URL connection* dengan menggunakan class HttpURLConnection

```
java.lang.Object
  └java.net.URLConnection
      └java.net.HttpURLConnection
```

Dan untuk service FTP digunakan class Socket() untuk membuka koneksi pada service tersebut dilanjutkan dengan proses *login*

```
java.lang.Object
  └java.net.Socket
```

Sedangkan untuk service DNS digunakan class inetAddress() untuk mengirimkan alamat *host* dari *node* yang dimonitoring.

```
java.lang.Object
  └java.net.InetAddress
```

3.6 Alur Proses

Alur proses dalam program ini dapat dilihat sebagai berikut :

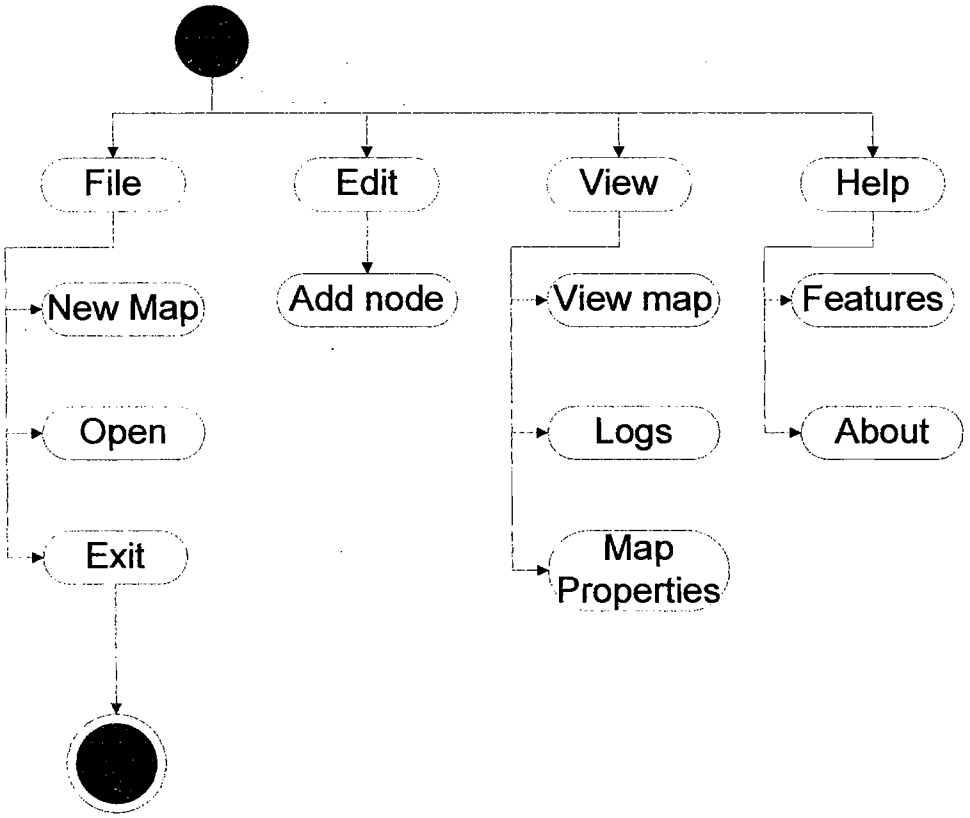
1. Setelah masuk kedalam program *user* dapat memilih sub menu new map pada menu file bila menjalankan program untuk pertama kali atau memilih sub menu open bila telah membuat peta monitoring sebelumnya.
2. Saat *user* memilih sub menu new map maka *user* akan diminta memasukkan nama dari map yang akan dimonitoring lalu klik tombol save, dan bila telah selesai *user* dapat menekan tombol exit. Untuk selanjutnya semua data dari

map ini akan disimpan sesuai nama yang diinputkan tadi dan akan tersimpan pada folder *c:\my document.*

3. Selanjutnya *user* dapat memilih sub menu add node pada menu edit untuk membuat *node – node* yang akan dimonitoring. Disini *user* dapat memasukkan semua data yang diminta oleh program.
4. Setelah selesai memasukkan data, maka peta akan membuka secara otomatis. Untuk memulai proses monitoring, *user* dapat menekan tombol Start. Warna node akan berubah seandainya program tidak menerima *reply node* tersebut. Dan untuk menghentikan proses monitoring maka *user* dapat menekan tombol Stop.
5. Untuk mengubah data dari *node* yang dimonitoring, *user* dapat meng-klik kanan mouse pada *node* dimaksud dan memilih sub menu properties. Sedangkan untuk menghapus suatu node dari peta, *user* dapat melakukan dengan meng-klik kanan mouse pada *node* tersebut dan memilih sub menu Delete.
6. Dan untuk melihat daftar logs untuk seluruh *node* dalam peta tersebut maka *user* harus memilih sub menu logs pada menu map.
7. Selesai.

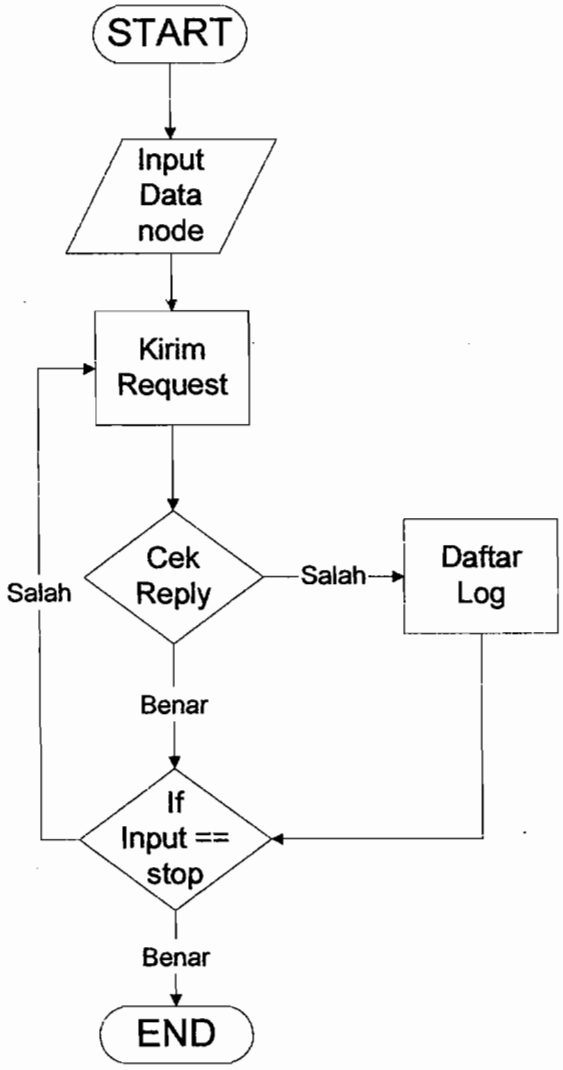
3.6 Flowchart Program

3.6.1 Alur Menu Program



Gambar 3.14 Alur Menu Program

3.6.2 Proses Request Reply



Gambar 3.15 Proses Request Reply

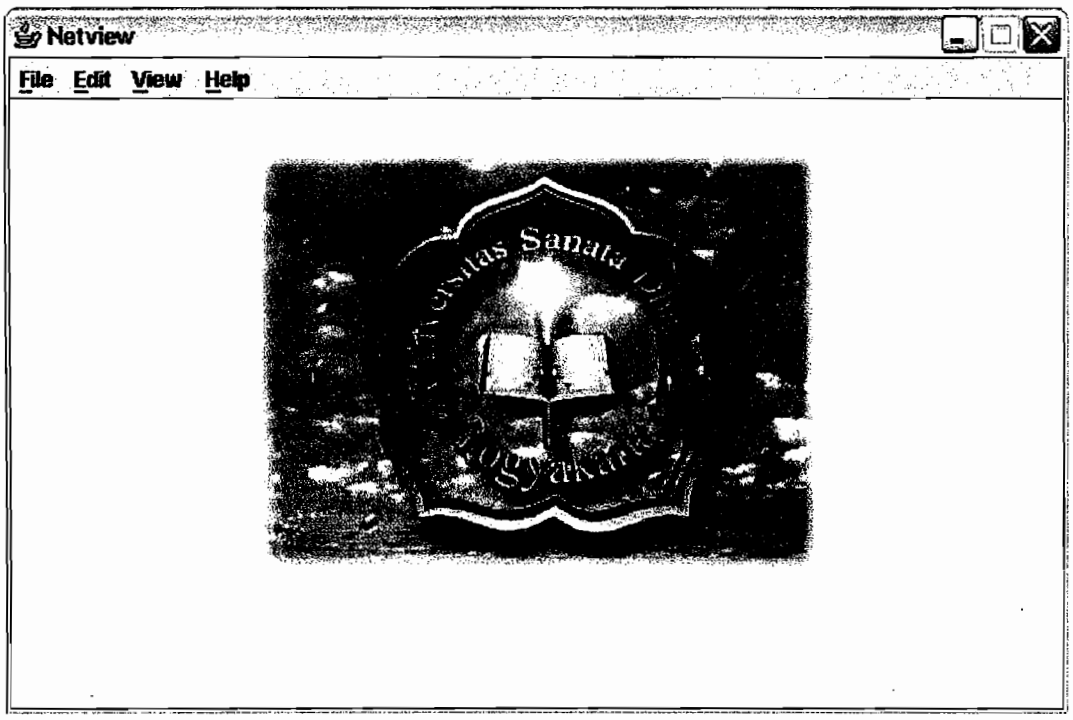
BAB IV

IMPLEMENTASI SISTEM

4.1 Implementasi Interface

4.1.1 Menu Utama

Tampilan utama pada program ini dapat dilihat pada gambar 4.1. Disini terdapat beberapa menu pada *toolbar* yaitu menu File, Edit, View dan Help.

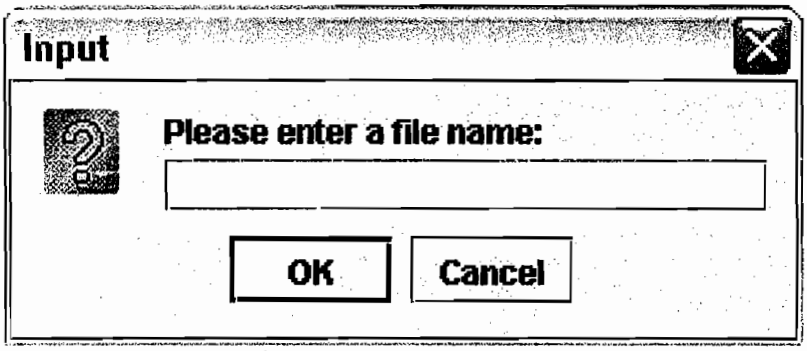


Gambar 4.1 Tampilan Utama

4.1.2 Sub Menu New Map

Untuk membuat suatu peta jaringan yang akan dimonitoring, *user* dapat memilih sub menu New Map pada menu File. Saat memilih sub menu ini, maka *user* harus memasukkan terlebih dahulu nama dari peta jaringan yang akan dibuat pada

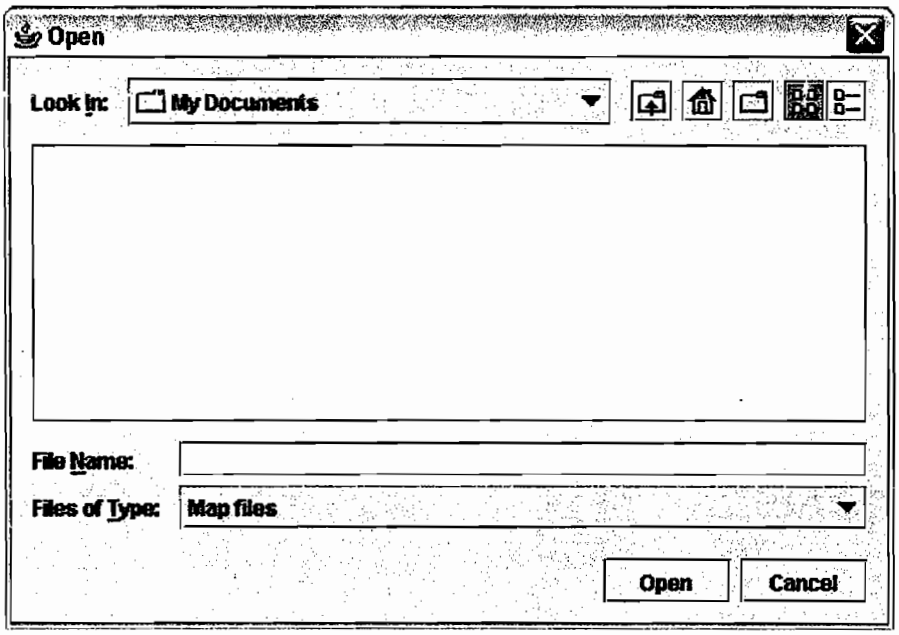
frame yang dapat dilihat pada gambar 4.2. Bila *user* tidak memasukkan nama file dan meng-klik tombol OK maka akan muncul peringatan "Nama File tidak Valid" dan bila *user* memasukkan nama file yang sudah ada maka akan muncul peringatan "Nama file sudah ada".



