SKRIPSI

DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR

Disusun Oleh :

FEBRY DANANG EKA HERMAWAN

05.12.513

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2010

LEMBAR PERSETUJUAN

DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR

SKRIPSI

Disusun dan Diajukan sebagai Salah Satu Syarat Untuk Memperoleh
Gelar Sarjana Teknik Komputer Dan Informatika Strata Satu (S–1)

Disusun Oleh :

FEBRY DANANG EKA HERMAWAN

NIM : 05.12.513

Mengetahui	Diperiksa dan Disetujui
Ketua Jurusan Teknik Elektro S-1	Dosen Pembimbing
Ir. Yusuf Ismail Nakhoda, MT	**I Komang Somawirata, ST, MT**
NIP Y. 1018800189	NIP. Y.1030100361

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2010

DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT PADA LCD PROJECTOR

FEBRY DANANG EKA H

(05.12.513)

e-mail: Dansboenz@gmail.com
Konsentrasi Teknik Informatika dan Komputer
Jurusan Teknik Elektro
Institut Teknologi Nasional Malang
Jl. Raya Karanglo KM 2 Telp (0341) 417636 Fax (0341) 417634
Malang

Abstraksi

Pemanfaatan teknologi komputer telah berkembang dengan sangat cepat, hampir disemua instansi telah memanfaatkan teknologi ini sebagai pendukung dari perkembangan teknologi informasi yang digunakan. Tidak hanya pada internet, penggunaan jaringan komputer lokal dengan teknologi nirkabel atau wireless yang tentunya akan sangat membantu dalam menjalankan aplikasi yang selama ini masih dihubungkan dengan kabel.

Satu contoh misalnya pada setiap pemasangan personal computer (PC) pada perangkat LCD Projector, dimana jika kita ingin menampilkan tampilan desktop PC tersebut pada layar LCD projector, maka hal yang harus dilakukan adalah dengan memasang kabel dari PC ke perangkat LCD projector tersebut. Disini jelas tidak efisien dan akan terjadi pemborosan waktu jika hal tersebut harus dilaksanakan berulang kali. Berdasarkan permasalahan tersebut, dibutuhkan sebuah aplikasi berbasis client server yang dapat menampilkan tampilan desktop komputer client pada perangkat LCD projector dengan tanpa memindah atau memasang kabel koneksi dengan perangkat LCD projector.

Kata kunci : screen capture, aplikasi client server

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat serta hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Desain Aplikasi Untuk Menampilkan Screen Capture Komputer Client Ke LCD Projector" 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.

Penulis menyadari bahwa keberhasilan peyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu patut kiranya penulis 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. Bapak I Komang Somawirata, ST, MT selaku Dosen Pembimbing.
4. Ayah dan Ibu serta keluarga yang telah memberikan do'a restu, dorongan, semangat, dan motivasi dalam menyusun skripsi ini.
5. Rekan – rekan mahasiswa Teknik Informatika dan Komputer ITN Malang angkatan 2005 dan 2006.
6. Rekan-rekan Instruktur di Laboratorium Jaringan Komputer ITN Malang.
7. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penulis telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu saran dan kritik yang bersifat membangun dari pembaca sangat penulis harapkan untuk kesempurnaan laporan ini selanjutnya.

Penulis berharap semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan.

Malang, Agustus 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemanfaatan teknologi komputer telah berkembang dengan sangat cepat, hampir disemua instansi telah memanfaatkan teknologi ini sebagai pendukung dari perkembangan teknologi informasi yang digunakan. Kebutuhan akan pertukaran data serta pemanfaatan suatu sumber daya yang dipakai bersama adalah beberapa faktor yang menyebabkan pentingnya peran jaringan komputer dalam lingkup kerja. Tidak hanya pada internet, penggunaan jaringan komputer lokal untuk menjalankan beberapa aplikasi juga telah banyak digunakan, terlebih lagi dengan pemanfaatan jaringan komputer dengan teknologi nirkabel atau *wireless* yang tentunya akan sangat membantu dalam menjalankan aplikasi yang selama ini masih dihubungkan dengan kabel. Diharapkan dengan pemanfaatan teknologi *wireless* ini dapat lebih meningkatkan efisiensi dan kemudahan untuk menjalankan aplikasi yang selama ini masih menggunakan kabel sebagai medianya.

Satu contoh misalnya pada setiap pemasangan *personal computer* (PC) pada perangkat LCD *projector*, dimana jika kita ingin menampilkan tampilan desktop PC tersebut pada layar LCD *projector*, maka hal yang harus dilakukan adalah dengan memasang kabel dari PC ke perangkat LCD *projector* tersebut. Disini jelas tidak efisien dan akan terjadi pemborosan waktu jika hal tersebut harus dilaksanakan berulang kali.

Berdasarkan permasalahan tersebut, dibutuhkan sebuah aplikasi berbasis *client server* yang dapat menampilkan tampilan desktop komputer *client* pada perangkat LCD *projector*. Selain aplikasi tersebut, juga dibutuhkan jaringan komputer dengan teknologi *wireless* sebagai media koneksi antara *client* dengan *server,* sehingga untuk menampilkan tampilan desktop komputer *client* hanya dengan mengetikkan alamat IP komputer *client* dan tanpa memasang atau memindah kabel pada perangkat LCD *projector*.

1.2 Rumusan Masalah

Dari latar belakang yang telah dikemukakan, maka dapat ditentukan rumusan masalah yaitu bagaimana membuat sebuah aplikasi yang dapat menampilakan *screen capture* komputer *client* pada LCD *projector*, mengakses *drive-drive* yang tersedia, melakukan *file transfer*, dengan menggunakan teknologi *wireless*.

1.3 Tujuan

Tujuan dari skripsi ini adalah untuk menampilkan *screen capture* dari komputer *client* ke LCD *projector* dan mens*witch* tampilan desktop antar komputer *client* tanpa memasang atau memindah kabel koneksi antar komputer dengan LCD *projector*. Sehingga diharapkan waktu yang digunakan untuk menampilkan tampilan desktop komputer *client* ke LCD *projector* dapat lebih efisien.

1.4 Batasan Masalah

Dalam pengerjaan skripsi ini diberi batasan-batasan sebagai berikut :

1. Pembuatan aplikasi untuk menampilkan *screen capture* pada perangkat LCD *projector* dan men*switch* tampilan *screen capture* antar PC *client.*
2. Lingkup dari desain dan implementasi aplikasi ini hanya pada jaringan komputer LAN *(Local Area Network)* dan pada jaringan *intranet.*
3. Tidak membahas teknologi jaringan dan sistem keamanan jaringan.
4. Pengembangan Aplikasi untuk menampilkan *screen capture* ini dibuat pada aplikasi berbasis windows.

1.5 Metodologi

Metodologi yang digunakan dalam penyusunan skripsi ini dapat dijelaskan sebagai berikut:

(a) Studi literatur.

Studi literatur ini sebagai dasar teori yang akan melandasi pengerjaan aplikasi ini dimulai dengan mencari literatur-literatur yang berkaitan di internet maupun pada perpustakaan. Sebagai bahan pertimbangan untuk pembuatan skripsi ini.

(b) Pembuatan perangkat lunak.

Untuk mengaktifkan hubungan antara PC *server* dengan PC *client* maka perlu dibuat perangkat lunak. Untuk pembuatan perangkat lunak meliputi

Pembuatan aplikasi untuk menampilkan *screen capture* pada *projector* dan men*switch* tampilan *screen capture* antar PC *client*

(c) Pengujian jaringan dan Perangkat Lunak

Dari hasil perancangan, dilakukan realisasi dan pembuatan baik penginstalan jaringan maupun pembuatan perangkat lunak serta diadakan pengukuran dan pengujian masing–masing dari perangkat – perangkat tersebut.

(d) Integrasi dan Pengujian Sistem

Hasil dari penginstalan jaringan dan pembuatan perangkat lunak, diintegrasikan ke dalam satu sistem kerja untuk bisa dijalankan sesuai dengan fungsinya yaitu mengaktifkan aplikasi untuk menampilkan *screen capture* ke LCD *projector* dengan *wireless connection.*

(e) Eksperimen dan Analisa Sistem

Sistem yang sudah dibangun, terintegrasi dan unjuk kerjanya dianggap memadai dapat digunakan untuk menampilkan *screen capture* pada LCD *projector* dan men*switch* tampilan *screen capture* antar computer *client* secara *wireless* dengan menggunakan Acces Point.

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 : Landasan Teori

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

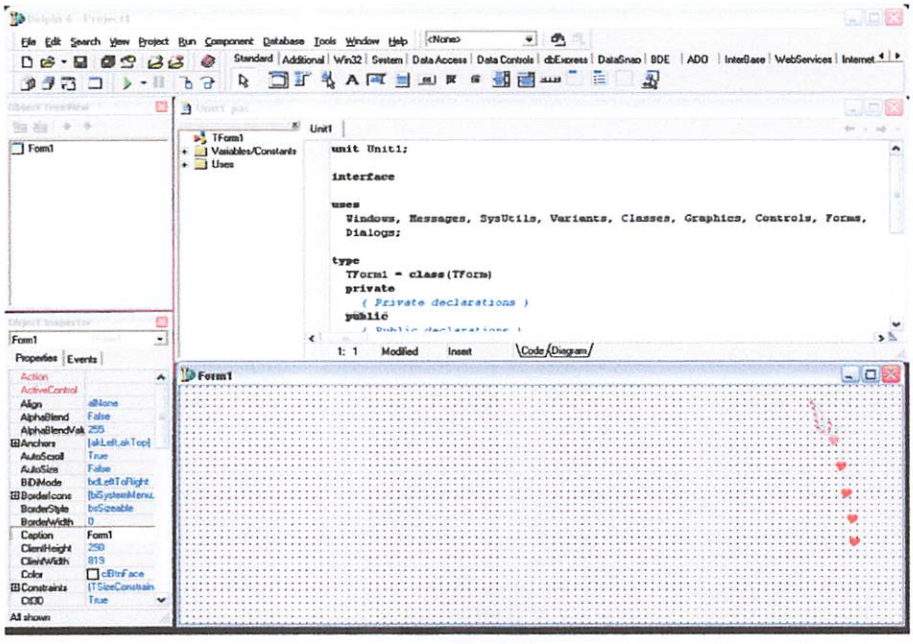
2.1 Delphi 7 Software

Delphi merupakan bahasa pemrograman yang mudah, karena Delphi adalah bahasa perograman tingkat tinggi (*high level*) sehingga sangat memudahkan user untuk bermain-main di tingkat ini, apa lagi bagi mereka yang malas berurusan dengan level-level yang rendah.

Pemrograman Delphi sangatlah mudah, kita tinggal *click and drag*, dan jadilah program aplikasi yang kita inginkan.

2.1.1 Form dan unit

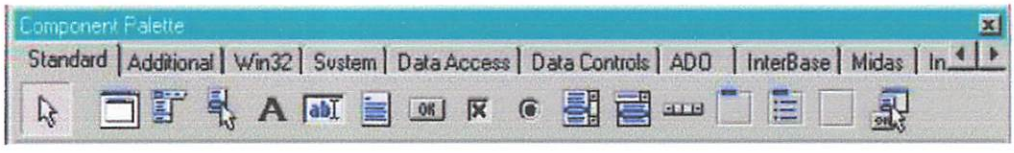
Bila pada pascal kita akan melihat tampilan yang menjemukan, pada Delphi kita bisa mengatur tampilan kita semenarik mungkin pada *form* yang kita gunakan. Caranya cukup mudah, kita hanya menaruh komponen-komponen yang kita inginkan pada *form* tersebut, dan memfungsikan masing-masing komponen sesuai dengan yang kita inginkan.



Gambar 2.1 Tampilan Form Delphi

Setiap perubahan pada *form* akan berakibat perubahan pada unit yang kita miliki, Untuk pindah dari *form* ke unit, anda bisa tekan **F12.**

2.1.2 Component Pallete



Gambar 2.2 Component Palette

Dalam membuat program, Delphi telah menyediakan banyak kemudahan, yaitu dengan disediakannya komponen-komponen. Komponen ini merupakan

sebuah *procedure* program yang sudah di *compile* dan langsung dapat digunakan, sesuai dengan fungsinya masing-masing.

Untuk menggunakan komponen ini kita dapat meng-klik komponen yang diinginkan, kemudian kita klik di *form*, maka komponen tersebut akan muncul di *form*.