Gambar 4.2 Sub Menu New Map

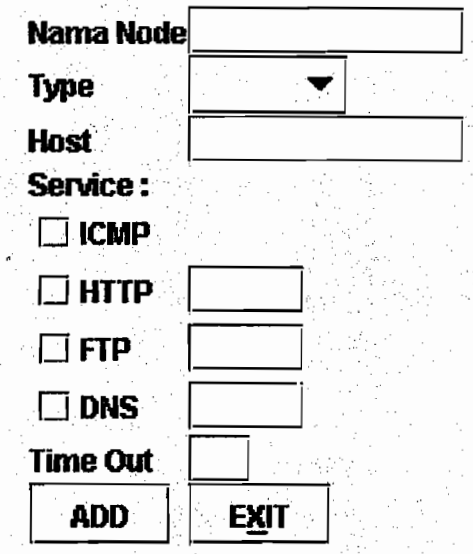
4.1.3 Sub Menu Open

Bila *user* ingin membuka peta jaringan yang telah dibuat sebelumnya, maka *user* dapat menggunakan sub menu Open untuk mencari peta jaringan yang dimaksud. Secara otomatis, frame ini akan menunjuk ke folder *My Document*. ini dilakukan untuk mempermudah *user* dalam pencarian file peta jaringan karena program akan menyimpan semua file dengan ekstensi *.map* ke folder tersebut. Tampilan sub menu open dapat dilihat pada gambar 4.3



Gambar 4.3 Sub Menu Open

4.1.4 Sub Menu Add Node



Gambar 4.4 Sub Menu Add Node

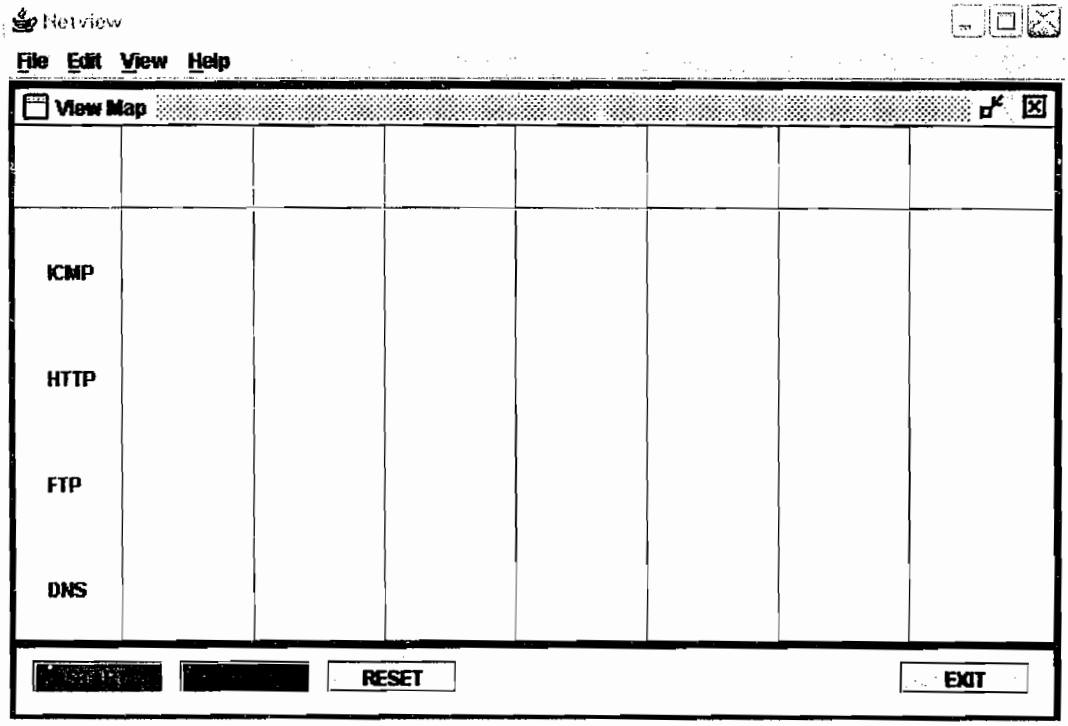
Untuk menambahkan *host* yang akan dimonitoring pada peta jaringan, maka *user* dapat memilih sub menu add node pada menu edit. Pada sub menu ini terdapat beberapa field yang harus diisi sebagai data dari *node* yang akan dimonitoring. Field-field tersebut diantaranya nama node, type, host, service, port dan time out. Untuk field nama node, host, time out user harus mengetikkan datanya. Sedangkan untuk field type disediakan *combo box* untuk memudahkan user dalam pengisian data. Pilihan pada *combo box* pada field type adalah host, router dan server.

Untuk memilih service yang akan dimonitoring pada host tersebut, user dapat memilih pada check box service yang disediakan. Service yang disediakan adalah ICMP, FTP, HTTP dan DNS. Field port akan terisikan secara otomatis sesuai dengan pilihan service yang dipilih oleh *user*, kecuali untuk sevice ICMP. Field ini dapat di edit sesuai kebutuhan. Bila *user* tidak mengisikan semua field data maka akan muncul peringatan "Data belum lengkap".

4.1.5 Sub Menu View Map

Pada sub menu ini, *user* dapat melihat peta jaringan dari *node – node* yang telah dimasukkan pada sub menu add node. Pada sub menu ini terdapat empat tombol yaitu start, stop, reset dan exit. Tombol start berfungsi untuk memulai proses *request reply* pada *node* yang terdapat pada peta jaringan. Tombol stop digunakan untuk menghentikan proses *request reply*. Sedangkan tombol reset digunakan untuk mengeset status semua *node* pada peta jaringan kembali seperti semula. Tombol ini diperlukan bila *user* ingin menyamakan kondisi tiap - tiap *node* setelah proses monitoring berjalan. Dan untuk menutup peta jaringan ini *user* dapat menekan tombol exit. Tampilan sub menu view map ini dapat dilihat pada gambar 4.5. Untuk

setiap *node* yang ditampilkan dapat mengirimkan *request* diluar waktu *polling* (*forced polling*) dengan meng-klik pada node tersebut. Untuk menambahkan servise yang dipantau user dapat mengklik kanan pada host *node* tersebut. Saat user meng-klik kanan pada *node* maka *user* dapat melihat properties dari *node* tersebut. Dalam sub menu properties ini, *user* dapat mengedit data port dari node tersebut. *User* juga dapat menghapus *node* tersebut dari peta jaringan dengan memilih delete.

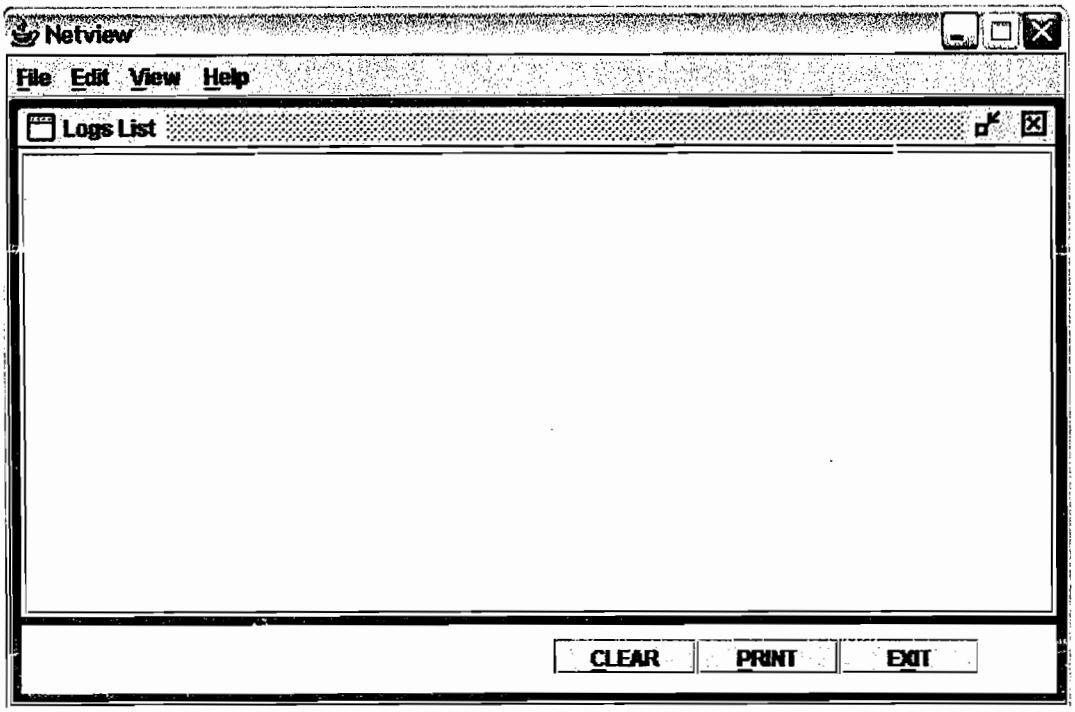


Gambar 4.5 Sub Menu View Map

4.1.6 Sub Menu Logs

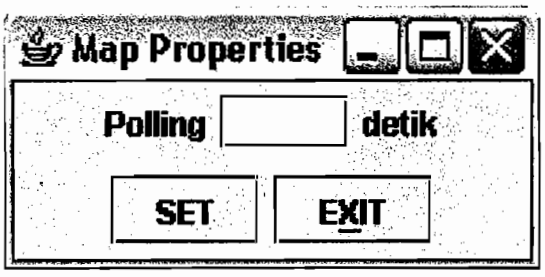
Saat terjadi kesalahan (*error*) pada proses pengiriman *request* maka kesalahan itu akan tercatat pada sub menu ini. Dalam sub menu ini terdapat tiga tombol yaitu clear, print dan exit. Tombol clear digunakan untuk menghapus semua daftar *request* yang tidak mendapatkan *reply* yang tercatat. Tombol print dapat

digunakan untuk mencetak semua kesalahan. Dan tombol exit digunakan untuk menutup sub menu ini. Tampilan sub menu ini dapat dilihat pada gambar 4.6.