Kegunaan beberapa komponen :

a. **Button/ Bitbtn**

   Biasa digunakan sebagai tombol kendali. Perbedaan antara bitbtn dengan btn : pada bitbtn kita dapat menyisipkan warna pada tombol dan icon tertentu, lain halnya bila kita menggunakan btn.

b. **Panel**

   Panel berfungsi untuk mengelompokan komponen-komponen didalamnya.

c. **Label**

   Kita dapat menamakan atau memberi keterangan pada program.

d. **Edit**

   Edit berfungsi sebagai masukan data (input) dalam bentuk *string*, dari bentuk *string* ini kita dapat mengolahnya menjadi bentuk integer atau bentuk lainnya. Yang kemudian dapat digunakan untuk operasi selanjutnya.

e. **Chart**

Data-data yang telah kita analisa, dapat kita tampilkan ke dalam grafik, sehinga memudahkan kita untuk menganalisanya.

f. **Stringgrid**

Stringgrid berguna untuk menaruh data *string* kedalam bentuk kolom tabel, seperti pada Excel. Kita harus mengubah tipe data ke dalam bentuk *string* bila data yang ingin kita tampilkan data bukan *string*.

g. **PopupMenu**

Popup Menu berfungsi sebagai perintah yang aktif bila kita meng-klik kanan mouse, Untuk mengaktifkannya kita harus mengaktifkan popup menu pada komponen yang dinginkan, caranya : ubah pada object inspector.

h. **MainMenu**

Contoh main menu adalah Option pada tiap aplikasi porgram, dengan komponen ini, kita bisa menaruh fungsi-fungsi program seperti pada aplikasi umumnya.

i. **ComboBox**

Combo Box berfungsi sebagai petunjuk untuk pemilihan berbagai masukan. Lihat contoh.

j. CheckBox

Bila komponen ini di check maka ada aplikasi yang bisa disetting untuk bekerja dibawahnya.

k. RadioButton

Prinsip kerjanya hampir sama dengan check box, cuma tampilannya saja yang berbeda.

l. Media Player

Biasa digunakan untuk mennyalakan atau memainkan musik (format wav atau midi) dan menjalankan film (format avi).

m. Timer

Timer berfungsi sebagai jam yang telah disediakan Delphi. Dengan timer kita juga dapat mendecode time, sehingga dapat terjadi akusisi data.

n. Clientsocket dan Serversocket

Digunakan untuk menyediakan layanan koneksi *client server.*

2.1.3 Object Inspector

Object inspector (gambar 2.3) berguna sebagai options dari masing masing komponen. Dengan *object inspector* ini kita dapat memanipulasi komponen yang kita gunakan (walaupun sebenarnya kita juga dapat menggunakannya dengan menuliskannya lewat *text mode*).

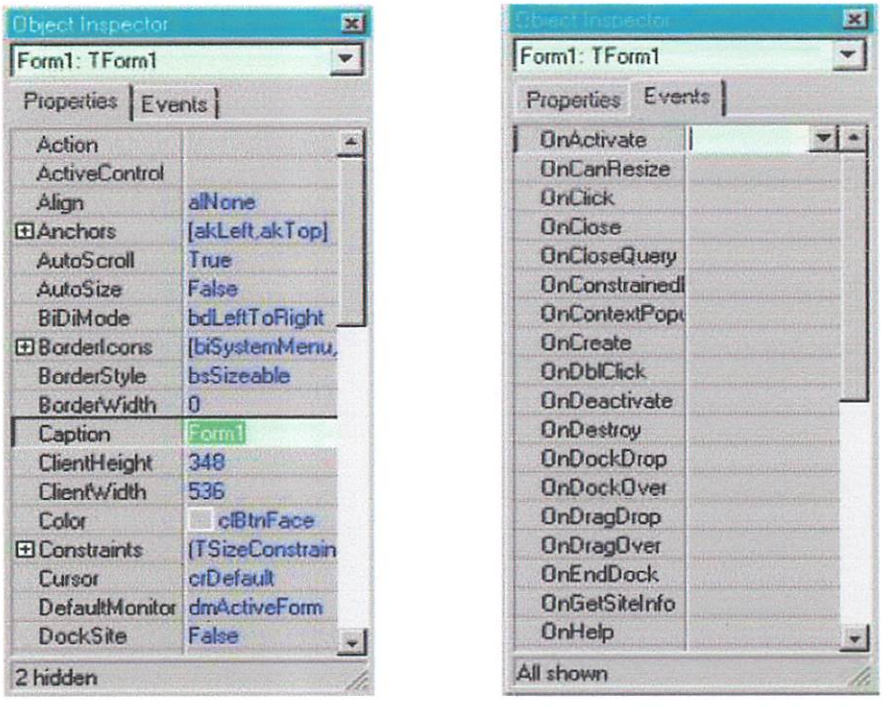
Beberapa contoh pada Propertis :

a. Font

Menunjukan *font* yang akan kita gunakan. Dalam menu ini juga terdapat macam-macam *font* berikut *style* dan ukurannya.

b. Caption

Merupakan judul dari *form* itu. Perhatikan perbedaannya dengan name.



Gambar 2.3 Object Inspector

c. **Name**

Merupakan nama dari komponen itu. Biasanya bila kita mengambil sebuah komponen, delphi otomatis menyediakan nama sesuai dengan nama komponen tersebut. Namun nama ini dapat dengan mudah kita ubah agar mudah mengingatnya. Berbeda dengan *caption*, pada *caption* kita tidak mengubah nama komponen, melainkan hanya mengubah judul yang tertera pada komponen tersebut. Sehingga bila kita ingin menggunakan komponen tersebut, yang kita gunakan adalah namanya, dan bukan nama pada *caption*.

d. **Enabled**

Menunjukan bahwa komponen ini bisa digunakan atau tidak. Perintah ini cukup penting bila kita tidak ingin *user* menekan tombol yang tidak diinginkan.

e. **Height**

Merupakan tinggi dari komponen yang akan kita taruh pada *form*, atau komponen lainnya.

f. **Width**

Merupakan lebar komponen.

g. **Popup menu**

Fungsi ini berguna saat kita mengklik kanan dan akan tampil perintah. Pada penggunaanya kita mengambuil komponen *popup* menu dan mengeset *popup* menu pada *form*.

h. **Auto size**

Fungsi ini berisi mengenai apakah *form* yang kita miliki bisa diperbesar atau tidak.

i. **Text**

Biasanya tedapat pada komponen edit dan *mask edit*. Kegunaanya untuk menaruh kata-kata pada komponen ini.

j. **Color**

Warna pada komponen. Kita dapat mengubah warna-warna tiap komponen (hanya yang disediakan) dengan point ini.

k. **Visible**

Berfungsi untuk menampilkan dan tidak menampilkan komponen yang kita inginkan.

l. **Hint**

Bila kursor kita diatas komponen, maka akan muncul suatu keterangan. Keterangan ini yang disebut dengan hint. Jangan lupa untuk mengganti properti *show hint* menjadi *true* untuk mengaktifkannya.

m. **Borderstyle**

Menunjukkan berbagai macam bingkai yang diinginkan. Dengan *Object inspector* kita juga dapat menentukan *event* apa yang akan kita buat dengan komponen tersebut. Gambar 2.3 menunjukan *object inspector* untuk *event* pada *form*.

Beberapa contoh event :

1. **Onclick**

Bila kita mengklik komponen tersebut maka prosedur yang kita inginkan akan dijalankan oleh program tersebut.

2. **Onkeypress**

Bila kita menekan suatu *key* (tombol) maka komponen tersebut akan aktif. Perlu diperhatikan pada event ini, delphi mengenal karakter yang kita tekan melalui kode yang masuk ke dalam variabel *key*.

2.1.4 File-file pada Delphi .

a. ***.pas**

Merupakan *source file*, disini akan disimpan kode pascal yang kita tulis.

2. *.dpr

Merupakan *project file*. Sebagai *project file*, file ini berguna untuk menghubungkan satu atau lebih *file-file source* (*.pas).

3. *.dcu (Delphi compiled Unit)

Pada saat kita membuat sebuah aplikasi, Delphi akan membuat file ini. File ini berfungsi untuk me-link-kan kita dengan file lain (*.dcu) sehingga kita bisa membuat beberapa *form* yang terhubung satu dengan yang lainnya.

4. *.dfm (Delpi Form)

File ini berisi informasi mengenai data-data *form*.

- *.res (Windows Resources)
- *.dof (Delphi project options)

Kita dapat mengubah icon aplikasi tersebut, dan datanya disimpan pada file ini.

- *.exe

Merupakan *appllication file* setelah kita *compile* program kita.

2.1.5 Pemrograman Socket

Pemograman soket merupakan dasar dari berbagai protokol TCP/IP seperti HTTP, SMTP, POP3, FTP dan sebagainya. Soket sendiri pada mulanya digunakan pada Berkeley UNIX untuk komunikasi TCP. Dalam perkembangannya soket

menjadi standar pada hampir semua sistem operasi, termasuk windows. Soket dalam sistem operasi *Windows* sering disebut juga *WinSock*.

Soket dalam dunia internet digunakan untuk mengkoneksi program kita ke komputer lain. Untuk menghubungkan soket satu dengan soket lainnya menggunakan alamat IP. Selain itu soket juga berhubungan dengan *port*. *Port* tersebut digunakan untuk memberi alamat koneksi pada suatu komputer. Dengan kata lain, jika IP digunakan untuk memberikan alamat pada komputer yang berbeda, *port* digunakan untuk memberikan alamat pada komputer yang sama.

Ada beberapa nomor *port* yang sudah menjadi standart, misalnya 80 untuk HTTP, 23 untuk Telnet, 25 untuk SMTP, 21 untuk FTP dan sebagainya. *Port-port* yang sudah menjadi standar ini terdapat pada RFC 1700. *Port* dibawah 1024 biasanya tidak boleh digunakan karena telah ada standar untuk masing-masing nomer. Apabila ingin menggunakan *port* sebaiknya menggunakan *port* yang diatas 1024, kecuali menggunakan protokol yang menjadi standar, seperti HTTP. Biasanya *port* yang digunakan adalah 22222.

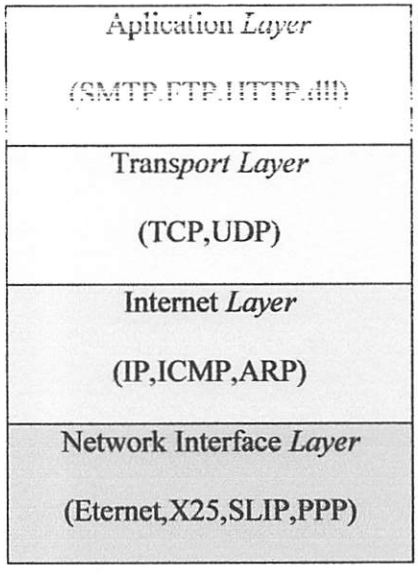
Pada *Delphi*, ada dua komponen utama untuk pemograman soket yaitu T*ServerSocket* dan T*ClientSocket*. T*ServerSocket* digunakan untuk mendengarkan pada *port* tertentu, sedangkan T*ClientSocket* digunakan untuk menghubungkan [illegible] digunakan untuk protokol TCP dan apabila menggunakan protokol UDP terdapat komponen lain yaitu [illegible].