Gambar 4.6 Sub Menu Logs

4.1.7 Sub Menu Map Properties



Gambar 4.7 Sub Menu Map Properties

Pada sub menu map properties, *user* dapat mengeset waktu yang akan digunakan untuk mengirimkan *request* pada peta jaringan. Fasilitas ini disediakan untuk membantu user agar tidak menganggu besar bandwidth dari jaringan dengan cara memperbesar selang waktu pengiriman request. Bila *user* mengosongkan waktu

polling dan menekan tombol set, maka akan muncul peringatan "Waktu polling belum diisi".

4.2 Implementasi Programming

Pembahasan implementasi programming disini dilakukan berdasarkan tampilan yang telah dibahas diatas.

4.2.1 Menu Utama

Pada tampilan utama terdapat frame utama beserta empat buah menu yang terletak pada menubar, beserta dengan animasi logo USD ditengahnya. Frame dan menubar tersebut dibuat dengan menggunakan class Jframe().

```
frame = new JFrame();
JMenuBar menubar = new JMenuBar();
frame.setJMenuBar(menubar);
```

Setiap menu yang terdapat pada frame utama dapat dipanggil dengan menggunakan shortcut pada keyboard. Seperti menu file dengan mengunakan tombol Alt + F, menu edit dengan tombol Alt + E, menu View dengan tombol Alt + V dan menu help dengan tombol Alt + H. Tombol pada keyboard dapat digunakan untuk memanggil menu - menu tersebut menggunakan perintah :

```
file.setMnemonic('F');
edit.setMnemonic('E');
view.setMnemonic('V');
help.setMnemonic('H');
```

Untuk menampilkan animasi logo USD tersebut fungsi main() memanggil class ImageDisplayer().

```
ImageDisplayer img = new ImageDisplayer();
```

4.2.2 Sub Menu New Map

Saat user memilih untuk membuat peta jaringan baru, user akan memasukkan nama dari peta jaringan pada frame yang diberikan.

```
String theInput = JOptionPane.showInputDialog("Please enter a file name:");
```

Setelah memasukkan nama peta yang baru, maka akan diperikasa apakan peta dengan nama tersebut sudah ada atau belum. Jika nama peta tersebut ternyata sudah ada maka akan muncul peringatan "File sudah ada". Bila tidak, maka akan dibuat sebuah file teks berekstensi *.map* sesuai data yang dimasukkan.

```
f.createNewFile();
filename = (theInput+".map");
```

Secara bersamaan dibuat pula file untuk menyimpan data log dari peta tersebut. File teks ini nantinya berekstensi *.log* akan menyimpan data *reply* yang mengalami error tiap node pada peta tersebut.

```
File log = new File(theInput+"-map.log");
filelog = (theInput+"-map.log");
```

4.2.3 Sub Menu Open Map

Saat *user* memilih untuk membuka file peta jaringan yang telah dibuat dengan memilih sub menu open map , maka fungsi main() akan memanggil class JFileChooser(). Secara default class ini akan menunjuk ke folder My document tempat file –file berekstensi *.map* berada. Setelah *user* memilih nama file yang dikehendaki, maka peta jaringan tersebut akan langsung terbuka pada sub menu view map. Perintah selengkapnya dapat dilihat sebagai berikut :

```
JFileChooser fcsr = new JFileChooser();
fcsr.setFileFilter(new Filter());
int returnVal = fcsr.showOpenDialog(frame);
if (returnVal == JFileChooser.APPROVE_OPTION) {
```

```
File f = fcsr.getSelectedFile();
filename = f.getName();
final viewmap view = new viewmap();
view.show();
```

4.2.4 Sub Menu Add Node

Untuk dapat mengakses file *.map* yang berupa teks, diperlukan java.io agar dapat memanfaatkan input/output stream. Pada file ini tanda koma digunakan untuk memisahkan antara satu data dengan yang lainnya. Pada sub menu ini jika user memasukkan data nama node yang telah ada maka akan muncul peringatan "Node Sudah Ada"

```
JOptionPane.showMessageDialog(editframe,"Node Sudah Ada!!","
NewMap",JOptionPane.WARNING_MESSAGE);
```

Dan bila user memasukkan data time out lebih besar dari waktu polling peta tersebut maka akan muncul peringatan "Timeout lebih besar dari waktu polling"

```
JOptionPane.showMessageDialog(editframe,"Timeout lebih besar dari waktu
polling","Edit Map",JOptionPane.WARNING_MESSAGE);
```

Dalam menyimpan data node tersebut, semua data akan disimpan pada file dengan FileOutputStream(). Data yang dimasukkan akan dipecah menjadi 4 baris data sesuai jumlah sevice yang dapat dipantau. Hal ini dilakukan untuk mempermudah melakukan pengeditan data tiap node service. Perintah selengkapnya dapat dilihat sebagai berikut :

```
fout = new FileOutputStream (netview.dir+"\\"+netview.filename);
new PrintStream(fout).print(mergeLine);
if(cicmp==true) {
new PrintStream(fout).println((jml+1) + "," + nodefield.getText() + "," +
comboaddtype + "," + hostfield.getText() + ",ICMP,-," + timeoutfield.getText());
jml = jml+1;
}
if(chttp==true) {
new PrintStream(fout).println((jml+1) + "," + nodefield.getText() + "," +
comboaddtype + "," + hostfield.getText() + ",HTTP," + httpfield.getText()+ "," +
timeoutfield.getText());
jml = jml+1;
}
```

```
if(cftp==true) {
new PrintStream(fout).println((jml+1) + "," + nodefield.getText() + "," +
comboaddtype + "," + hostfield.getText() + ",FTP," + ftpfield.getText()+ "," +
timeoutfield.getText());
jml = jml+1;
}
if(cdns==true) {
new PrintStream(fout).println((jml+1) + "," + nodefield.getText() + "," +
comboaddtype + "," + hostfield.getText() + ",DNS," + dnsfield.getText()+ "," +
timeoutfield.getText());
jml = jml+1;
```

4.2.5 Sub Menu View Map

Saat user memilih sub menu ini maka fungsi main() akan memanggil fungsi viewmap(). Pada fungsi ini frame view map dibuat dengan meletakkan dua buah internal frame didalam frame utama dengan menggunakan perintah :

```
public static JFrame frame = netview.frame;
public static JInternalFrame viewframe;
```

Frame utama (netview) merupakan induk bagi internal frame viewframe, dan viewframe merupakan induk bagi internal frame inviewframe. Internal frame ini beroperasi dalam ruang lingkup JDesktopPane.

```
JDesktopPane desktop = new JDesktopPane();
```

Parameter dalam JInternalFrame digunakan untuk menampilkan judul, tombol resize, tombol close dan tombol maximize pada internal frame.

```
viewframe = new JInternalFrame("View Map",false,true,false,true);
JInternalFrame inviewframe = new JInternalFrame("",false,false,false,false);
```

Pada saat viewframe dipanggil, maka fungsi set() akan tereksekusi. Fungsi set() ini meletakkkan tombol start, stop, reset dan exit pada viewframe. Pada tiap – tiap tombol ini diberikan Action Listener yang bila nantinya tombol ini dieksekusi oleh user maka akan memanggil fungsi actionPerformed(). Saat tombol start dieksekusi, maka akan memanggil fungsi konek(). Fungsi konek() membaca data host, service dan port dari node – node yang terdapat pada peta jaringan dan

memanggil fungsi cek() untuk melakukan proses *request*. Berikut perintah yang digunakan untuk mengirimkan *request* pada service HTTP :

```
try {
URL u = new URL(service.toLowerCase(), host, Integer.parseInt(port), "/");
                    conn = (HttpURLConnection) u.openConnection();
                    conn.setRequestMethod("HEAD");
                    conn.connect();
int s = conn.getResponseCode();
if(s == 200 || s == 401){
          code = "OK";
}else{
          logging();
          code = "NO";
}
}catch( java.io.IOException e) {
          logging();
          code = "NO";
}
```

Pada HTTP, proses *request* dilakukan dengan membuka sebuah koneksi URL terlebih dahulu. Dan metode *request* yang dipakai adalah metode HEAD yaitu untuk mendapatkan informasi tentang dokumen dari server tetapi bukan dokumen itu sendiri. Ini dilakukan karena program ini bukan browser murni, dan input yang diperlukan hanya bagian response code dari *reply* yang diterima. Int s merupakan variabel sementara yang digunakan untuk menyimpan nilai dari response code pada header pesan *reply* yang diterima. Pembacaan response code tersebut dilakukan dengan memanggil kelas getResponseCode().

Pada FTP, koneksi dibuat melalui socket menggunakan alamat host dan port. Untuk variabel port yang bertipe string perlu dikonversi terlebih dahulu menjadi variabel integer dengan menggunakan perintah Integer.parseInt(). Data socket tersebut nantinya akan dirubah menjadi byte – byte biner dengan menggunakan kelas DataOutputStream(). Semua data ini akan disimpan pada buffer. Dan berikut perintah yang digunakan untuk mengirimkan *request* pada service FTP :

```
try{
s = new Socket(host, Integer.parseInt(port));
dos = new DataOutputStream(s.getOutputStream());
buff = new BufferedReader(new InputStreamReader(s.getInputStream()));
dos.writeBytes("anonymous\n");
dos.writeBytes("anonymous@yahoo.com\n");
str=buff.readLine();
if (str.startsWith("220")) {
        code = "OK";
}else {
        logging();
}
```

Sedangkan perintah yang digunakan untuk mengirimkan *request* pada service DNS adalah sebagai berikut:

```
try {
String result = InetAddress.getByName(host).toString();
        code = "OK";
}catch(UnknownHostException u) {
logging();
code = "NO";
}catch(IOException e) {
}
```

Untuk mengirimkan *request* pada service DNS digunakan class getByName(). Dengan class ini dikirimkan alamat host yang nantinya akan di*reply* dengan alamat IP oleh service DNS yang disediakan pada node yang dimonitor.

Internal frame inviewframe digunakan untuk meletakkan node – node yang akan di monitoring. Node - node tersebut merupakan sebuah tombol yang dibuat menggunakan JButton(). Pada setiap node di peta tersebut diberikan Mouse Listener. Class ini digunakan untuk menerima input dari user berupa klik kanan atau kiri dari mouse. Saat user meng-klik kiri pada node tersebut maka program akan mengirimkan *request* pada node tersebut diluar waktu polling menggunakan class actionPerformed(). Sedangkan saat user meng-klik kiri pada node tersebut, program akan mengeksekusi class JPopupMenu(). Class ini menampilkan sub menu properties dan delete. Sub menu properties digunakan untuk mengedit data dari node tersebut.

Prinsip kerjanya hampir sama dengan sub menu add node. Tetapi sebelumnya, pada sub menu ini akan membaca data pada file lalu menampilkannya pada form edit.

```
try {
fin = new FileInputStream(netview.dir+"\\"+netview.filename);
din = new BufferedReader(new InputStreamReader(fin));
String isidata[] = new String[7];
String thisLine;
while ((thisLine = din.readLine()) != null) {
isidata = thisLine.split(",");
if (isidata[1].equals(data[0]) && isidata[4].equals(data[1])) {
        nodefield.setText(isidata[1]);
        combotype.setSelectedItem(isidata[2]);
        hostfield.setText(isidata[3]);
        if(isidata[4].equals("HTTP")) {
        http.setSelected(true);
        httpfield.setEnabled(true);
        httpfield.setText(isidata[5]);
        chttp=true;
}else if(isidata[4].equals("FTP")) {
        ftp.setSelected(true);
        ftpfield.setEnabled(true);
        ftpfield.setText(isidata[5]);
        cftp=true;
}else if(isidata[4].equals("DNS")) {
        dns.setSelected(true);
        dnsfield.setEnabled(true);
        dnsfield.setText(isidata[5]);
        cdns=true;
}else if(isidata[4].equals("ICMP")) {
        icmp.setSelected(true);
        cicmp=true;
}
timeoutfield.setText(isidata[6]);
}
}
}catch(IOException ioe) {}
```

Sub menu ini ada 2, dibuat berbeda untuk mempermudah pengeditan node dan host. Untuk properties pada host, user dapat mengedit semua data dari host tersebut termasuk menambahkan service yang ingin dipantau. Sedangkan pada node service user hanya dapat port dari service tersebut dan mengubah waktu timeout saja.

Bila user melakukan perubahan pada data node tersebut dan menekan tombol edit, maka data baru dari node tersebut akan menimpa data yang lama. Dan untuk

sub menu delete, data node akan diset dengan karakter "-" untuk menandakan node pada host tersebut tidak diaktifkan.

```
try {
fin = new FileInputStream(netview.dir+"\\"+netview.filename);
din = new BufferedReader(new InputStreamReader(fin));
String isidata[] = new String[7];
String thisLine;
int b = 0;
while ((thisLine = din.readLine()) != null) {
isidata = thisLine.split(",");
if (isidata[1].equals(data[0]) && isidata[4].equals(data[1])) {
line = Integer.parseInt(isidata[0])+",-,-,"+isidata[3]+","+isidata[4]+",-,-"+"\n";
        b = 1;
}else{
        if (b == 1){
                line =
                Integer.parseInt(isidata[0])+","+isidata[1]+","+isidata[2]+","+isid
                ata[3]+","+isidata[4]+","+isidata[5]+","+isidata[6]+"\n";
        }else {
                line = thisLine + "\n";
                }
        }
mergeLine = mergeLine + line;
}
fin.close();
fout = new FileOutputStream (netview.dir+"\\"+netview.filename);
new PrintStream(fout).print(mergeLine);
fout.close();
set();
}catch (Exception exc) {
System.err.println("error :" + exc);
}
}
```

Setelah semua komponen dari internal frame viewframe diatur lokasi dan ukurannya oleh fungsi set(), maka internal frame ini dimasukkan pada desktop. Dan semua komponen dari internal frame inviewframe dimasukkan pada desktopin.

```
desktopin.add(inviewframe);
desktop.add(viewframe);
```

4.2.6 Sub Menu Logs

Sama seperti sub menu view map, sub menu ini dibuat menggunakan dua internal frame pada frame utama. Untuk menuliskan daftar logs pada peta jaringan

digunakan JTextArea(). *Scroll bar* pada teks area ini akan muncul pada saat isi dari teks area melebihi ukuran dari internal frame saja.

```
JTextArea logsmap = new JTextArea();
JScrollPane scroll = new JScrollPane(logsmap,
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
```

Saat digunakan user, program akan membaca isi dari file log dari peta jaringan yang dibuka pada saat tersebut dengan menggunakan perulangan dan isi dari file tersebut kemudian ditampilkan pada teks area yang telah disediakan.

```
while ((baca = buff.readLine()) != null) {
        logsmap.append(baca+"\n");
}
```

Pada sub menu ini disediakan tombol operasi pada data logs yaitu tombol print dan clear. Dengan tombol print user dapat mencetak semua data logs peta jaringan tersebut. Class printDialog() dipanggil untuk melakukan proses pencetakan data logs. Dan dengan tombol clear user dapat menghapus semua daftar log yang telah ada. Operasi clear ini dilakukan dengan menghapus file log dari peta jaringan tersebut dan membuat yang baru dengan nama yang sama sesuai dengan nama peta. Perintah selengkapnya dapat dilihat sebagai berikut:

```
clear.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                log.delete();
                logs = new File(filelog);
                try{
                        if(logs.createNewFile()){
                                show();
                        }
                }catch(IOException ioe){}
        }
});
```

4.2.7 Sub Menu Map Properties

Sub menu ini dibuat dengan menggunakan Jframe(). Saat user tidak memasukkan waktu polling dan menekan tombol set maka akan tampil pesan peringatan menggunakan JoptionPane.

```
JOptionPane.showMessageDialog(frame,"Waktu polling belum diisi","Map
Properties",JOptionPane.WARNING_MESSAGE);
```

Saat dieksekusi, program akan membaca file *conf.ini* yang telah dibuat pada awal program ini dijalankan. File ini akan dibuat pada folder yang sama dengan tempat program dijalankan.

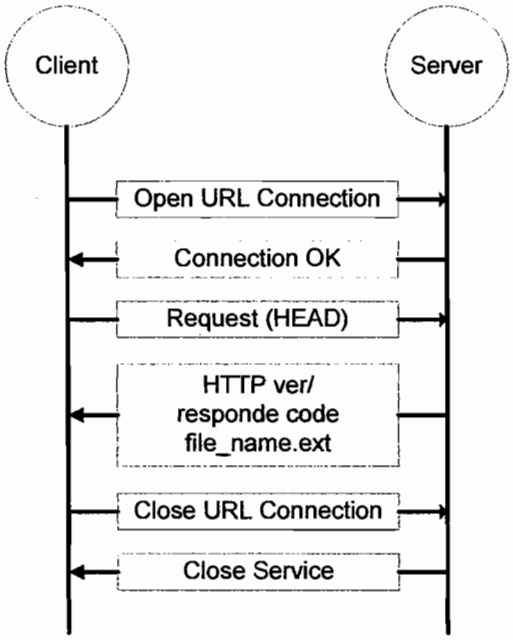
```
try {
FileInputStream f = new FileInputStream("conf.ini");
BufferedReader br = new BufferedReader(new InputStreamReader(f));
temp = br.readLine();
}catch (IOException ioe) {}
```

Saat user meng-klik tombol set pada frame ini, maka data waktu polling yang baru akan ditulis pada file *conf.ini* untuk menggantikan data yang lama. Untuk tugas ini digunakan FileWriter(), dikarenakan penulisan data ini ditentukan pada satu file saja yaitu *conf.ini.*

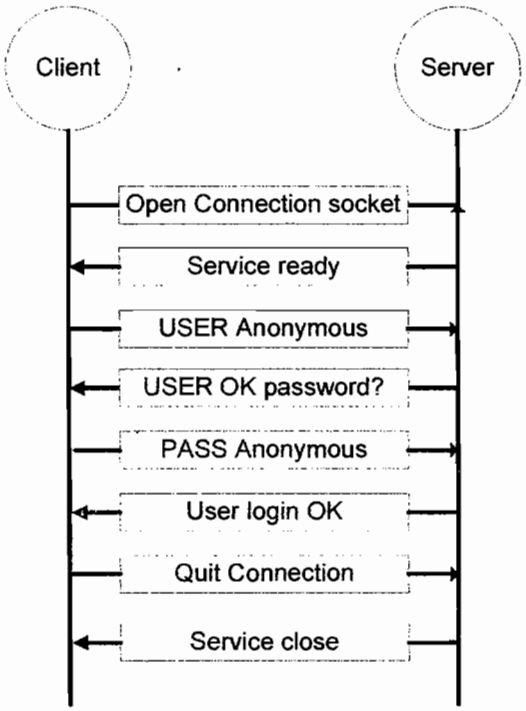
```
FileWriter fw = new FileWriter("conf.ini");
BufferedWriter r = new BufferedWriter(fw);
r.write(pollingfield.getText());
```

4.3 Proses Koneksi

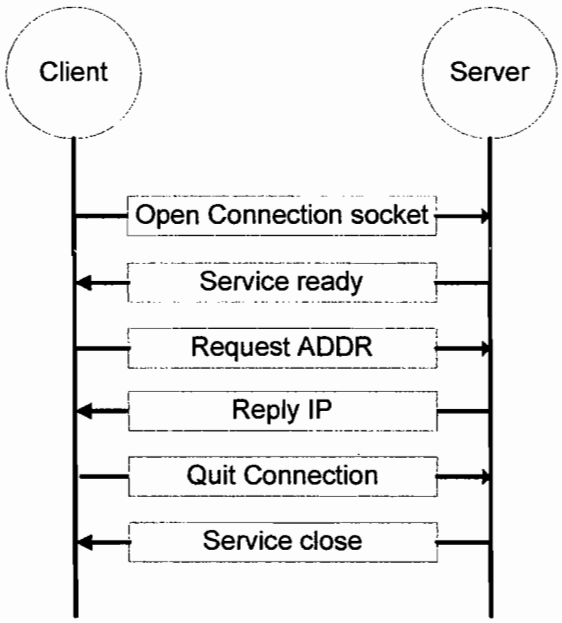
4.3.1 Proses Koneksi HTTP



4.3.2 Proses Koneksi FTP



4.3.3 Proses Koneksi DNS



BAB V

ANALISA SISTEM

5.1 Analisa Program

Proses monitoring pada program ini dilakukan pada dua lapisan TCP/IP yaitu pada lapisan network dan lapisan aplikasi. Pada lapisan network service yang dimonitoring adalah ICMP dan pada lapisan aplikasi, service yang dimonitoring adalah service HTTP, FTP dan DNS. Proses *request reply* pada masing-masing service dilakukan dengan metode yang berbeda.

Pada ICMP :

- *Request* yang dilakukan dengan perintah *ping.*
- Bentuk pesan yang dikirim berupa echo.
- Bentuk *reply* yang diterima berupa echo *reply.*

Pada HTTP :

- Metode yang digunakan saat melakukan *request* adalah metode HEAD
- *Request* yang dikirimkan hanya meminta keterangan tentang data yang diinginkan, bukan data itu sendiri.
- *Request* dikirimkan melalui port default dari service tersebut atau yang telah diketahui sebelumnya.
- *Reply* yang diterima berupa *response code* dan *status.*

Pada FTP :

- Pada service FTP, program melakukan *request* dengan cara login pada host tersebut.

- *Request* dikirimkan melalui port default dari service tersebut atau yang telah diketahui sebelumnya.
- Pada header *reply* yang diterima dibaca *response code* dan *status* dari service.

Pada DNS :

- *Request* yang dikirimkan program pada service DNS berupa alamat host
- *Request* dikirimkan melalui port default dari service tersebut atau yang telah diketahui sebelumnya.
- *Reply* yang diterima berupa alamat IP node yang dimonitoring.

Dalam program ini tidak menggunakan tabel sebagai tempat penyimpanan data dikarenakan pertimbangan data pada program ini cukup sederhana dan cukup disimpan dalam bentuk file teks.

5.2 Analisa Bahasa Pemrograman

Bahasa pemrograman Java dapat digunakan untuk membangun program aplikasi monitoring jaringan karena :

1. Dalam Java telah disediakan kelas – kelas yang mendukung untuk *socket programming*. Java juga menyediakan kelas untuk mendukung tipe koneksi jaringan yang bersifat *connectienless* maupun *connection oriented.*
2. Java juga memiliki kemampuan untuk menangani dan menjalankan banyak thread sekaligus (multi threading). Dalam program ini penulis menggunakan multi threading untuk mengirimkan berbagai *request* pada node yang dimonitoring.

3. Java mempunyai package untuk mengaplikasikan GUI (graphical user interface) pada program ini. GUI digunakan dalam program ini untuk mempermudah interaksi antara user dengan program.
4. Untuk aplikasi dalam sistem operasi Linux, digunakan JRE untuk Linux. Perintah untuk eksekusi progam ini adalah ***jre –jar path/nama_file.jar.*** File hasil dari program ini akan disimpan dalam *root.*

5.3 Kelebihan dan Kekurangan Program

5.3.1 Kelebihan

Kelebihan program ini antara lain :

- *User* dapat melakukan percepatan proses *request* pada salah satu node pada peta jaringan tanpa harus menunggu waktu polling.
- Terdapat lima tingkat kegagalan dalam reply yang disajikan dalam bentuk perubahan warna pada node service. Sehingga user dapat mengetahui kondisi sebenarnya dari host yang dituju.
- User interface yang digunakan oleh program ini sudah berbentuk grafis.
- *Event handler* untuk input *user* yang tidak tepat sudah terpenuhi.
- Program ini dapat dijalankan dalam berbagai sistem operasi (*Multiplatform*).