2.2 Dasar Arsitektur TCP/IP

TCP/IP (*Transmission Control Protocol / Internet Protocol*) adalah sekelompok protokol yang mengatur komunikasi data komputer di Internet. Karena menggunakan bahasa yang sama, yaitu protokol TCP/IP, perbedaan jenis komputer dan sistem operasi tidak menjadi masalah. Komputer komputer yang saling terhubung tersebut tetap dapat berkomunikasi

2.2.1 Layer pada TCP IP

TCP/IP didefinisikan dalam 4 *layer* model, yaitu seperti digambarkan dalam diagram di bawah ini :



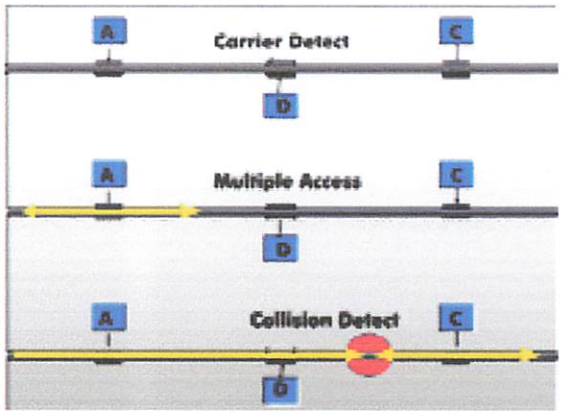
Gambar 2.4 Layer-layer TCP/IP[1]

2.3.1.1 Network Interface Layer

Network interface layer merupakan lapisan yang bertanggung jawab mengirimkan dan menerima data ke dan dari media fisik, media fisiknya dapat berupa kabel, serat optik, atau gelombang radio. Karena itu protokol pada *layer* ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang dimengerti komputer. Beberapa contoh dari *network interface* adalah *Ethernet.*

Ethernet merupakan sebuah kartu yang terhubung ke kartu lain melalui *ethernet hub* dengan kabel UTP atau hanya menggunakan kabel BNC yang diterminasi ujungnya. Dasar pemikiran dirancangnya *ethernet* ialah "berbagi kabel". Sehingga dua atau lebih komputer dapat berkomunikasi. Karena menggunakan satu kabel saja, maka proses pemancaran data harus dilakukan bergantian. Sebelum satu kartu *ethernet* mengirimkan datanya pada kabel, dia harus mendeteksi terlebih dahulu ada tidaknya kartu lain yang sedang memancar. Jika tidak ada maka dia akan memancarkan data tersebut. Jika ada maka kartu tersebut akan menunggu sampai kabel dalam keadaan kosong (tidak terpakai).

Apabila dua atau lebih station mengirimkan data pada saat yang bersamaan maka akan terjadi *collision* (tabrakan) sehingga tiap terminal yang terlibat akan mengalami random transmit. Dalam pengertian sederhananya setiap terminal akan diam dan menghitung secara acak sebelum melakukan transmit ulang.



Gambar 2.5 Deteksi collision (tabrakan)[2]

Selanjutnya kedua *station* akan mundur selama beberapa mikrodetik (yang dipilih secara acak) dan mencoba lagi. Keacakan periode mundur tersebut untuk masing-masing *station* akan mengurangi risiko tabrakan pada usaha kedua mereka. *Station* yang mengirim informasi akan bisa mengirim dengan tenang dan bisa mempersiapkan pesan berikutnya, tetapi pesan pertama yang bertabrakan dengan lalu lintas lain pada *switch* tujuan akan dibuang. Walaupun hal itu tak berarti hilangnya data. Jika *adapter port* tujuan mendrop sebuah pesan, protokol tingkat yang lebih tinggi akan memperbaikinya. TCP/IP akan bermanfaat dalam situasi seperti itu karena TCP akan mendeteksi kehilangan data dan mentransmisikan ulang setelah selang waktu tertentu.

Collision tidak mendeteksi kesalahan atau error pada data, karena ini hanya menunjukan metoda akses yang dipakai pada jaringan *ethernet* dengan CSMA/CD. *Failure* pada paket digunakan dcteksi CRC, makin banyak peluang terjadinya *collision*, maka proses transfer data setiap terminal menjadi turun (bukan kuantitas) tapi kualitasnya. Namun *collision* bukanlah ukuran kualitas sebuah network. *Collision* indikator dalam *ethernet* untuk melakukan *random*

delay (penundaan sesaat dalam jangka waktu acak) sebelum transmit agar tidak terjadi tabrakan data. Jika *retry-time* terjadi lebih dari misalkan 10x pada paket yang sama, maka kemungkinan ada masalah pada network. *Collision* sebenarnya adalah cara *ethernet* memberitahu kepada *source* : "jangan kirim sekarang, coba nanti".

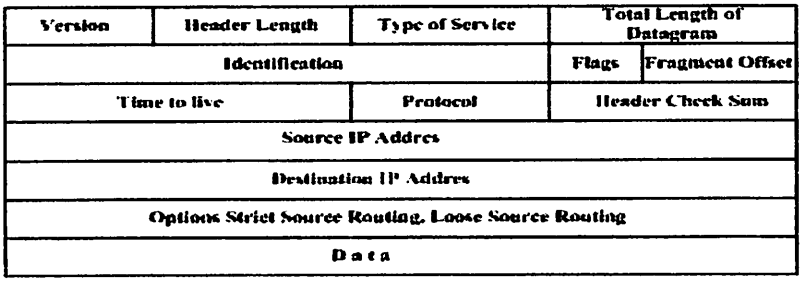
2.2.1.2 Internet Layer

Internet Layer adalah lapisan yang bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. *Layer* ini ada tiga macam protokol, yaitu IP, ARP dan ICMP. IP berfungsi untuk menyampaikan paket data ke alamat yang tepat. ARP *(Address Resolution Protocol)* ialah protokol yang digunakan untuk menemukan alamat hardware dari komputer yang terletak pada *network* yang sama. Sedangkan ICMP (Internet Control Message Protocol) ialah protokol yang digunakan untuk mengirimkan pesan dan melaporkan kegagalan pengiriman data.

Protokol IP merupakan inti dari protokol TCP/IP. IP *(Internet Protocol)* ialah lapisan Internet yang paling utama yang bertanggung jawab untuk pengiriman data melalui jaringan. Dalam pengiriman data, IP memiliki sifat yang dikenal sebagai *unreliable, connectionless* dan *datagram delivery service*.

Unreliable / ketidakhandalan berarti bahwa protokol IP tidak menjamin *datagram* yang dikirim pasti sampai ke tempat tujuan. Jika diinginkan kehandalan yang lebih baik, kehandalan itu harus disediakan oleh protokol yang berada diatas *layer* IP ini (yaitu TCP dan aplikasi pengguna). *Connectionless* berarti bahwa dalam pengiriman paket dari tempat asal ke tujuan, pihak pengirim dan penerima paket IP sama sekali tidak mengadakan perjanjian terlebih dahulu. *Datagram*

delivery service berarti setiap paket data yang dikirim adalah independent terhadap paket data yang lain. Akibatnya, jalur yang ditempuh oleh masing-masing paket data IP ke tujuannya bisa jadi berbeda satu sama yang lainnya. Karena jalur yang yang ditempuh berbeda, kedatangan paket pun bisa jadi tidak berurutan. Format *datagram* IP seperti pada gambar 2.6.



Gambar 2.6 Diagram format datagram IP[3]

Setiap paket IP membawa data terdiri atas :

1. *Version*, berisi versi dari protokol yang dipakai.
2. *Header Length*, berisi panjang dari header paket IP. Ini dalam hitungan 32 bit *word*.
3. *Type of Service*, berisi kualitas servis yang dapat mempengaruhi cara penanganan paket IP ini.
4. *Total Length of Datagram*, panjang IP *datagram* total dalam ukuran *byte*.
5. *Identification Flags* dan *Fragment Offset*, berisi beberapa data yang berhubungan dengan *fragmentasi* paket.
6. *Time to Live*, berisi jumlah *router* maksimal yang boleh dilewati paket IP. Setiap kali paket IP melewati satu *router*, isi dari *field* ini dikurangi satu.

7. Protokol mengandung angka yang mengidentifikasi protokol *layer* atas pengguna isi dari paket IP ini
8. *Header Checksum*, berisi nilai *checksum* yang dihitung dari seluruh *field* dari *header* paket IP. Sebelum dikirim, protokol IP menghitung dahulu *checksum header* paket IP tersebut, untuk nantinya dihitung kembali ke sisi penerima. Jika terjadi perbedaan maka paket ini dianggap rusak dan dibuang
9. *IP address* dan penerima data, berisi alamat pengirim dan penerima paket.
10. Beberapa *byte option* antara lain:
    - *Strict Source Routing*, berisi daftar lengkap *IP address* dari *router* yang harus dilalui oleh paket data menuju *host* tujuan
    - *Loose Source Paket*, paket yang dikirim singgah ke beberapa *router*.

Dalam *IP address* ada lima kelas yaitu kelas A, kelas B, kelas C, kelas D dan kelas E

- Kelas A cocok untuk mendisain organisasi komputer yang jumlahnya sangat besar dalam jaringannya
- Kelas B cocok untuk mendisain organisasi komputer dalam jumlah menengah.
- Kelas C cocok untuk mendisain organisasi komputer dalam jumlah banyak masing masing masing dengan sedikit *host*. Kelas ini dapat digunakan pada suatu *internetwork* korporat yang terdiri dari *Local Area Network* dalam jumlah besar.
- Kelas D digunakan untuk tujuan *multicasting*. Dalam kelas ini tidak dibahas menganai *netid* dan *hostid*.

- Kelas E disisakan untuk penggunaan khusus, biasanya untuk kepentingan riset juga tidak ada *netid* dan *hostid*.

TCP/IP saat ini dipergunakan dalam banyak jaringan komputer lokal (LAN) yang terhubung ke Internet, karena memiliki sifat:

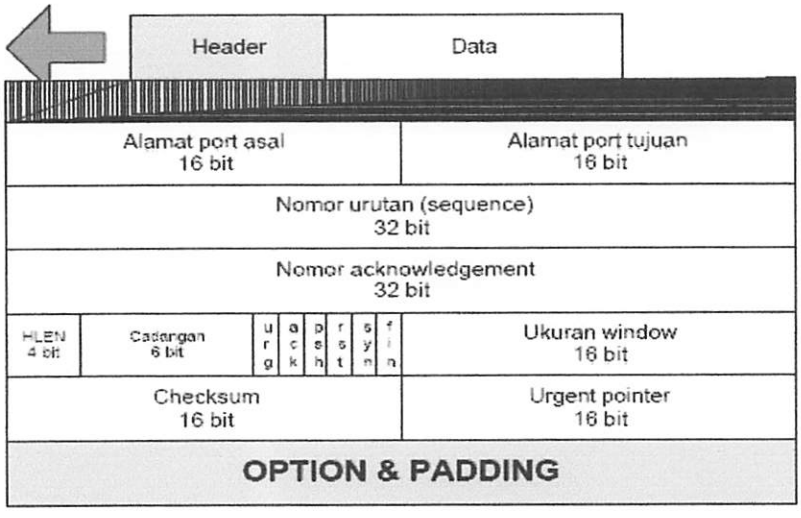
1. Merupakan protokol standar yang terbuka, gratis dan dikembangkan terpisah dari perangkat keras komputer tertentu. Karena itu protokol ini banyak didukung oleh *vendor* perangkat keras, sehingga TCP/IP merupakan pemersatu perangkat keras komputer yang beragam merk begitu juga sebagai pemersatu berbagai perangkat lunak yang beragam merk.
2. Berdiri sendiri dari perangkat keras jaringan apapun. Sifat ini memungkinkan TCP/IP bergabung dengan banyak jaringan komputer. TCP/IP bisa beroperasi melalui sebuah *Ethernet*, sebuah *token ring*, sebuah saluran *dial-up*, sebuah X-25 dan secara vitrual melalui berbagai media fisik transmisi data.
3. Bisa dijadikan alamat umum sehingga tiap perangkat yang memakai TCP/IP akan memiliki sebuah alamat unik dalam sebuah jaringan komputer lokal, atau dalam jaringan kumputer global seperti Internet.

2.2.1.3 Transport Layer

Transport Layer merupakan lapisan yang bertanggung jawab untuk mengadakan komunikasi antara dua komputer dimana jaminan pengiriman data persis seperti pada saat pengiriman data. Fungsi ini dilakukan oleh TCP *(Transmission Transport Protocol)*. UDP *(User Datagram Protocol)* juga merupakan layanan *Transport Layer* hanya saja UDP tidak mementingkan pemeriksaan kcandalan komunikasi *end to end*.

2.2.1.3.1 Transmission Control Protocol (TCP)

Unit data yang ditransfer melalui TCP disebut dengan Segmen. Segmen memuat 20 – 60 *byte header*. Jika tanpa *option*, besar header hanya 20 *byte*. Format segmen TCP dapat dilihat pada gambar 2.7



Gambar 2.7 Format segmen TCP[4]

TCP melakukan *process-to-process communication*. Ada beberapa layanan pada TCP antala lain :

- *Stream Data Servis*

  TCP melakukan layanan stream data pada lapisan *transport*. Untuk pengiriman *stream*, pengirim dan penerima TCP menggunakan *buffer*. Data yang dilalukan secara *streaming* itu berupa segmen-segmen.

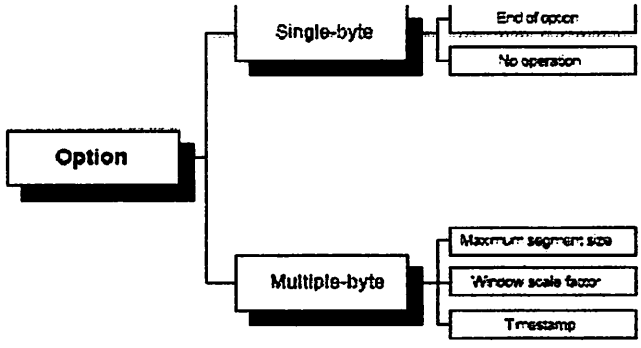
- Layanan *Full Duplex*

TCP memberikan layanan *full duplex*, dimana data dapat berpindah dalam dua arah pada saat bersamaan.

- Layanan *Reliable*

  TCP merupakan protokol di lapisan *transport* yang sifatnya *reliable*. Karena TCP menggunakan mekanisme *acknowledgement*.

Header TCP dapat bertambah besar sampai penambahan maksimal 40 byte yang disebut header option. Kategori option dapat dilihat seperti pada gambar 2.8



Gambar 2.8 Option pada TCP[5]

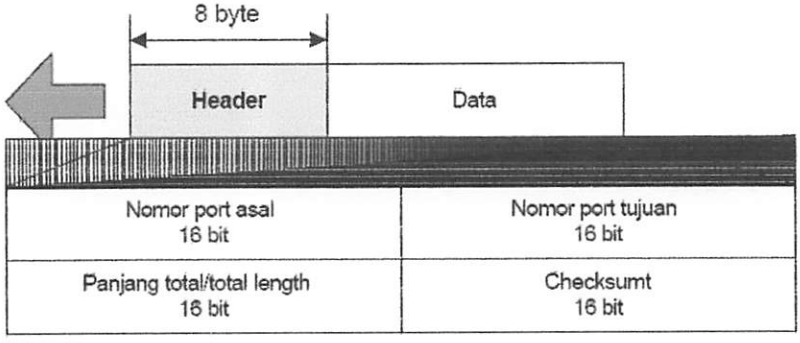
Beberapa keuntungan yang didapat dari TCP adalah melakukan operasional sebagai berikut : enkapsulasi dan dekapsulasi (pembungkusan dan pembuka bungkusan), *queuing* (pengantrian), *multiplexing* dan *demultiplexing*, *pushing* data.

2.2.1.3.2 User Data Protokol (UDP)

UDP melakukan *process-to-process communication*. UDP merupakan protokol yang sederhana dan tidak handal yang hanya mampu mengirimkan data saja tanpa berusaha untuk mengembalikan *datagram* yang hilang. Sebagai proses

pada lapisan atas harus bertanggung jawab untuk mendeteksi data yang hilang atau rusak dan mengirimkan ulang data tersebut bila dibutuhkan.

UDP tidak mengurus masalah penerimaan aliran data dan pembuatan segmen yang sesuai untuk IP. Akibatnya, UDP adalah protokol sederhana yang berjalan dengan kemampuan jauh dibawah TCP.



Gambar 2.9 Format Datagram User (UDP)[6]

Keterangan :

- Nomor *port* asal dan tujuan berisi angka yang mengidentifikasi aplikasi pengirim dan penerima segmen UDP ini.
- Panjang total/total length berisi panjang dari *datagram* dan biasanya juga disebut *datagram length*.
- *Checksum* berisi angka hasil perhitungan matematis yang digunakan untuk memerikasa kesalahan data.

Kegunaan protokol UDP antara lain :

- UDP cocok untuk proses yang memerlukan request-respon communication dan sedikit sekali memperhatikan masalah *flow control* dan *error control*.
- UDP yang memerlukan proses dengan mekanisme internal *flow control* dan *error control* hanya untuk proses TFTP (*Trivial File Transfer Protocol*).

- UDP cocok untuk multicasting dan broadcasting pada lapisan trans*port.*
- UDP digunakan untuk manajemen proses seperti aplikasi SNMP.
- UDP digunakan pengupdate protokol ruting, seperti pada RIP (*Routing Information Protocol*).

2.2.1.4. Application

Aplication *layer* adalah lapisan paling atas dalam hararki dimana proses jaringan yang bisa diakses user berada dan berinteraksi. Atau dengan kata lain terletak semua aplikasi yang menggunakan TCP/IP. Contoh dari protokol yang berada pada *layer* ini adalah, SMTP, HTTP, FTP dan lain-lain.

2.3 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, *harddisk, 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.

2.3.1 Jenis Jaringan Komputer.

Jenis - jenis jaringan komputer dapat dibedakan sebagai berikut :

2.3.1.1. Berdasarkan cakupan areanya, maka jaringan komputer dibedakan sebagai berikut :

1. 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.

2. 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).

3. 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.

2.3.1.2. Berdasarkan media transmisi datanya, dapat dibedakan menjadi :

1. Jaringan Berkabel (Wired Network),

   Untuk menghubungkan satu komputer dengan komputer lain diperlukan penghubung berupa kabel jaringan, yang berfungsi dalam mengirim informasi dalam bentuk sinyal listrik antar komputer jaringan. Ada dua kabel yang sering digunakan yaitu :

   - *Unshielded twisted-pair* (UTP)

     Adalah sebuah jenis kabel jaringan yang menggunakan bahan dasar tembaga, yang tidak dilengkapi dengan *shield* internal. UTP merupakan jenis kabel yang paling umum yang sering digunakan di dalam jaringan lokal (LAN), karena memang harganya yang rendah, fleksibel dan kinerja yang ditunjukkannya relatif bagus. Dalam kabel UTP, terdapat insulasi satu lapis yang melindungi kabel dari ketegangan fisik atau kerusakan.

   - *Shielded twisted pair* (STP)

     Adalah kabel pasangan berpilin yang memiliki perlindungan dari logam untuk melindungi kabel dari intereferensi elektromagnetik luar.

2. Jaringan *Wireless*

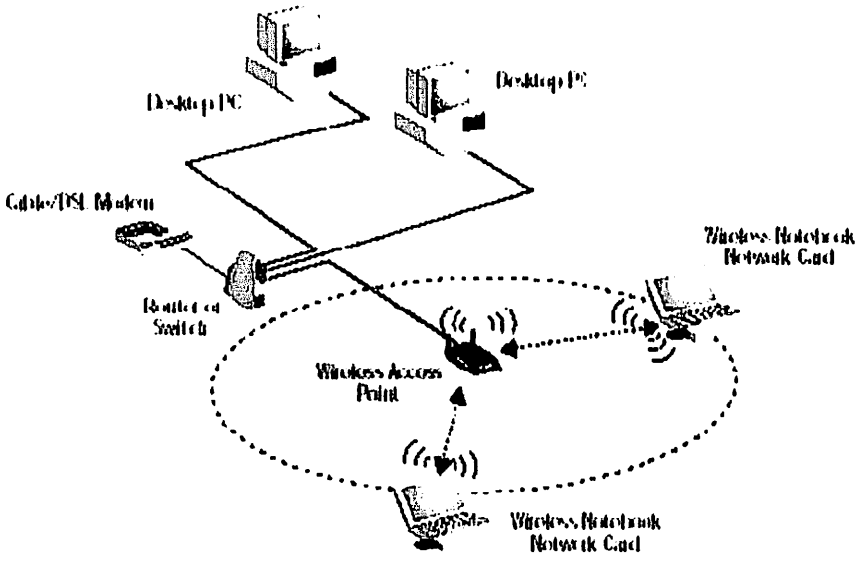
   Jaringan nirkabel atau *wireless* adalah suatu jaringan area lokal nirkabel atau tanpa kabel yang menggunakan gelombang radio sebagai media tranmisi untuk komunikasi antara perangkat komputer. Prinsip dasar dari jaringan *Wireless* LAN sama saja dengan jaringan

pada ethernet card hanya beda pada media transmisi yang melalui udara.

Access point (AP) pada *wireless* LAN berfungsi mirip sebagi *switch*, tanpa *access point* peralatan *wireless* (komputer yang mempunyai wereless adapter) hanya dapat berkomunikasi lewat point to point (dua komputer atau lebih).

Agar semua komputer dapat berkomunikasi dengan WLAN yang sama, pada *access point* akan mengeluarkan sinyal (code) SSID (*Service Set Identification*) dan pada semua komputer yang akan terhubung dengan *access point* tersebut harus dikonfigurasi mengunakan SSID yang di keluarkan accsess point tersebut Kelebihan utama dari jaringan *wireless* adalah mobilitas dan terbebasnya perangkat dari kerumitan bentangan kabel.

Selain itu ada beberapa kekurangannya, antara lain adanya interferensi radio oleh cuaca, perangkat *wireless* lain, halangan tembok, gedung, bukit, gunung atau bahkan pohon besar yang tinggi.



Gambar 2.10 Jaringan Wireless LAN (WLAN)[7]

Dengan Memakai *access point* jaringan dapat di hubungkan antara jaringan yang mengunakan kabel dan jaringan yang menggunakan perangkat *wireless*. Standar yang di gunakan pada perangkat *wireless* yang di pakai IEEE (*Institute of Electrical and Electronic Engineers*) adalah 802.11. Untuk lebih lengkapnya :

- 802.11b

  Digunakan mulai akhir tahun 1999 dengan menggunakan frekuensi 2,4 GHz , maksimum bandwidth yang bisa di capai 11 Mbps, modulasi sinyal yang di gunakan adalah DSSS. Kanal yang tidak *overlapping* 3. Kompatibel dengan *type* g jika *type* g jalan pada mode mixed.

- 802.11a

  Digunakan mulai akhir tahun 2001 dengan menggunakan frekuensi 5 Ghz, Maksimum bandwidth yang bisa di capai 54 Mbps, modulasi siny al yang di gunakan adalah OFDM. Kanal yang tidak overlapping 12 (bisa lebih). Tidak kompatibel dengan *type* b dan g.

- 802.11g

  Digunakan pada pertengahan tahun 2003 dengan mengunakan frekuensi 2, 4 GHz, maksimum bandwidth yang bisa di capai 54 Mbps , modulas i sinyal yang di gunakan adalah OFDM. Kanal yang tidak overlapping 3. Kompatibel dengan *type* b namun hasilnya mengikuti *type* b.

- 802.11a/g

  Digunakan mulai pertengahan tahun 2003 dengan menggunakan frekuensi 2,4 GHz dan 5 Ghz, maksimum bandwidth yang bisa di capai 54 Mbs, modulasi sinyal yang di gunakan OFDM. Kanal yang tidak overlapping 16. Bila jalan pada modus a tidak kompatibel dengan *type* b dan g. Bila jalan pada modus g kompatibel dengan *type* b. Disamping itu ada juga standar *type* 802.11e yang mem punyai kelebihan pada se curity, dan *type* 802.11n yang bisa mencapai kecepatan 100 - 320 Mbps.

Jarak yang bisa di capai dengan WLAN bisanya bisa sampai puluhan meter (indor) dan sampai ratusan meter (kilo meter) (outdor) tergantung jenis dan merek , penguat dan antenna yang di gunakan.