5.3.2 Kekurangan

Kekurangan program ini antara lain :

- Pilihan service yang ditawarkan oleh program ini untuk dapat dimonitoring terbatas hanya pada ICMP, HTTP, FTP dan DNS. Pemilihan keempat service ini didasari pertimbangan besarnya frekuensi penggunaan service – service tersebut dalam jaringan internet.

- Kurangnya tool untuk mengambarkan keadaan topologi jaringan secara riil.
- Jumlah host yang dapat dipantau dibatasi.

BAB VI
KESIMPULAN

Bab ini merupakan bagian akhir dari penulisan skripsi ini. Pada bab ini akan dicantumkan beberapa kesimpulan dan saran dari hal-hal yang terkait dengan proses *request reply* dan monitoring jaringan.

6.1. Kesimpulan

Dari implementasi proses *request reply* dalam program monitoring jaringan maka dapat disimpulkan:

1. Monitoring jaringan dengan pada level aplikasi dilakukan berdasarkan karakteristik dari service tersebut.
2. Pengiriman pesan *request* pada level aplikasi cenderung lebih tepat dibandingkan pada level network. Pada level network ada kemungkinan service ICMP ditutup oleh administrator jaringan atau *firewall* sehingga pesan *ping* tidak mendapat *reply* walaupun service lain pada host tersebut dalam keadaan aktif.

6.2. Saran

Saran yang dapat diberikan penulis dalam pengembangan program monitoring jaringan lebih lanjut:

1. Program ini hanya mampu melakukan monitoring pada beberapa service lapisan aplikasi TCP/IP saja, untuk selanjutnya diharapkan program ini dapat melakukan monitoring dengan mengunakan proses *request reply* pada semua service yang tersedia pada lapisan aplikasi TCP/IP.

2. Tampilan grafis yang diberikan kepada user masih bisa dioptimalkan. Misalkan dengan memberikan tool – tool untuk mengambarkan struktur jaringan agar *user* dapat membayangkan keadaan peta jaringan sesungguhnya.
3. Jumlah *host* yang dapat dimonitoring pada satu peta agar tidak dibatasi. Hal ini dapat dilakukan agar user tidak memerlukan banyak peta untuk memonitoring suatu jaringan.
4. Program ini agar dapat diaplikasikan dalam pemrograman J2ME, karena berasal dari platform yang sama, beberapa bagian dari program ini dapat digunakan untuk aplikasi J2ME (thread dan koneksi service). Untuk pengembangan selanjutnya, peneliti perlu menyesuaikan fungsi main dan user interface.

DAFTAR PUSTAKA

Stallings, William, "*Komunikasi Data dan Komputer Jaringan Komputer*", Salemba Teknika, 2000.

Susanto, Budi, "*Pemrograman Client/Server dengan Java 2*", Elek Media Komputindo, 2003

Sanjaya, Ridwan, "*Membuat Aplikasi Windows Multiplatform dengan Java GUI*", Elek Media Komputindo, 2003

Indrajani, S.Kom, MM dan Martin, S.Kom, "*Pemrograman Berorientasi Objek Dengan Java*", Elek Media Komputindo, 2004

http://www.javaalmanac.com

http://www.java.sun.com

http://www.ftpplanet.com

http://www.simpledns.com

http://www.jasakom.com

http://www.yogyafree.tk

http://www.crack.am

LAMPIRAN

Listing Program

Netview.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.util.Properties;

public class netview {
	public static String IP, teks, tempport, dir, filelog, lmn;
	public static String filename = "";
	public static int port, poll;
	public static String[] LOG = {"","","","","","",""};
	public final static int ONE_SECOND = 1;
	public static JProgressBar pb;
	public static Timer timer;
	public static LongTask task;
	public static JFileChooser choose;
	public static File temporary;
	public static File log;
	public static int quit = 0;
	public static konek con;
	public static JFrame frame;

	public static int[] kode2 = new int[28];
	public static final JMenuItem mnunewmap = new JMenuItem("New map",'N');
	public static final JMenuItem mnuopen = new JMenuItem("Open",'O');
	public static final JMenuItem mnusave = new JMenuItem("Save",'S');
	public static final JMenuItem mnueditmap = new JMenuItem("Add Node");
	public static final JMenuItem mnuviewmap = new JMenuItem("View Map");
	public static final JMenuItem mnulogs = new JMenuItem("Logs");
	public static final JMenuItem mnusetmap = new JMenuItem("Map Properties");
	public static JMenuItem mnufeat = new JMenuItem("Feature");


	public static void main(String[] args) {
		for(int i=0;i<kode2.length;i++){
			kode2[i] = 0;
		}
		try{
			File f = new File("conf.ini");
			if (!f.exists()) {
				f.createNewFile();
				FileWriter fw = new FileWriter("conf.ini");
				BufferedWriter w = new BufferedWriter(fw);
				String str = "60";
				w.write(str);
				w.close();
				fw.close();
			}
```

```
}catch(IOException ioe) {
}
try{
        choose = new JFileChooser();
        temporary = choose.getCurrentDirectory();
        dir = temporary.getCanonicalPath();
        FileInputStream f = new FileInputStream("conf.ini");
        BufferedReader br = new BufferedReader(new InputStreamReader(f));
        poll = Integer.parseInt(br.readLine());
}catch(IOException ioe){}

frame = new JFrame();
JMenuBar menubar = new JMenuBar();
        frame.setJMenuBar(menubar);
JMenu file = new JMenu("File");
        file.setMnemonic('F');
JMenu edit = new JMenu("Edit");
        edit.setMnemonic('E');
JMenu view = new JMenu("View");
        view.setMnemonic('V');
JMenu help = new JMenu("Help");
        help.setMnemonic('H');
JMenuItem exit = new JMenuItem("Exit",'X');
JMenuItem about = new JMenuItem("About");

TextArea area = new TextArea();
Font Arl = new Font("Arial",0,14);

exit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
                { System.exit(0);}
});

menubar.add(file);
menubar.add(edit);
menubar.add(view);
menubar.add(help);
file.add(mnunewmap);
file.add(mnuopen);
file.add(mnusave);
file.add(exit);
edit.add(mnueditmap);
view.add(mnuviewmap);
view.add(mnulogs);
view.add(mnusetmap);
help.add(mnufeat);
help.add(about);
menubar.setSize(400,100);
menubar.setLocation(60,60);
menubar.setVisible(true);
Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
int w = frame.getSize().width;
int h = frame.getSize().height;
int x = (dim.width-w)/2;
int y = (dim.height-h)/2;
```

```
frame.setLocation(x-325,y-205);
frame.setSize(650,410);
ImageDisplayer img = new ImageDisplayer();


mnusave.setEnabled(false);
mnueditmap.setEnabled(false);
mnuviewmap.setEnabled(false);
mnulogs.setEnabled(false);

//FUNGSI SAVE
mnusave.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
        task = new LongTask();
        final JFrame saving = new JFrame("Saving data");
        pb = new JProgressBar(0, task.getLengthOfTask());
        pb.setValue(0);
        pb.setStringPainted(true);
        timer = new Timer(ONE_SECOND, new ActionListener() {
public void actionPerformed(ActionEvent evt) {
        pb.setValue(task.getCurrent());
        if (task.done()) {
        Toolkit.getDefaultToolkit().beep();
                timer.stop();
                saving.dispose();
        }
        }
});
        System.gc();
        task.go();
        timer.start();
saving.getContentPane().setLayout(new FlowLayout());
saving.getContentPane().add(pb);
saving.setVisible(true);
Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
int w = frame.getSize().width;
int h = frame.getSize().height;
int x = (dim.width-w)/2;
int y = (dim.height-h)/2;
saving.setLocation(x+180,y+150);
saving.setSize(200,70);
saving.setVisible(true);
}
});     //tutup SAVE


//FUNGSI NEW MAP
mnunewmap.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
        System.gc();
        for(int i=0;i<28;i++){
                kode2[i] = 0;
        }
String theInput = JOptionPane.showInputDialog("Please enter a file name:");
                if (theInput.equals("")) {
```

```
JOptionPane.showMessageDialog(null, "Nama file tidak valid", "Netview",
JOptionPane.INFORMATION_MESSAGE);
}else{
try{
File f = new File(dir+"\\"+theInput+".map");
if(!f.exists()){
        f.createNewFile();
        filename = (theInput+".map");
        JOptionPane.showMessageDialog(null, "File "+filename+" berhasil
        dibuat", "Netview", JOptionPane.INFORMATION_MESSAGE);
        log = new File(theInput+"-map.log");
        filelog = (theInput+"-map.log");
if (!log.exists()) {
        log.createNewFile();
}
final viewmap view = new viewmap();
        System.gc();
        view.show();
}else {
        JOptionPane.showMessageDialog(null, "File sudah ada", "Netview",
        JOptionPane.INFORMATION_MESSAGE);
}
}catch(IOException ioe){
JOptionPane.showMessageDialog(null, "File sudah ada", "Netview",
JOptionPane.INFORMATION_MESSAGE);
}
mnusave.setEnabled(true);
mnueditmap.setEnabled(true);
mnuviewmap.setEnabled(true);
mnulogs.setEnabled(true);
                }
        }
});     //tutup NEW MAP


//FUNGSI OPEN
mnuopen.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
System.gc();
for(int i=0;i<28;i++){
kode2[i] = 0;
}
JFileChooser fcsr = new JFileChooser();
fcsr.setFileFilter(new Filter());
int returnVal = fcsr.showOpenDialog(frame);
if (returnVal == JFileChooser.APPROVE_OPTION) {
        File f = fcsr.getSelectedFile();
        filename = f.getName();
        final viewmap view = new viewmap();
        view.show();
        mnusave.setEnabled(true);
        mnueditmap.setEnabled(true);
        mnuviewmap.setEnabled(true);
        mnulogs.setEnabled(true);
        }
```

```
                try{
                        log = new File(filename.replace('.','-')+".log");
                        filelog = (filename.replace('.','-')+".log");
                        if (!log.exists()) {
                        log.createNewFile();
                        }
                }catch(IOException ioe) {}
                }
                });     //tutup OPEN


//FUNGSI FILE EDIT MAP
                mnueditmap.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        System.gc();
                        final editmap edit = new editmap();
                        edit.show();
                        mnueditmap.setEnabled(false);
                }
                });     //tutup EDIT MAP


//FUNGSI VIEW MAP
                mnuviewmap.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        System.gc();
                        final viewmap view = new viewmap();
                        view.show();
                }
                });     //tutup VIEW MAP


//FUNGSI LOGS
                mnulogs.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        System.gc();
                        final logs viewlog = new logs();
                        viewlog.show(log,filelog);
                }
                });     //tutup LOGS


//FUNGSI SET MAP
                mnusetmap.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        mnusetmap.setEnabled(false);
                        show();
                }
                });


//FUNGSI FEATURE
                mnufeat.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        System.gc();
```

```
                final feat featframe = new feat();
                featframe.show();
                mnufeat.setEnabled(false);
            }
            });

//FUNGSI ABOUT
            about.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
            JOptionPane.showMessageDialog(null,"Program ini dibuat oleh Saya" + "\n" +
            "Email – street2sleepers@yahoo.com","NetView",
            JOptionPane.INFORMATION_MESSAGE);
            }
            });