Gambar 2.11 Access Point[8]

BAB III

ANALISA DAN PERANCANGAN SISTEM

3.1 Analisa Sistem

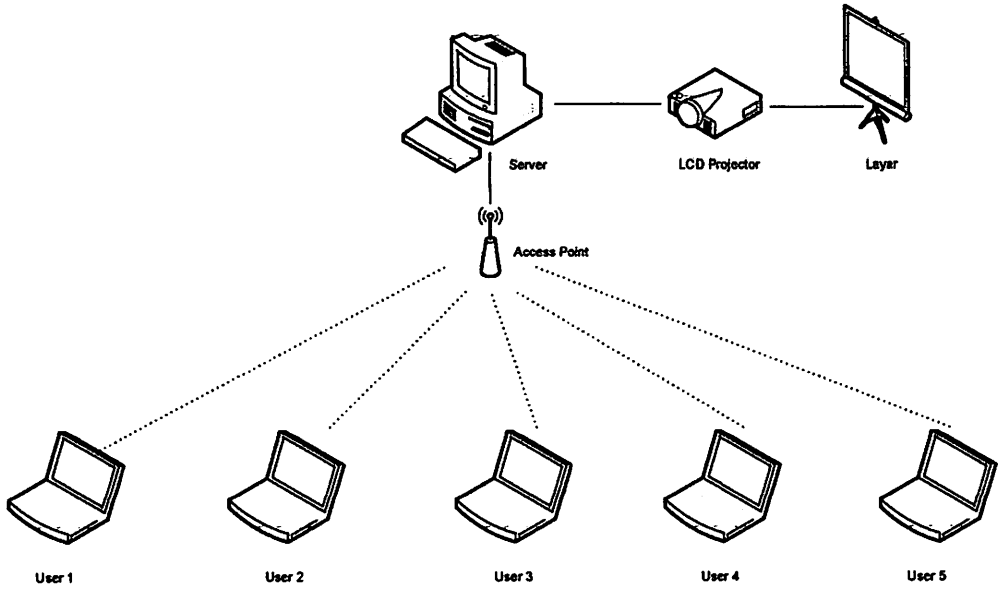
Untuk dapat mendesain sebuah aplikasi ini diperlukan pemahaman dasar agar dapat mempermudah didalam merancang aplikasi *screen capture* yang akan dibuat. Aplikasi ini dirancang untuk dapat mempermudah didalam menampilkan tampilan desktop dari beberapa komputer *client* pada LCD *projector* dengan tanpa memindah kabel koneksi antara *client* dengan perangkat LCD *projector* tersebut. Untuk itu diperlukan sebuah aplikasi berbasis *client server* dengan menggunakan media nirkabel atau *wireless.*

Dalam merancang sebuah aplikasi *screen capture* ini, akan dibagi menjadi dua sub sistem, yaitu subsistem *server* dan subsistem *client.* Subsistem *server* adalah aplikasi yang dijalankan pada komputer *server* yang digunakan untuk menerima hasil dari *screen capture* komputer *client* dan menampilkannya pada perangkat LCD monitor yang terintegrasi dengan komputer *server* tersebut. Sedangkan subsistem *client* adalah aplikasi yang berjalan pada komputer *client* yang akan mengirimkan *screen capture* dari tampilan desktop komputer *client* itu sendiri.

Pada dasarnya jika sebuah aplikasi berbasis *client server* ini berjalan pada dua komputer yang akan melakukan pertukaran data atau informasi, maka diperlukan protokol yang bertugas untuk mengatur bagaimana komunikasi antar komputer tersebut.

Untuk itu dalam aplikasi ini digunakan protokol TCP/IP. TCP merupakan protokol yang berperan untuk mengatur paket – paket yang akan ditransmisikan melalui jaringan dan diterima oleh TCP *layer* yang akan menyusun paket – paket tersebut kedalam bentuk yang sebenarnya. Sedangkan IP merupakan protokol yang membawa informasi tersebut dari mana dan harus kemana disampaikan. Dengan memanfaatkan protokol tersebut maka hubungan komunikasi antara dua buah komputer dapat dilakukan, sehingga dapat mendukung berjalannya aplikasi *screen capture* ini.

3.2 Desain Sistem



Gambar 3.1 Desain Sistem Aplikasi Screen Capture Komputer Client

Pada gambar diatas, komputer *user* atau *client* terhubung ke *server* dengan menggunakan perangkat WIFI (*Wireless Fidelity*) sebagai penghubungnya. Perangkat WIFI yang digunakan disini adalah AP atau *Acces Point*. AP berfungsi sebagai *switch* untuk pengguna ataupun perangkat *wireless* agar dapat terkoneksi ke jaringan, sehingga dapat mempermudah *user* untuk menampilkan tampilan *desktop* kedalam perangkat LCD *projector* dengan tanpa memasang atau memindah kabel penghubung antara komputer dengan perangkat LCD *Projector*.

Untuk dapat menampilkan tampilan *desktop* komputer *client* ke perangkat LCD *projector* yang telah terhubung dengan komputer *server*, maka pada komputer *server* dan *client* harus dalam kondisi siap untuk melakukan proses pengiriman dan penerimaan *screen capture*. Pada aplikasi disisi *client*, saat aplikasi dijalankan, maka aplikasi ini akan menampilan nama dan nomor IP dari komputer *client* yang nantinya akan digunakan oleh komputer *server* untuk dapat terkoneksi dengan komputer *client*. Pada saat yang bersamaan, aplikasi *client* juga telah siap untuk mengirimkan *screen capture* ke aplikasi disisi *server*. Setelah nomor IP komputer *client* dimasukkan ke aplikasi disis *server* maka *server* akan terhubung dengan *client* dan dapat melakukan proses penerimaan *screen capture* dari komputer *client*. Selanjutnya perangkat LCD *projector* yang telah terhubung dengan komputer *server* akan menanpilkan tampilan desktop komputer *client* pada layar. Aplikasi ini dirancang dengan menggunakan sistem *connect-disconnect*, yaitu hanya *client* yang akan ditampilkan ke LCD *projector* itu saja yang akan terhubung dengan *server*.

Sedangkan *client* yang lainnya akan *disconnect* dengan *server*. Dengan adanya sistem *connect-disconnect* tersebut, maka aplikasi ini akan dengan mudah mengatur *client* mana saja yang akan terhubung dengan *server* dan menampilkannya pada perangkat LCD *projector*.

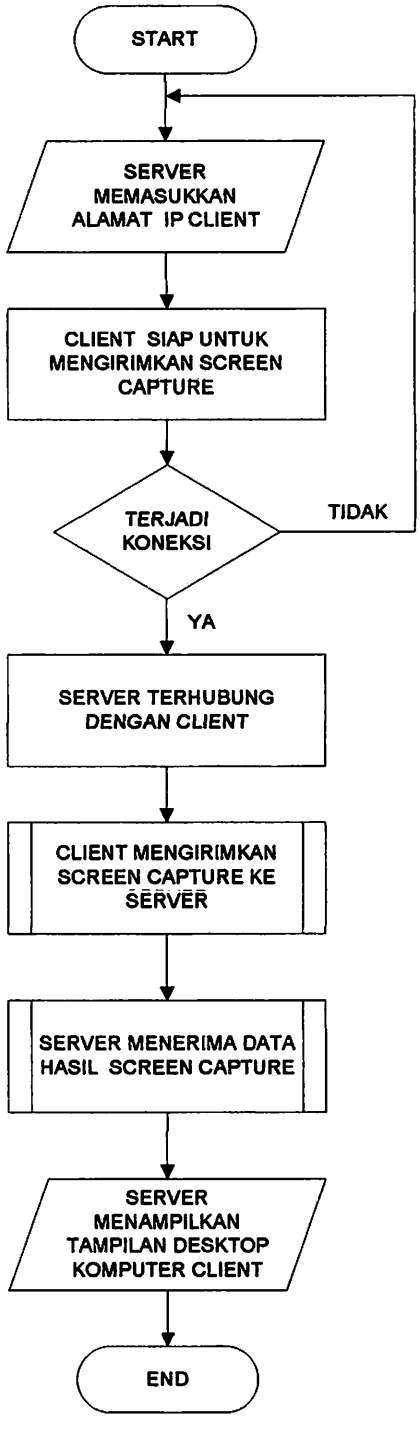
3.3 Pembuatan Algoritma Sistem

Pembuatan algoritma ini digunakan untuk menjelaskan mekanisme kerja dari desain aplikasi *screen capture* yang akan dibuat.

Susunan *algoritma*nya adalah sebagai berikut:

1. Aplikasi *client* akan *running* pada komputer *client,* sedangkan aplikasi *server* akan *running* pada komputer *server.*
2. Aplikasi *server* akan memasukkan nomor IP dari komputer *client.*
3. Aplikasi *server* melakukan pengecekan, apakah *client* aktif
   - Jika aktif, akan lanjut ke no 4
   - Jika tidak aktif, kembali ke no 2
4. Aplikasi *client* telah siap untuk mengirimkan *screen capture*
5. Aplikasi *client* mengirimkan *screen capture* ke aplikasi *server*
6. Aplikasi *server* menampilkan tampilan *desktop* dari komputer *client*

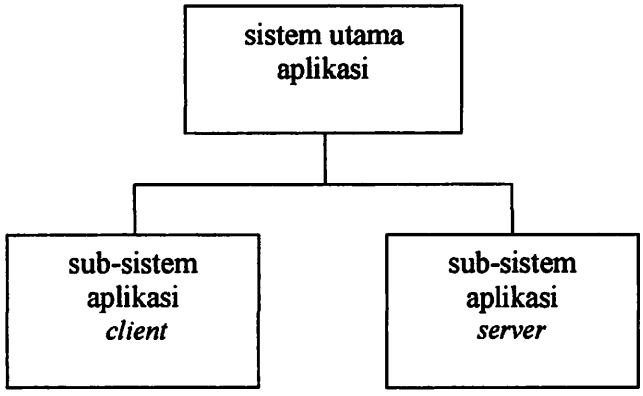
Dari algoritma diatas, maka jika direpresentasikan ke dalam sebuah diagram alir akan tampak sebagai berikut :



Gambar 3.2: Diagram Alir Aplikasi

3.4 Perancangan Aplikasi

Dalam perancangan aplikasi ini akan dibagi menjadi dua sub-sistem, yaitu subsistem aplikasi *client* dan subsistem aplikasi *server*.



Gambar 3.3 Desain Sistem Utama Aplikasi

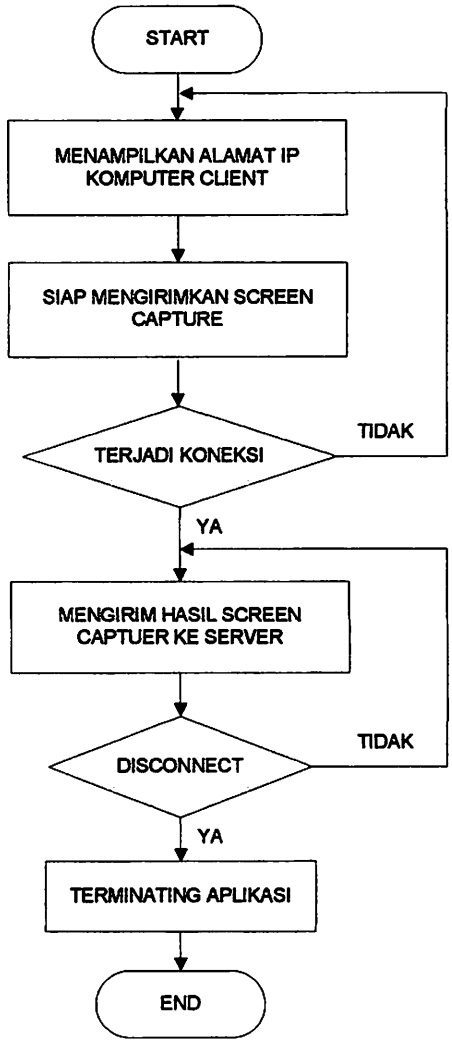
Subsistem aplikasi *client* merupakan aplikasi yang akan *running* pada komputer *client* dan aplikasi ini bertugas untuk mengirimkan hasil dari *screen capture* komputer *client*, sedangkan subsistem aplikasi *server* merupakan aplikasi yang akan *running* pada komputer *server*, dan aplikasi ini nantinya yang akan menerima hasil dari *screen capture* yang dikirimkan oleh aplikasi *client*.

3.5 Flowchart

Sesuai dengan desain sistem yang direncanakan, maka dapat dibuat diagram alir dari aplikasi untuk menampilkan *screen capture* komputer *client* pada perangkat LCD *projector* ini.

1. Flowchart aplikasi *client.*

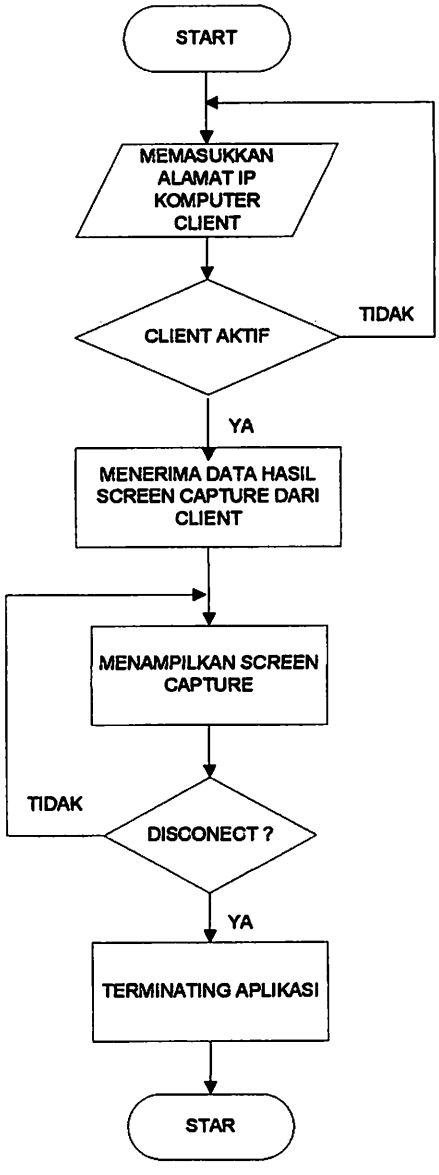
   Flowchart ini menggambarkan suatu aplikasi pada sisi *client* yang merupakan aplikasi untuk mengirimkan *screen capture* komputer *client.*



Gambar 3.4 Flowchart Aplikasi Client

2. Flowchart Aplikasi *Server*

Flowchart ini menggambarkan mekanisme suatu aplikasi pada sisi *server* yang merupakan aplikasi yang digunakan untuk menerima hasil *screen capture* yang dikirim oleh aplikasi yang berjalan pada komputer *client.*



Gambar 3.5 Flowchart Aplikasi Server

3.6 Perancangan Pada Subsistem Aplikasi

3.6.1. Perancangan Pada Subsistem Aplikasi Client

Perancangan aplikasi pada subsistem *client* merupakan satu form utama yang didalamnya berisi beberapa proses. Rangkaian proses tersebut terdiri dari proses *startup* untuk proses siap mengirimkan *screen capture*, proses penyettingan *client port*, proses menampilkan status koneksi, dan proses menampilkan log.

3.6.1.1. Startup

Aplikasi *client* akan melakukan startup pada saat pertama kali dijalankan. Kemudian *form client* akan langsung melakukan proses *minimize* supaya *form* tidak begitu mengganggu tampilan pada tampilan *client* yang akan ditampilkan ke *server*

3.6.1.2. Pengaturan Port Client.

Port digunakan untuk memberikan alamat koneksi pada suatu komputer. Dengan kata lain, jika alamat IP digunakan untuk memberikan alamat pada komputer yang berbeda, port digunakan untuk memberikan alamat pada komputer yang sama. Port disini di *setting* awal dengan nilai 8886, hal ini disebabkan karena port dengan nilai dibawah 1024 biasanya tidak boleh untuk digunakan karena telah ada standart untuk masing-masing nomor. Sedangkan *range port* antara 1024 – 49151 merupakan *range port* yang bebas dipilih untuk dapat digunakan untuk aplikasi.

3.6.1.3. Menampilkan Status Koneksi

Untuk mempermudah dalam mengetahui status koneksi dengan *server* serta mengetahui seberapa lama koneksi yang terjadi, maka *client* akan menampilkan status koneksi, waktu mulai koneksi.

3.6.2. Perancangan Pada Subsistem Aplikasi Server

Sub sistem aplikasi *server* merupakan satu form utama yang di dalamnya terdapat beberapa proses. Adapaun beberapa proses tersebut antara lain proses dimana *server* telah siap menerima *screen capture* dari *client,* mengakses *drive-drive* yang tersedia dalam komputer *client* serta melakukan *file transfer.*

3.6.2.1. Menampilkan Screen Capture

Dalam proses menampilkan *screen capture* pada aplikasi *server* ini, aplikasi harus *running* sebagai aplikasi *server* terlebih dahulu. Kemudian ketika *server* telah memasukkan IP komputer *client* dan terjadi koneksi, maka aplikasi ini akan menampilkan hasil *screen capture* yang dikirimkan oleh aplikasi yang berada pada sisi *client.* Dalam menampilkan tampilan desktop komputer *client* pada form aplikasi *server* ini, nantinya juga dapat menampilkan dalam bentuk *full screen* yang artinya tampilan *screen capture* komputer *client* akan ditampilkan pada aplikasi *server* dalam layar penuh, sehingga pada perangkat LCD *projector* pun akan menampilkan tampilan desktop komputer *client.*

3.6.2.2. Melakukan File Transfer

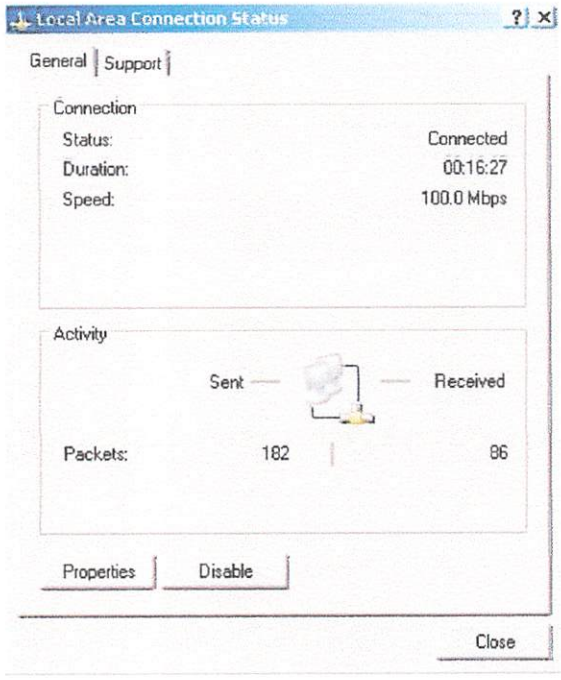
Aplikasi *server* ini nantinya akan dapat melalukan *file transfer* dengan aplikasi yang berada pada sisi *client.* Sehingga bilamana komputer *client* akan melakukan pemindahan data dengan komputer *server* maka dapat dilakukan dengan aplikasi ini.

3.7. Konfigurasi Access Point

Untuk dapat mengkoneksikan jaringan antar *client* dan *server* secara *wireless* maka harus terlebih dahulu menginstall dan mengkonfigurasi *Access Point* nya. *Acces Point* sendiri adalah perangkat keras yang berfungsi sebagai *switch* untuk pengguna ataupun perangkat *wireless* agar dapat terkoneksi ke dalam suatu jaringan komputer.

Proses pertama adalah dengan memberi nama komputer pada *server* untuk memastikan bahwa komputer yang dipakai dapat dikenali oleh pemakai komputer lain yang terhubung di dalam jaringan komputer. Komputer harus menggunakan nama yang unik dan berbeda dengan komputer lain supaya tidak terjadi adanya tumpang-tindih dengan komputer lainnya.

Setelah *access point* dikoneksikan ke komputer *server* maka *Local Area Connection Status* akan menampilkan status *connected*, hal ini berarti access *point* sudah dalam keadaan tekoneksi ke komputer *server.*



Gambar 3.7 LAN Status setelah terkoneksi dengan Access Point

Setelah *access point* terkoneksi dengan komputer *server*, access point akan secara otomatis memberi nomor IP *Address* pada komputer *server*. Nomor IP tersebut kemudian di ping dari komputer *client* ke komputer *server* maupun dari komputer *server* ke komputer *client*. Hal in berfugsi untuk mengetahui Access Point apakah komputer *client* dan *server* sudah terkoneksi atau belum. Jika data bisa terkirim dan terjadi reply maka kedua komputer tersebut sudah terkoneksi dalam satu jaringan.

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi Sistem

Implementasi dilakukan dengan menerapkan hasil desain yang telah dibuat ke dalam bahasa pemrograman (*Coding*) Borland Delphi 7, sehingga prosedur-prosedur yang telah dibuat dapat dimengerti oleh mesin sehingga menghasilkan keluaran seperti yang diharapkan. Aplikasi ini merupakan hasil implementasi yang telah dilakukan, berikut ini penjelasan bagian-bagian dari aplikasi tersebut yang dibagi kedalam dua aplikasi yaitu *Client* dan *Server*.

4.1.1 Aplikasi Screen Capture

Aplikasi *screen capture* ini merupakan aplikasi yang berjalan dengan mekanisme *client server*, sehingga aplikasi ini menggunakan dua aplikasi, yaitu aplikasi yang berjalan pada sisi *server* dan aplikasi yang berjalan pada sisi *client*. Meskipun aplikasi ini terbagi menjadi dua yaitu aplikasi *client* dan aplikasi *server*, namun aplikasi ini dibuat untuk menjadi satu, sehingga memudahkan *user* untuk menggunakan aplikasi ini untuk diaktifkan sebagai *client* ataupun diaktifkan sebagai *server*. Selain itu dengan mekanisme satu aplikasi tersebut maka pada saat pemasangan perangkat LCD *projector* pada komputer tidak akan terjadi kendala, karena jika aplikasi ini dijalankan pada komputer yang terhubung dengan perangkat LCD *projector*, maka pasti aplikasi *server* yang akan dijalankan.

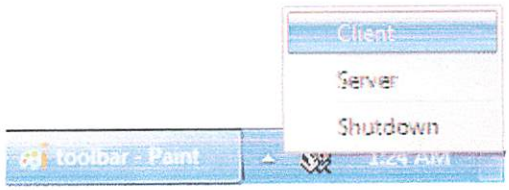
4.1.1.1 Aplikasi Client

Aplikasi *client* merupakan aplikasi yang berjalan pada komputer *client.* Aplikasi ini akan mengirimkan *screen capture* komputer *client* pada aplikasi yang berada pada komputer *server.* Mekanisme pengiriman hasil *screen capture* ini dilakukan dengan membuka *port* koneksi. Berikut ini adalah implementasi mekanisme tersebut ke dalam bahasa pemrograman Delphi 7.

4.1.1.1.1 Form Utama Client

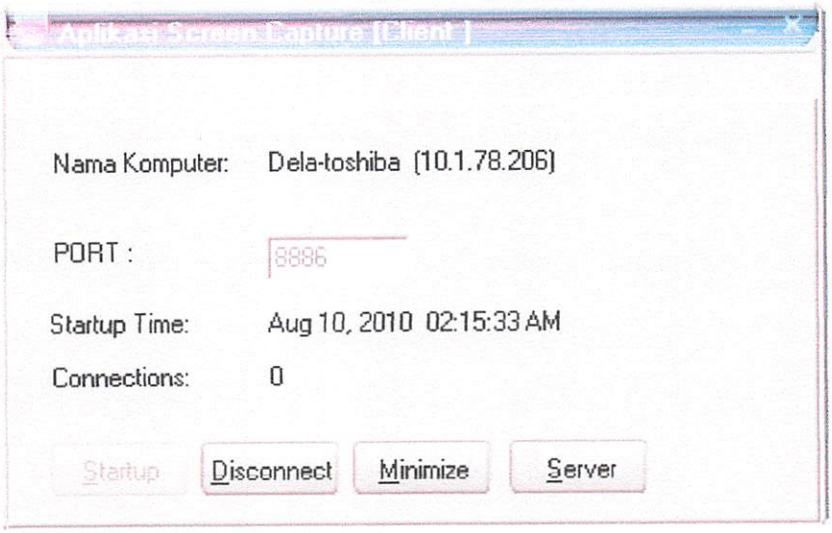
Pada saat aplikasi *screen capture* ini *running* untuk pertama kalinya, maka yang akan aktif adalah aplikasi *client* terlebih dahulu, kemudian form *client* dari aplikasi *screen capture* ini akan *minimize* secara otomatis dan pada *taskbar* akan terdapat *icon* dari aplikasi *screen capture* ini. Selanjutnya pada *icon* aplikasi ini, jika diklik kanan maka akan muncul pilihan apakah aplikasi ini akan digunakan sebagai aplikasi *client* atau akan diaktifkan sebagai aplikasi *server.*

Gambar 4.1 Icon Aplikasi Pada Taskbar



Gambar 4.2 Pemilihan Aplikasi

Selanjutnya jika diklik kanan pada *icon* aplikasi yang ada pada *taskbar* akan muncul pilihan untuk mengaktifkan aplikasi *screen capture* ini sebagai aplikasi *client* atau sebagai aplikasi *server.* Dan jika diklik *client* maka aplikasi *client* akan aktif dan form utama *client* akan aktif.