            JDesktopPane desktop = new JDesktopPane();
            desktop.setBackground(Color.white);
            frame.setTitle("Netview");
            frame.setResizable(false);
            frame.setJMenuBar(menubar);
            frame.getContentPane().setLayout(new FlowLayout());
            frame.setContentPane(desktop);
            frame.setVisible(true);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }       //tutup main

        public static void show(){
            final JFrame setframe = new JFrame("Map Properties");
            JLabel pollinglabel = new JLabel("Polling");
            final JTextField pollingfield = new JTextField(4);
            JLabel libil = new JLabel("detik");
            JButton set = new JButton("SET");
            JButton keluar = new JButton("EXIT");
            keluar.setMnemonic('X');
            String temp = "";
            try {
                FileInputStream f = new FileInputStream("conf.ini");
                BufferedReader br = new BufferedReader(new InputStreamReader(f));
                temp = br.readLine();
            }catch (IOException ioe) {
            }
            pollingfield.setText(temp);
            set.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            try{
                if (pollingfield.getText().trim().equals("")){
                JOptionPane.showMessageDialog(setframe,"Waktu polling belum
                diisi","Map Properties",JOptionPane.WARNING_MESSAGE);
                }else{
                FileWriter fw = new FileWriter("conf.ini");
                BufferedWriter r = new BufferedWriter(fw);
                r.write(pollingfield.getText());
                poll = Integer.parseInt(pollingfield.getText());
                    setframe.dispose();
                    r.close();
```

```
                                        fw.close();
                                }
                        }catch(IOException ioe) {
                        }
                        }
                });

                keluar.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e){
                if (pollingfield.getText().trim().equals("")){
                JOptionPane.showMessageDialog(setframe,"Waktu polling belum diisi","Map
                Properties",JOptionPane.WARNING_MESSAGE);
                }else{
                        setframe.dispose();
                        mnusetmap.setEnabled(true);
                }
                }
                });

                JPanel p1 = new JPanel();
                JPanel p2 = new JPanel();
                p1.setLayout(new FlowLayout());
                p2.setLayout(new FlowLayout());
                p1.add(pollingiabel);
                p1.add(pollingfield);
                p1.add(libil);
                p2.add(set);
                p2.add(keluar);
                Container konten;
                konten = setframe.getContentPane();
                konten.setLayout(new BorderLayout());
                konten.add(p1,"North");
                konten.add(p2,"South");

                Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
                int w = setframe.getSize().width;
                int h = setframe.getSize().height;
                int x1 = (dim.width-w)/2;
                int y1 = (dim.height-h)/2;
                setframe.setLocation(x1-95, y1-50);
                setframe.setSize(200,100);
                setframe.setVisible(true);
        }

}       //tutup kelas
```

ImageDisplayer.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Timer;
import java.util.TimerTask;
```

```
public class ImageDisplayer {
	static String imageFile = "Logousd.jpg";
	static JWindow f;

	public static void main(String[] args) {
		ImageDisplayer apa = new ImageDisplayer();
	}

	public ImageDisplayer() {
		Image image = Toolkit.getDefaultToolkit().getImage(ImageDisplayer.imageFile);
		ImagePanel imagePanel = new ImagePanel(image);
		f = new JWindow(netview.frame);
		f.addWindowListener(new WindowAdapter() {
		public void windowClosing(WindowEvent e) {
			f.dispose();
		}
		});

		f.getContentPane().add(imagePanel, BorderLayout.CENTER);
		f.setSize(new Dimension(320,240));
		Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
		int w = f.getSize().width;
		int h = f.getSize().height;
		int x = (dim.width-w)/2;
		int y = (dim.height-h)/2;
		f.setLocation(x,y);
		f.setVisible(true);
	}
}

class ImagePanel extends JPanel implements Runnable{
	Image img;
	private Timer timer = new Timer();
	private boolean running = true;
	private int numX,numY,startX=-1,startY=0;
	private int cellX=30,cellY=30;
	private int selang = 1;
	private int awalX,awalY,imgHeight,imgWidth;
	private boolean[][] show;

	public void run() {
		try {
			while (running) {
				Thread.sleep(100);
				updateSel();
				repaint();
			}
		} catch (InterruptedException e) {}
	}

	protected void updateSel(){
		if(startY < numY){
			if(selang>0){
				if(startX < numX-1){
					startX++;
```

```
                                        if(show[startX][startY]){
                                                show[startX][startY] = false;
                                        }
                                }else{
                                        startY++;
                                        startX = numX;
                                        selang = -1;
                                };
                        }else{
                                if(startX > 0){
                                        startX += selang;
                                        if(show[startX][startY]){
                                                show[startX][startY] = false;
                                        }
                                }else{
                                        startY++;
                                        startX = -1;
                                        selang = 1;
                                };
                        }
                }else{
                // Hentikan thread
                        running = false;
                        timer.schedule( new CountDown(), 500 );
                }
        }

        public ImagePanel(Image img) {
                this.img = img;
                imgHeight = 270;
                imgWidth = 330;
                awalX = 0;
                awalY = 0;
                numX = imgWidth / cellX;
                numY = imgHeight / cellY;
                show = new boolean[numX][numY];
                for(int i=0;i<numX;i++){
                        for(int j=0;j<numY;j++){
                                show[i][j] = true;
                        }
                }
                new Thread(this).start();
        }

        private void dismiss(){
                timer.cancel();
                ImageDisplayer.f.dispose();
        }

        private class CountDown extends TimerTask {
                public void run(){
                        dismiss();
                }
        }
```

```
        public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.white);
        g.fillRect(0,0,330,270);
        g.drawImage(img, 0, 0, this);
        for(int i=0;i<numX;i++){
                for(int j=0;j<numY;j++){
                        if(show[i][j]){
                                g.drawRect(awalX + i*cellX,awalY + j*cellY,cellX,cellY);
                                g.fillRect(awalX + i*cellX,awalY + j*cellY,cellX,cellY);
                                }
                        }
                }
        }
}
```

SwingWorker.java

```
import javax.swing.SwingUtilities;

        public abstract class SwingWorker {
                private Object value;
                private Thread thread;

                private static class ThreadVar {
                   private Thread thread;
                   ThreadVar(Thread t) { thread = t; }
                   synchronized Thread get() { return thread; }
                   synchronized void clear() { thread = null; }
                }

                private ThreadVar threadVar;

                protected synchronized Object getValue() {
                   return value;
                }

                private synchronized void setValue(Object x) {
                   value = x;
                }

                public abstract Object construct();

                public void finished() {
                }

                public void interrupt() {
                   Thread t = threadVar.get();
                   if (t != null) {
                      t.interrupt();
                   }
                   threadVar.clear();
                }

                public Object get() {
                   while (true) {
```

```
            Thread t = threadVar.get();
            if (t == null) {
                return getValue();
            }
            try {
                t.join();
            }
            catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                return null;
            }
        }
    }

    public SwingWorker() {
        final Runnable doFinished = new Runnable() {
          public void run() { finished(); }
        };

        Runnable doConstruct = new Runnable() {
            public void run() {
                try {
                    setValue(construct());
                }
                finally {
                    threadVar.clear();
                }

                SwingUtilities.invokeLater(doFinished);
            }
        };

        Thread t = new Thread(doConstruct);
        threadVar = new ThreadVar(t);
    }

    public void start() {
        Thread t = threadVar.get();
        if (t != null) {
            t.start();
        }
    }
}
```

Editmap.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class editmap {
        boolean chttp;
        boolean cftp;
        boolean cdns;
```

```
boolean cicmp;
JCheckBox icmp;
JCheckBox ftp;
JCheckBox http;
JCheckBox dns;
public static JFrame editframe;
public static JFrame frame = netview.frame;
public void show() {
int x = 0;
int close = 0;
chttp = false;
cftp = false;
cdns = false;
cicmp = false;
try{
        FileInputStream fin;
        BufferedReader din;
        fin = new FileInputStream(netview.dir+"\\"+netview.filename);
        din = new BufferedReader(new InputStreamReader(fin));
        String isidata[] = new String[7];
        String thisLine;
        int j = 0;
        while ((thisLine = din.readLine()) != null) {
                j++;
                if(j==28){
                JOptionPane.showMessageDialog(frame,"Map sudah penuh","Edit
                Map",JOptionPane.WARNING_MESSAGE);
                x = 1;
                break;
                }else{ x = 0; }
                }
        }catch(IOException ioe){}

        if(x == 0){
                editframe = new JFrame("Add Node");
                JLabel node = new JLabel("Nama Node");
                final JTextField nodefield = new JTextField(10);
                JLabel type = new JLabel("Type");
                String[] typenode = {"","Host","Router","Server"};
                final JComboBox combotype = new JComboBox(typenode);
                JLabel host = new JLabel("Host  ");
                final JTextField hostfield = new JTextField(10);
                JLabel service = new JLabel("Service :");
                icmp = new JCheckBox("ICMP");
                ftp = new JCheckBox("FTP");
                final JTextField  ftpfield= new JTextField(4);
                dns = new JCheckBox("DNS");
                final JTextField  dnsfield= new JTextField(4);
                http = new JCheckBox("HTTP");
                final JTextField  httpfield= new JTextField(4);
                JLabel timeout = new JLabel("Time Out");
                final JTextField  timeoutfield= new JTextField(2);
                timeoutfield.setText("");
                JLabel detik = new JLabel("detik                ");
                JButton tambah = new JButton("ADD");
```

```
                JButton keluar = new JButton("EXIT");
                keluar.setMnemonic('X');

editframe.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);

ItemListener klik = new ItemListener() {
public void itemStateChanged(ItemEvent e) {
        Object sors = e.getItemSelectable();
        if (sors==ftp) {
        ftpfield.setText("21");
        cftp=true;
        }
        if (sors==http) {
        httpfield.setText("80");
        chttp=true;
        }
        if (sors==dns) {
        dnsfield.setText("53");
        cdns=true;
        }
        if (sors==icmp) {
        cicmp=true;
        }
        if (e.getStateChange()==ItemEvent.DESELECTED) {
        if(sors==ftp) {
        ftpfield.setText("");
        cftp=false;
        }
        if(sors==http) {
        httpfield.setText("");
        chttp=false;
        }
        if(sors==icmp) {
        cicmp=false;
        }
        if(sors==dns) {
        dnsfield.setText("");
        cdns=false;
        }
        }
        }
        };

        http.addItemListener(klik);
        ftp.addItemListener(klik);
        icmp.addItemListener(klik);
        dns.addItemListener(klik);

//tombol tambah EDIT MAP
tambah.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
        FileInputStream fin;
        BufferedReader din;
        FileOutputStream fout;
                String comboaddtype = (String)combotype.getSelectedItem();
```