Gambar 4.3 Form Client

Pada form *client* tersebut terdapat *textbox* untuk mengisikan nomor *port* yang dipakai, nomor *port* yang diisikan haruslah sesuai dengan nomor *port* pada aplikasi *server*. Form tersebut juga akan menampilkan nama dan alamat IP dari komputer *client*, waktu ketika aplikasi ini dijalankan. Dalam form ini juga terdapat beberapa tombol, antara lain :

- Tombol Server

  Tombol ini digunakan apabila aplikasi *screen capture* ini akan digunakan sebagai aplikasi *server*.

- Tombol Minimize

  Tombol *minimize* ini digunakan untuk me-*minimize* form *client* agar tidak menggangu tampilan *desktop* computer *client*.

- Tombol Disconnect

  Digunakan untuk memutuskan koneksi dengan aplikasi *server*. Pada saat tombol ini diklik, maka tombol *start* akan *enable*, dan *textbox* untuk mengisikan nomor *port* akan aktif.

- Tombol Startup

  Tombol ini digunakan untuk siap memulai koneksi dengan aplikasi *server*. Pada saat form *client* pertama kali *running*, tombol ini diatur untuk tidak aktif (*disable*).

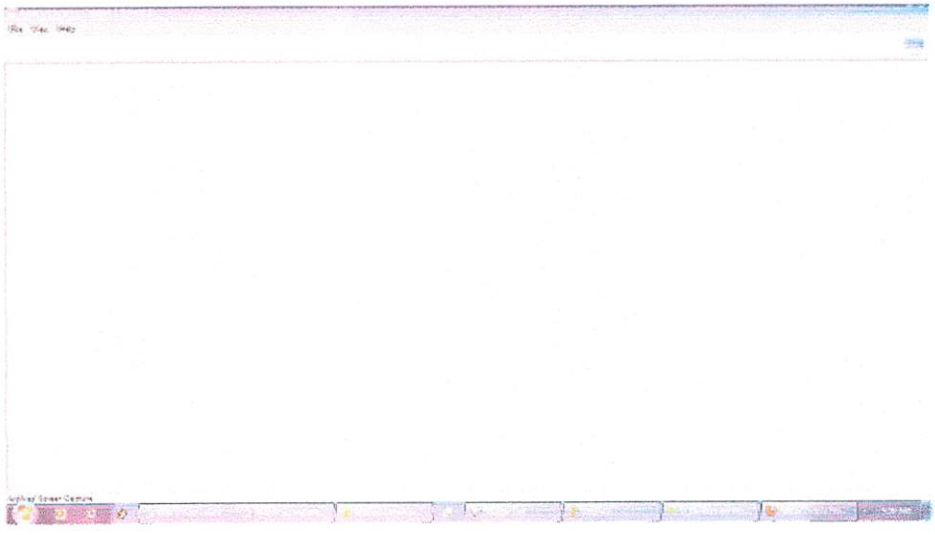
Kemudian dalam beberapa detik, form *client* akan *minimize* secara otomatis dan berada pada *taskbar,* dan pada saat diklik kanan maka akan tampil pilihan untuk mengaktifkan sebagai aplikasi *client* ataupun sebagai aplikasi *server* atau ingin mengakhiri aplikasi dengan memilih *shutdown.*

4.1.1.2. Aplikasi Server

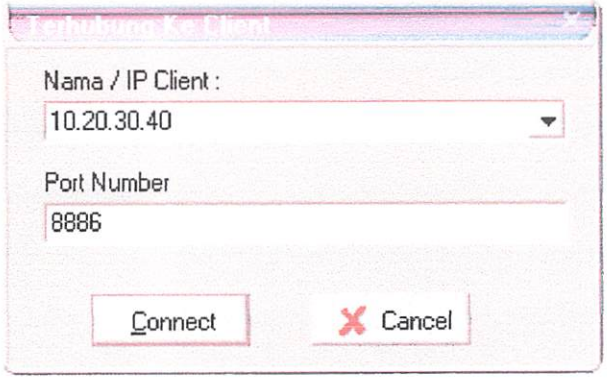
Seperti halnya dengan aplikasi *client,* aplikasi *server* ini merupakan aplikasi yang berjalan pada komputer *server.* Aplikasi ini digunakan untuk menerima hasil *screen capture* yang dikirimkan oleh aplikasi *client.* Dalam aplikasi ini ada beberapa form, antara lain :

4.1.1.2.1 Form Server

Merupakan form untuk menampilkan tampilan *desktop* komputer *client.* Sebelum menerima *screen capture* dari *client,* dalam form *server* terdapat sebuah *dialogbox* yang merupakan form untuk mengetik alamat IP komputer *client* serta nomor *port* yang digunakan. Sehingga antara aplikasi *server* dan aplikasi *client* dapat terkoneksi dan dapat mengirim serta menerima *screen capture.* Berikut adalah tampilan dari form *server.*



Gambar 4.4 Form Server

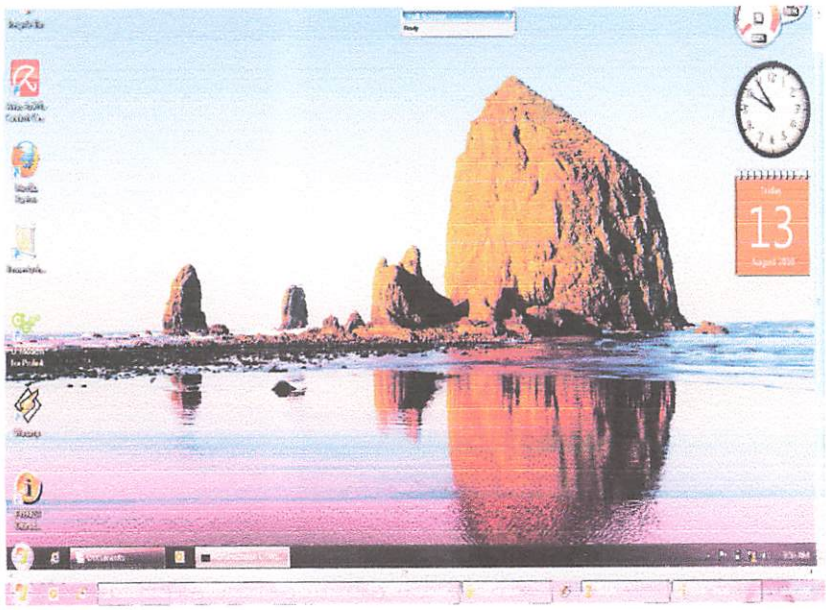


Gambar 4.5 Form Koneksi Ke Client

Berikut adalah simulasi untuk dapat menampilkan tampilan *desktop* computer *client* pada form *server,* yang mana selanjutnya tampilan ini juga yang akan ditampilkan pada layar LCD *projector.*

4.1.1.2.2 Form Full Screen

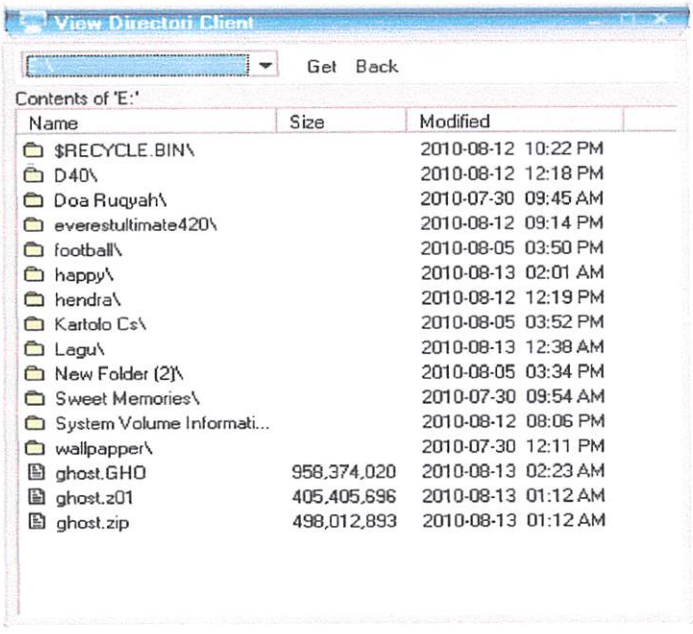
Digunakan untuk menampilkan tampilan *desktop* komputer *client,* namun dalam bentuk *fullscreen,* sehingga apa yang ada pada *desktop* komputer *client* akan ditampilkan pada *desktop* komputer *server* tanpa terganggu oleh menu dan *toolbar* dari form *server.*



Gambar 4.7 Form Full Screen

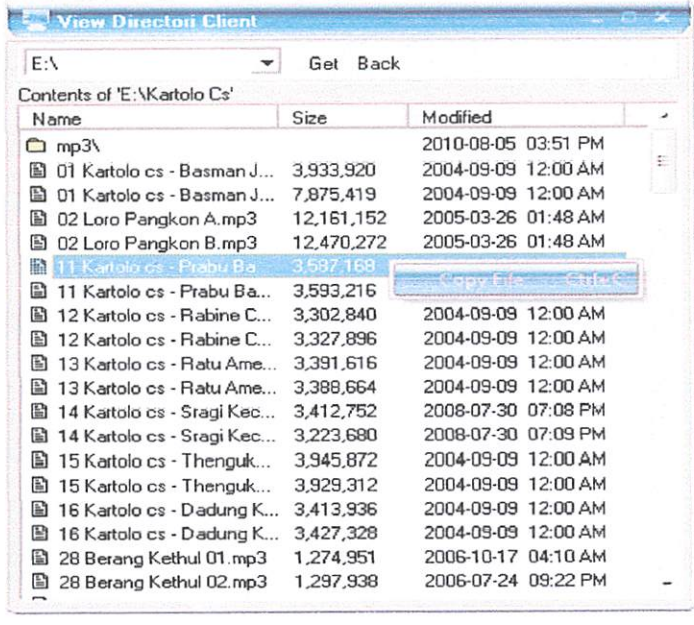
4.1.1.2.3 Form View Directori

Form ini digunakan untuk melihat dan melakukan *transfer file* dari *diretori* komputer *client* ke *directori* komputer *server.*



Gambar 4.7 Form View Directori

Kemudian jika akan melakukan *transfer file* maka dapat dilakukan dengan melakukan klik kanan dan akan muncul *option copy file*, kemudian simpan.



Gambar 4.8 Copy File Dari Directori Client

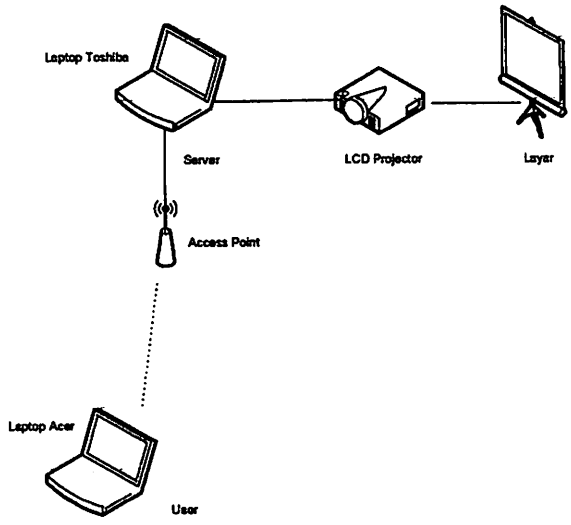
4.2 Pengujian Sistem

Pengujian ini dilakukan untuk mengetahui kinerja aplikasi *screen capture* pada beberapa komputer yang berbeda dengan spesifikasi *hardware* yang digunakan pada pengujian adalah sebagai berikut:

Tabel 4.1. Hardware pengujian.

NO	Hardware	Spesifikasi	Keterangan
1.	Access Point	D-Link	
2.	Laptop Toshiba	Processor	Core 2 duo 2,10 GHz
		Memori	2 Gb DDR2
		Hardisk	250 Gb SATA
		Operating System	Windows 7
3.	Laptop Acer	Processor	Dual core 2 GHz
		Memori	1 Gb DDR2
		Hardisk	300 Gb SATA
		Operating System	Windows XP

Desain dari *hardware* diatas yang digunakan pada saat pengujian adalah sebagai berikut:



Gambar 4.9 Desain Sistem Pengujian

4.2.1. Berdasarkan Aplikasi Yang Dijalankan.

Berikut ini adalah hasil pengujian yang telah dilakukan pada beberapa aplikasi yang telah dilakukan untuk mengetahui kinerja dari aplikasi *screen capture* ini untuk menampilkan tampilan *desktop* computer *client* pada computer *server* dan ke perangkat LCD *projector*. Dan hasil yang dipaparkan berikut adalah hasil dari aplikasi yang berada pada computer *server*.