```
String mergeLine = "";
int i = 0;
int jml = 0;
int to;
int hf=1;
int ff=1;
int df=1;
try{
        to = Integer.parseInt(timeoutfield.getText());
}catch(NumberFormatException num) {
to = 0;
}
if (chttp==true) {
try{
        hf = Integer.parseInt(httpfield.getText());
}catch(NumberFormatException num) { hf = 0; }
                }
if (cftp==true) {
try{
        ff = Integer.parseInt(ftpfield.getText());
}catch(NumberFormatException num) { ff = 0; }
}
if (cdns==true) {
try{
        df = Integer.parseInt(dnsfield.getText());
}catch(NumberFormatException num) { df = 0; }
}
if (nodefield.getText().trim().equals("") ||
hostfield.getText().trim().equals("") || timeoutfield.getText().equals("")){
JOptionPane.showMessageDialog(editframe,"Data belum Lengkap","Edit
Map",JOptionPane.WARNING_MESSAGE);
}else if(chttp==true && httpfield.getText().equals("")){
JOptionPane.showMessageDialog(editframe,"Port HTTP tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
}else if(cftp==true && ftpfield.getText().equals("")){
JOptionPane.showMessageDialog(editframe,"Port FTP tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
}else if(cdns==true && dnsfield.getText().equals("")){
JOptionPane.showMessageDialog(editframe,"Port DNS tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
}else if (to == 0) {
JOptionPane.showMessageDialog(editframe,"Timeout tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
timeoutfield.setText("");
}else if (hf == 0) {
JOptionPane.showMessageDialog(editframe,"Port HTTP tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
httpfield.setText("");
}else if (ff == 0) {
JOptionPane.showMessageDialog(editframe,"Port FTP tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
ftpfield.setText("");
}else if (df == 0) {
JOptionPane.showMessageDialog(editframe,"Port DNS tidak valid","Edit
Map",JOptionPane.WARNING_MESSAGE);
```

```
dnsfield.setText("");
}else if (to > netview.poll) {
JOptionPane.showMessageDialog(editframe,"Timeout lebih besar dari
waktu polling","Edit Map",JOptionPane.WARNING_MESSAGE);
timeoutfield.setText("");
}else{
try {
fin = new FileInputStream(netview.dir+"\\"+netview.filename);
din = new BufferedReader(new InputStreamReader(fin));
String isidata[] = new String[7];
String thisLine;
int j = 0;
int full=0;
while ((thisLine = din.readLine()) != null) {
        j++;
        jml = j;
        isidata = thisLine.split(",");
if(isidata[3].equals(hostfield.getText())){
        full=2;
}else if(isidata[1].equals(nodefield.getText())){
        full = 3;
}else if(j==25){
        full = 1;
}else full = 0;
        mergeLine += thisLine + "\n";
}
if(full==1){
JOptionPane.showMessageDialog(editframe,"File sudah penuh","New
Map",JOptionPane.WARNING_MESSAGE);
editframe.dispose();
}else if(full==2){
JOptionPane.showMessageDialog(editframe,"Host sudah ada","New
Map",JOptionPane.WARNING_MESSAGE);
}else if(full==3){
JOptionPane.showMessageDialog(editframe,"Node sudah ada","New
Map",JOptionPane.WARNING_MESSAGE);
}else if(full==0){
fout = new FileOutputStream (netview.dir+"\\"+netview.filename);
new PrintStream(fout).print(mergeLine);
if(cicmp==true) {
        new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
        + "," + comboaddtype + "," + hostfield.getText() + ",ICMP,-," +
        timeoutfield.getText());
        jml = jml+1;
}else{
        new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
        + "," + comboaddtype + "," + hostfield.getText() + ",ICMP,-,-");
        jml = jml+1;
        }
if(chttp==true) {
        new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
        + "," + comboaddtype + "," + hostfield.getText() + ",HTTP," +
        httpfield.getText()+ "," + timeoutfield.getText());
        jml = jml+1;
}else{
```

```
            new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
            + "," + comboaddtype + "," + hostfield.getText() + ",HTTP,-,-");
            jml = jml+1;
}
if(cftp==true) {
            new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
            + "," + comboaddtype + "," + hostfield.getText() + ",FTP," +
            ftpfield.getText()+ "," + timeoutfield.getText());
            jml = jml+1;
}else{
            new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
            + "," + comboaddtype + "," + hostfield.getText() + ",FTP,-,-");
            jml = jml+1;
            }
if(cdns==true) {
            new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
            + "," + comboaddtype + "," + hostfield.getText() + ",DNS," +
            dnsfield.getText()+ "," + timeoutfield.getText());
            jml = jml+1;
}else{
            new PrintStream(fout).println((jml+1) + "," + nodefield.getText()
            + "," + comboaddtype + "," + hostfield.getText() + ",DNS,-,-");
            jml = jml+1;
}
fout.close();
}
            fin.close();
            if(jml==28) {
            JOptionPane.showMessageDialog(editframe,"File sudah
            penuh\n"+nodefield.getText()+" adalah node terakhir.",
            "New Map",JOptionPane.WARNING_MESSAGE);
            editframe.dispose();
            final viewmap view = new viewmap();
            view.show();
}else {
            nodefield.setText("");
            combotype.setSelectedItem("");
            hostfield.setText("");
            timeoutfield.setText("");
            http.setSelected(false);
            httpfield.setText("");
            chttp=false;
            ftp.setSelected(false);
            ftpfield.setText("");
            cftp=false;
            icmp.setSelected(false);
            cicmp=false;
            dns.setSelected(false);
            dnsfield.setText("");
            cdns=false;
            }
}catch (Exception exc) {
System.err.println("error :" + exc);
}
}
```

```
        }
        });

//tombol keluar EDIT MAP
        keluar.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        editframe.dispose();
                        final viewmap view = new viewmap();
                        netview.mnueditmap.setEnabled(true);
                        view.show();
                        }
        });

        editframe.getContentPane().setLayout(new FlowLayout());
        editframe.getContentPane().add(node);
        editframe.getContentPane().add(nodefield);
        editframe.getContentPane().add(type);
        editframe.getContentPane().add(combotype);
        editframe.getContentPane().add(host);
        editframe.getContentPane().add(hostfield);
        editframe.getContentPane().add(http);
        editframe.getContentPane().add(httpfield);
        editframe.getContentPane().add(ftp);
        editframe.getContentPane().add(ftpfield);
        editframe.getContentPane().add(icmp);
        editframe.getContentPane().add(dns);
        editframe.getContentPane().add(dnsfield);
        editframe.getContentPane().add(service);
        editframe.getContentPane().add(timeout);
        editframe.getContentPane().add(detik);
        editframe.getContentPane().add(timeoutfield);
        editframe.getContentPane().add(tambah);
        editframe.getContentPane().add(keluar);
        Container konten;
        konten = editframe.getContentPane();
        konten.setLayout(new GridBagLayout());
        GridBagConstraints pos = new GridBagConstraints();
        pos.anchor = GridBagConstraints.WEST;
        pos.gridx = 7;
        pos.gridy = 0;
        konten.add(node,pos);
        pos.gridx++;
        konten.add(nodefield,pos);
        pos.gridy++;
        pos.gridx = 7;
        konten.add(type,pos);
        pos.gridx++;
        konten.add(combotype,pos);
        pos.gridy++;
        pos.gridx = 7;
        konten.add(host,pos);
        pos.gridx++;
        konten.add(hostfield,pos);
        pos.gridy++;
        pos.gridx = 7;
```

```
                konten.add(service,pos);
                pos.gridy++;
                pos.gridx = 7;
                konten.add(icmp,pos);
                pos.gridy++;
                pos.gridx = 7;
                konten.add(http,pos);
                pos.gridx++;
                konten.add(httpfield,pos);
                pos.gridy++;
                pos.gridx = 7;
                konten.add(ftp,pos);
                pos.gridx++;
                konten.add(ftpfield,pos);
                pos.gridy++;
                pos.gridx = 7;
                konten.add(dns,pos);
                pos.gridx++;
                konten.add(dnsfield,pos);
                pos.gridy++;
                pos.gridx = 7;
                konten.add(timeout,pos);
                pos.gridx++;
                konten.add(timeoutfield,pos);
                pos.anchor = GridBagConstraints.EAST;
                pos.gridx = 8;
                konten.add(detik,pos);
                pos.anchor = GridBagConstraints.WEST;
                pos.gridy++;
                pos.gridx = 7;
                konten.add(tambah,pos);
                pos.gridx++;
                konten.add(keluar,pos);


                editframe.setVisible(true);

                Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
                int w = editframe.getSize().width;
                int h = editframe.getSize().height;
                int x1 = (dim.width-w)/2;
                int y1 = (dim.height-h)/2;

                editframe.setResizable(false);
                editframe.setLocation(x1-265,y1-191);
                editframe.setSize(650,410);
                editframe.setVisible(true);

            }
        }
}
```

Logs.java

```
import java.awt.*;
import java.awt.event.*;
```

```
import java.io.*;
import javax.swing.*;
import java.awt.print.*;

public class logs {
	public static JFrame frame = netview.frame;
	public static File log;
	public static File logg;
	public static String filelog;
	public static PrinterJob printme = PrinterJob.getPrinterJob();

	public void show(File Log,String Filelog) {
		log = Log;
		filelog = Filelog;
		JDesktopPane desktop = new JDesktopPane();
		final JInternalFrame logsframe = new JInternalFrame("Logs
		List",false,true,false,true);
		JTextArea logsmap = new JTextArea();
		JScrollPane scroll = new JScrollPane(logsmap,
		JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
		JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
		JButton clear = new JButton("CLEAR");
		JButton print = new JButton("PRINT");
		JButton keluar = new JButton("EXIT");


		keluar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				logsframe.dispose();
			}
		});

		FileInputStream fi;
		BufferedReader buff;
		try {
			fi = new FileInputStream(filelog);
			buff = new BufferedReader(new InputStreamReader(fi));
			String baca;
			while ((baca = buff.readLine()) != null) {
				logsmap.append(baca+"\n");
			}
			fi.close();
		}catch (IOException ioe) {
			JOptionPane.showMessageDialog(logsframe,"Tidak ada log","Logs
			List",JOptionPane.WARNING_MESSAGE);
		}

		clear.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				try{
					FileWriter fw = new FileWriter(filelog);
					BufferedWriter r = new BufferedWriter(fw);
					r.write("");
						show(log,filelog);
				}catch(IOException ioe){}
```

```
                    System.gc();
                }
            });

            print.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    printme.printDialog();
                }
            });

            Container konten;
            konten = logsframe.getContentPane();
            konten.setLayout(null);
            konten.add(clear);
            clear.setBounds(380,290,80,20);
            konten.add(print);
            print.setBounds(460,290,80,20);
            konten.add(keluar);
            keluar.setBounds(540,290,80,20);

            JDesktopPane desktopin = new JDesktopPane();
            JInternalFrame inlogsframe = new JInternalFrame("",false,false,false,false);
            inlogsframe.setLocation(-28,-28);
            inlogsframe.setVisible(true);
            inlogsframe.setBackground(Color.white);
            inlogsframe.setSize(700,300 );

            Container kontenin;
            kontenin = inlogsframe.getContentPane();
            kontenin.setLayout(null);
            kontenin.add(scroll);
            scroll.setBounds(23,1,635,270);

            konten.add(inlogsframe);
            logsframe.getContentPane().setLayout(new FlowLayout());
            logsframe.getContentPane().add(inlogsframe);
            logsframe.setLocation(1,1);
            logsframe.setSize(644,356);
            logsframe.setVisible(true);

            desktopin.add(inlogsframe);
            desktopin.add(clear);
            desktopin.add(print);
            desktopin.add(keluar);
            desktopin.putClientProperty("JdesktopPane.dragMode", "outline");
            desktopin.setBackground(Color.white);
            desktop.add(logsframe);
            desktop.setBackground(Color.white);
            desktop.putClientProperty("JdesktopPane.dragMode", "outline");
            logsframe.getContentPane().setLayout(new FlowLayout());
            logsframe.setContentPane(desktopin);
            frame.getContentPane().setLayout(new FlowLayout());
            frame.setContentPane(desktop);
        }
}
```

feat.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class feat {
	public static JFrame featframe;
	public static JFrame frame = netview.frame;
	public void show() {
				featframe = new JFrame("Feature");
				JLabel keterangan = new JLabel("KETERANGAN");
				JButton green = new JButton("  ");
				green.setBackground(Color.green);
				green.setEnabled(false);
				JButton yellow = new JButton("  ");
				yellow.setBackground(Color.yellow);
				yellow.setEnabled(false);
				JButton orange = new JButton("  ");
				orange.setBackground(Color.orange);
				orange.setEnabled(false);
				JButton pink = new JButton("  ");
				pink.setBackground(Color.pink);
				pink.setEnabled(false);
				JButton red = new JButton("  ");
				red.setBackground(Color.red);
				red.setEnabled(false);
				JButton blue = new JButton("  ");
				blue.setBackground(Color.blue);
				blue.setEnabled(false);
				JButton black = new JButton("  ");
				black.setBackground(Color.black);
				black.setEnabled(false);

		featframe.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);

				JLabel labelgreen = new JLabel("     : Request sukses");
				JLabel labelyellow = new JLabel("     : Request gagal 1x");
				JLabel labelorange = new JLabel("     : Request gagal 2x");
				JLabel labelpink = new JLabel("     : Request gagal 3x");
				JLabel labelred  = new JLabel("     : Service tidak aktif");
				JLabel labelblue  = new JLabel("     : Kondisi awal");
				JLabel labelblack = new JLabel("     : Service tidak dimonitor");

				JButton keluar = new JButton("EXIT");
				keluar.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent e) {
						featframe.dispose();
						netview.mnufeat.setEnabled(true);
					}
				});

				featframe.getContentPane().setLayout(new FlowLayout());
```

```
featframe.getContentPane().add(keterangan);
featframe.getContentPane().add(keluar);
Container konten;
konten = featframe.getContentPane();
konten.setLayout(new GridBagLayout());
GridBagConstraints pos = new GridBagConstraints();
pos.anchor = GridBagConstraints.WEST;
pos.gridx = 8;
pos.gridy = 1;
konten.add(keterangan,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(green,pos);
pos.gridx++;
konten.add(labelgreen,pos);
pos.gridy++;
pos.gridx = 7;
konten.add(yellow,pos);
pos.gridx++;
konten.add(labelyellow,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(orange,pos);
pos.gridx++;
konten.add(labelorange,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(pink,pos);
pos.gridx++;
konten.add(labelpink,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(red,pos);
pos.gridx++;
konten.add(labelred,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(blue,pos);
pos.gridx++;
konten.add(labelblue,pos);

pos.gridy++;
pos.gridx = 7;
konten.add(black,pos);
pos.gridx++;
konten.add(labelblack,pos);

pos.anchor = GridBagConstraints.EAST;
pos.gridy++;
pos.gridx = 8;
```

```
                    konten.add(keluar,pos);

                    featframe.setVisible(true);

                    Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
                    int w = featframe.getSize().width;
                    int h = featframe.getSize().height;
                    int x1 = (dim.width-w)/2;
                    int y1 = (dim.height-h)/2;

                    featframe.setVisible(true);
                    featframe.setLocation(x1-60,y1-120);
                    featframe.setSize(220,280);
                    featframe.setResizable(false);
                    featframe.setVisible(true);

                }

}
```

konek.java

```
import java.io.*;
import java.awt.*;
import java.util.*;

public class konek extends Thread {
        String status, Dir, Filename, Filelog;
        String[] log = new String[28];
        private boolean running;
        private TimerTask task;
        private int count = netview.poll*1000;

        public konek() {
                running = true;
        }

        public void run() {
                while(running) {
                        connect();
                }
        }

        public void cancel() {
                stop();
                viewmap.viewframe.setTitle("View Map");
                System.gc();
        }

        public void connect() {
                Dir = netview.dir;
                Filename = netview.filename;
                Filelog = netview.filelog;
                FileInputStream fin;
        BufferedReader din;
```

```
        try {
                fin = new FileInputStream(Dir+"\\"+Filename);
        din = new BufferedReader(new InputStreamReader(fin));
        String thisLine;
        int i = 0;
                while ((thisLine = din.readLine()) != null) {
                        String isidata[] = new String[7];
                        isidata = thisLine.split(",");
                        if (isidata[6].equals("-")) {
                                log[i] = "-";
                                i++;
                        }else {
                                new cek(isidata[3],isidata[4],isidata[5],Filelog,i);
                                log[i] = cek.getmsg();
                                viewmap.viewframe.setTitle("Connecting to
                                "+isidata[3]+" at "+isidata[4]);
                                i++;
                        }
        }
        viewmap.viewframe.setTitle("Done");
 }catch (IOException e){
    }catch (NullPointerException n){
    }
                try{
                sleep(count);
    }catch (InterruptedException i) {
    }
  }
}


cek.java
import java.io.*;
import java.net.*;
import java.util.Date;

public class cek {
        InputStream is;
        BufferedReader in;
        public static String code;
        String host, service, port, filelog;
        int xx;
        int[] kode2 = netview.kode2;

        public cek(String Host, String Service, String Port, String Filelog, int X){
        host = Host;
        service = Service;
        port = Port;
        filelog = Filelog;
                xx = X;
                if(service.equals("HTTP")) {
                        HttpURLConnection conn = null;
                        BufferedReader in;
                        try {
                URL u = new URL(service.toLowerCase(), host, Integer.parseInt(port), "/");
```

```
                    conn = (HttpURLConnection) u.openConnection();
                    conn.setRequestMethod("HEAD");
                    conn.connect();
int s = conn.getResponseCode();

if(s == 200 || s == 401){
        code = "OK";
}else{
        logging();
        if (kode2[xx]==3) {
                code = "NO";
        }else if (kode2[xx]==2){
                code = "NO3";
                        netview.kode2[xx] = 3;
        }else if (kode2[xx]==1){
                code = "NO2";
                        netview.kode2[xx] = 2;
        }else if (kode2[xx]==0){
                code = "NO1";
                        netview.kode2[xx] = 1;
        }
}
        }catch(UnknownHostException u){
        logging();
                if (kode2[xx]==3) {
                code = "NO";
        }else if (kode2[xx]==2){
                code = "NO3";
                netview.kode2[xx] = 3;
        }else if (kode2[xx]==1){
                code = "NO2";
                netview.kode2[xx] = 2;
        }else if (kode2[xx]==0){
                code = "NO1";
                netview.kode2[xx] = 1;
        }
        }catch( java.io.IOException e) {
        logging();
                if (kode2[xx]==3) {
                code = "NO";
        }else if (kode2[xx]==2){
                code = "NO3";
                netview.kode2[xx] = 3;
        }else if (kode2[xx]==1){
                code = "NO2";
                netview.kode2[xx] = 2;
        }else if (kode2[xx]==0){
                code = "NO1";
                netview.kode2[xx] = 1;
        }
        }
        System.gc();
}else if(service.equals("FTP")) {

        Socket s;
```

```
DataOutputStream dos;
BufferedReader buff;
String str=null;
try{
        s = new Socket(host, Integer.parseInt(port));
        dos = new DataOutputStream(s.getOutputStream());
        buff = new BufferedReader(new
        InputStreamReader(s.getInputStream()));
        dos.writeBytes("anonymous\n");
        dos.writeBytes("anonymous@yahoo.com\n");
        str=buff.readLine();

        if (str.startsWith("220")) {
                code = "OK";
        }else {
logging();
if (kode2[xx]==3) {
        code = "NO";
}else if (kode2[xx]==2){
        code = "NO3";
                netview.kode2[xx] = 3;
}else if (kode2[xx]==1){
        code = "NO2";
                netview.kode2[xx] = 2;
}else if (kode2[xx]==0){
        code = "NO1";
                netview.kode2[xx] = 1;
}
        }
}catch(UnknownHostException u){
logging();
        if (kode2[xx]==3) {
        code = "NO";
}else if (kode2[xx]==2){
        code = "NO3";
        netview.kode2[xx] = 3;
}else if (kode2[xx]==1){
        code = "NO2";
        netview.kode2[xx] = 2;
}else if (kode2[xx]==0){
        code = "NO1";
        netview.kode2[xx] = 1;
}
}catch(IOException ioe){
logging();
        if (kode2[xx]==3) {
        code = "NO";
}else if (kode2[xx]==2){
        code = "NO3";
        netview.kode2[xx] = 3;
}else if (kode2[xx]==1){
        code = "NO2";
        netview.kode2[xx] = 2;
}else if (kode2[xx]==0){
        code = "NO1";
```

```
                netview.kode2[xx] = 1;
        }
        }
        System.gc();
}else if(service.equals("DNS")) {

        Socket s;
        try{
                String result = InetAddress.getByName(host).getHostAddress();
                s = new Socket(result, Integer.parseInt(port));
                code = "OK";
        }catch(IOException e) {
        logging();
                if (kode2[xx]==3) {
                code = "NO";
        }else if (kode2[xx]==2){
                code = "NO3";
                netview.kode2[xx] = 3;
        }else if (kode2[xx]==1){
                code = "NO2";
                netview.kode2[xx] = 2;
        }else if (kode2[xx]==0){
                code = "NO1";
                netview.kode2[xx] = 1;
        }
        }
        System.gc();
}else if(service.equals("ICMP")) {

        Socket s = null;
        DataInputStream dis = null;
        try{
                s = new Socket(host, 1000);
                dis = new DataInputStream(new
                BufferedInputStream(s.getInputStream()));
                if(s == null && dis == null){
        logging();
        if (kode2[xx]==3) {
                code = "NO";
        }else if (kode2[xx]==2){
                code = "NO3";
                        netview.kode2[xx] = 3;
        }else if (kode2[xx]==1){
                code = "NO2";
                        netview.kode2[xx] = 2;
        }else if (kode2[xx]==0){
                code = "NO1";
                        netview.kode2[xx] = 1;
        }
                }else{
                        code = "OK";
                        s.close();
                        dis.close();
                        s = null;
                        dis = null;
```

```
                    }
                }catch(NoRouteToHostException n) {
                logging();
                    if (kode2[xx]==3) {
                    code = "NO";
                }else if (kode2[xx]==2){
                    code = "NO3";
                    netview.kode2[xx] = 3;
                }else if (kode2[xx]==1){
                    code = "NO2";
                    netview.kode2[xx] = 2;
                }else if (kode2[xx]==0){
                    code = "NO1";
                    netview.kode2[xx] = 1;
                }
                }catch(ConnectException c) {
                    code = "OK";
                }catch(IOException u) {
                logging();
                    if (kode2[xx]==3) {
                    code = "NO";
                }else if (kode2[xx]==2){
                    code = "NO3";
                    netview.kode2[xx] = 3;
                }else if (kode2[xx]==1){
                    code = "NO2";
                    netview.kode2[xx] = 2;
                }else if (kode2[xx]==0){
                    code = "NO1";
                    netview.kode2[xx] = 1;
                }
                }
                System.gc();
            }
        }

        private boolean checkReply(String str) {
  if(str.length() > 3 &&
                    str.charAt(3) == ' ' &&
    Character.isDigit(str.charAt(0)) &&
    Character.isDigit(str.charAt(1)) &&
    Character.isDigit(str.charAt(2)))
  {
    return false;
  }else{
    return true;
        }
}

public static String getmsg() {
        return code;
}

        public void logging() {
                FileOutputStream fos;
```

```
                FileInputStream fis;
                BufferedReader buf;
                String isi;
                String total = "";
                Date d = new Date();
                try {
                        fis = new FileInputStream(filelog);
                        buf = new BufferedReader(new InputStreamReader(fis));
                        while ((isi = buf.readLine()) != null) {
                                total += isi+"\n";
                        }
                        fis.close();
                        fos = new FileOutputStream (filelog);
                        new PrintStream(fos).print(total);
                        new PrintStream(fos).println("Error at "+host+" ("+service+":"+port+") :
"+d);
                }catch (IOException ioe) {
                }
                System.gc();
        }
}
```