Tabel 4.2. Aplikasi Untuk Pengujian

NO	Software	Delay (detik)	Keterangan
1	Microsoft Power Point 2007	1,03	Untuk menampilkan slide presentasi
2	Windows Media Player 9 Series	1,44	Untuk menampilkan animasi
3	Microsoft Word 2007	0,73	Untuk menampilkan ketikan pada lembar kerja

4.2.2. Penggunaan Memori dan CPU

Untuk mengetahui pengaruh aplikasi ini terhadap kerja computer yang digunakan untuk menjalankannya, maka dilakukan pengujian terhadap penggunaan memori fisik maupun kinerja dari CPU. Tabel berikut akan menjelaskan hasil pengujian pada computer *client.*

Tabel 4.3. Pengujian Terhadap Kinerja Komputer Client

NO	Spesifikasi Aplikasi Pengujian	Penggunaan CPU (%)	Penggunaan Memori (Kb)
1.	Keadaan Normal	1	3,25
1	Microsoft Power Point 2007	49	8,07
2	Windows Media Player 9 Series	51	8,58
3	Microsoft Word 2007	47	8,26

Pengujian selanjutnya adalah pada computer *server*, untuk hasil pengujiannya akan dijelaskan pada table berikut :

Tabel 4.3. Pengujian Terhadap Kinerja Komputer Client

NO	Spesifikasi Aplikasi Pengujian	Penggunaan CPU (%)	Penggunaan Memori (Kb)
1.	Keadaan Normal	1	3,25
2.	Aplikasi Screen Capture	1	3,88
3.	Microsoft Power Point 2007	49	8,07
4.	Windows Media Player 9 Series	51	8,58
5.	Microsoft Word 2007	47	8,26

Pengujian selanjutnya adalah pada computer *server*, untuk hasil pengujiannya akan dijelaskan pada table berikut :

Tabel 4.4. Pengujian Terhadap Kinerja Komputer Server

NO	Spesifikasi Aplikasi Pengujian	Penggunaan CPU (%)	Penggunaan Memori (Kb)
1.	Keadaan Normal	3	12,30
2.	Aplikasi Screen Capture	3	7,20
3.	Microsoft Power Point 2007	22	18,13
4.	Windows Media Player 9 Series	25	11,94
5.	Microsoft Word 2007	18	12,42

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari perancangan serta pengujian yang dilakukan, maka didapatkan beberap kesimpulan sebagai berikut :

1. Mekanisme pengiriman *screen capture* ini dapat berjalan jika kedua aplikasi yaitu aplikasi *client* dan aplikasi *server* aktif.
2. Untuk menampilkan tampilan *desktop* komputer *client* cukup dengan mengetik alamat IP komputer *client,* tanpa harus memindah kabel koneksi antara komputer dengan perangkat LCD *projector.*
3. Waktu yang digunakan untuk menampilkan tampilan *desktop* komputer *client* pada perangkat LCD *projector* lebih efisien.
4. Lingkup dari desain dan implementasi aplikasi ini hanya pada jaringan komputer LAN *(Local Area Network).*
5. Aplikasi *screen capture* ini dibuat dengan bahasa pemrograman Borland Delphi 7 dan menggunakan system operasi berbasis windows.
6. Berdasarkan hasil dari pengujian aplikasi yang dilakukan, maka dapat disimpulkan bahwa dalam menjalankan aplikasi *screen capture* ini pemakaian CPU (dalam %) dan pemakaian memori (dalam Kb) sangat beragam. Besar kecilnya pemakaian CPU dan penggunaan memori tergantung dari aplikasi yang dijalankan dari komputer *clinet* yang nantinya

akan ditampilkan pada computer *server* dan selanjutnya ditampilkan pada perangkat LCD *projector.* Adapun penggunaan CPU dan penggunaan memori dari komputer *client* dan komputer *server* untuk menjalankan beberapa apliasi adalah sebagai berikut:

- Pada Komputer *Client*

  Dalam keadan normal dimana komputer tidak menjalankan aplikasi, didapat penggunaan CPU adalah 1 % dan penggunaan memori adalah 3,25 Kb, kemudian ketika aplikasi *screen capture* ini dijalankan penggunaan CPU adalah 1 % dan penggunaan memori adalah 3,38 Kb . Selanjutnya ketika aplikasi ini digunakan untuk menampilkan Microsoft Power Point 2007, didapat penggunaan CPU = 49 % dan penggunaan memori = 8,07 Kb. Kemudian untuk menampilkan Windows media player 9, penggunaan CPU = 51 % dan memori memerlukan 8,58 Kb. Selanjutnya untuk menampilkan Microsoft Word 2007 penggunaan CPU adalah 47 % dan menggunakan memori sebesar 8,26 Kb.

- Pada Komputer *Server*

  Dalam keadan normal dimana komputer tidak menjalankan aplikasi, didapat penggunaan CPU adalah 3 % dan penggunaan memori adalah 12,30 Kb, kemudian ketika aplikasi *screen capture* ini dijalankan penggunaan CPU adalah 3 % dan penggunaan memori adalah 7,20 Kb . Selanjutnya ketika aplikasi ini digunakan untuk menampilkan Microsoft Power Point 2007, didapat penggunaan CPU = 22 % dan penggunaan memori = 18,13 Kb.

Kemudian untuk menampilkan Windows media player 9, penggunaan CPU = 25 % dan memori memerlukan 11,94 Kb. Selanjutnya untuk menampilkan Microsoft Word 2007 penggunaan CPU adalah 18 % dan menggunakan memori sebesar 12,42 Kb.

5.2 Saran

Berikut adalah saran yang diberikan untuk pengembangan aplikasi ini selanjutnya.

1. Perlu dikembangkan aplikasi sejenis yang memiliki kompabilitas tinggi terhadap beberapa system operasi, sehingga dapat berjalan antar system operasi (*multiplatform*).
2. Tampilan visualisasi dapat dibuat lebih atraktif, sehingga tampilan akan lebih menarik.
3. Bermanfaat sebagai referensi untuk pengembangan pada generasi yang akan datang.

Daftar Pustaka

[1]. Parasimax, *TCP/IP* Part I

[2]. *Jaringan wireless LAN*, http://aboen.atekbl.com

[3]. *Zarko's Delphi Programming Blog*,
http://delphi.about.com/b/2005/05/05/doing-screen-shots-with-delphi.htm

[4]. Modul Pemrograman Jaringan, Laboratorium Jaringan Komputer ITN Malang,2010.

[5]. Pemrograman Socket Jaringan j3ck3y.wordpress.com/2008

[6]. Penerbit Andi, *Tip Dan Trik Pemrograman Delphi 7.0*, Wahana Komputer.

[7]. Martina, Inge,*Database Client Server Menggunakan Delphi*, Elex Media Komputindo, 2002.

[8]. Wahana Komputer, *Pemrograman Delphi 7*, Penerbit Andi, 2005

[9]. Setiawan, Yudha, *Trik dan Tip Delphi*, Penerbit Andi, 2003.

L A M P I R A N

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAM TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA	:	FEBRY DANANG EKA HERMAWAN
NIM	:	05.12.513
JURUSAN/KONSENTRASI	:	TEKNIK ELEKTRO / T. KOMPUTER DAN INFORMATIKA S-1
JUDUL SKRIPSI	:	**DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR**

Dipertahankan dihadapan penguji skripsi jenjang program starata satu (S-1) pada :

Hari/Tanggal : Selasa 24 Agustus 2010

Dengan nilai : 88,5 (A)

PANITIA UJIAN SKRIPSI

Mengetahui,
Ketua Majelis Penguji

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

ANGGOTA PENGUJI

Dosen Penguji I

Dr. Eng.Aryuanto Soetedjo,ST MT
NIP.Y. 1030800417

Dosen Penguji II

Irmalia S Faradisa, ST MT
NIP.Y. 1030000365

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAM TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

FORMULIR PERBAIKAN SKRIPSI

lama : **FEBRY DANANG EKA HERMAWAN**
lim : **05.12.513**
urusan : **Teknik Elektro S-1**
onsentrasi : **Teknik Komputer dan Informatika S-1**
lasa Bimbingan : **30 Maret s/d 30 September 2010**
udul Skripsi : **DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR**

Tanggal	Penguji	Uraian	Paraf
	I	1. Pengujian serta pemakaian CPU dan memory antara aplikasi screen capture dengan aplikasi lainnya.	
24 Agustus 2010	II	1. Tabel pada BAB IV. 2. Kesimpulan.	

Disetujui,

Dosen Penguji I

Dr.Eng.Aryuanto Soetedjo,ST, MT
NIP. Y. 1030800417

Dosen Penguji II

Irmalia S Faradisa, ST, MT
NIP. Y. 1030000365

Mengetahui,

Dosen Pembimbing I

I Komang Somawirata, ST, MT
NIP. Y. 1030100361

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : FEBRY DANANG . EH
NIM : 05 12513
Perbaikan meliputi :

Tabel bab IV
kesimpulan.

Malang, 24 - 08 2010

(Irmalia SF.)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Feby Danang
NIM : 0512513
Perbaikan meliputi :

- Pengujian pemakaian CPU & memory antara aplikasi dengan Office atau lainnya, sehingga bisa diketahui pemakaian CPU oleh aplikasi yg dibuat.

Malang, 27/8/2010

( )

Aryanto

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, Km. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

N a m a	: FEBRY DANANG E.H
N I M	: 05.12.513
Semester	: 10
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: ~~TEKNIK ELEKTRONIKA~~ ~~TEKNIK ENERGI LISTRIK~~ **TEKNIK KOMPUTER DAN INFORMATIKA**
Alamat	: JL. SENGGANI NO 33 MALANG

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana.*** Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (...)
2. Telah lulus dan menyerahkan Laporan Praktek Kerja (...)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (...)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (...)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan Jurusan (...)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(Putri Kranda Yani)

Malang, 19 FEB 2010
Pemphon

(FEBRY DANANG E.H)

Disetujui
Ketua Jurusan Teknik Elektro

Ir. F. Yudi Limpraptono, MT
NIP. P. 1039500274

Mengetahui
Doseh Wali

(Dr. CAHYO CRISDIAN)

Catatan :
Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Jurusan/Sekretaris Jurusan T. Elektro S-1

1. 407.5/138 = 2.95
2. .....
3. - 3 praktikum

Form. S-1a

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika/Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: FERRY DWANG E.H			Nim: 0512513
2.	Waktu Pengajuan	Tanggal:	Bulan:	Tahun:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen*) I. Komang Somawirata, ST MT	Ketua Jurusan Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274		
5.	Judul yang diajukan mahasiswa:	PENGEMBANGAN APLIKASI REMOTE DESKTOP WIRELESS CONNECTION UNTUK MENAMPILKAN SCREEN CAPTURE PADA LCD PROJECTOR		
6.	Perubahan judul yang disetujui Dosen sesuai materi bidang ilmu	DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE CLIENT KE LCD PROJECTOR		
7.	Catatan: judul masukan di sekretariat			
	Persetujuan Judul skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu	Disetujui 11 MARET 2010 Dosen I. Komang Somawirata, ST MT		

Perhatian:

1. Formulir pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-2

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

Lampiran : 1 (satu) berkas
Pembimbing Skripsi

Kepada : Yth. I Komang Somawirata, ST MT
Dosen Institut Teknologi Nasional
MALANG

Yang bertanda tangan di bawah ini:

Nama	: FEBRY DANANG EKA HERMAWAN
Nim	: 05.12.513
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Komputer & Informatika S-1

Dengan ini mengajukan permohonan, kiranya Bapak / Ibu bersedia menjadi Dosen Pembimbing Utama / ~~Pendamping~~ *), untuk penyusunan Skripsi dengan judul (proposal terlampir) :

DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik.
Demikian permohonan ini kami buat.

Malang, Maret 2010

Mengetahui

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

Hormat Kami,

Febry Danang Eka H
05.12.513

Catatan :
***) Coret yang tidak perlu**

Form S-3 a

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

PERNYATAAN KESEDIAAN DALAM PEMBIMBING SKRIPSI

Sesuai permohonan dari Mahasiswa :

Nama	: FEBRY DANANG EKA HERMAWAN
Nim	: 05.12.513
Semester	: X (Sepuluh)
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Komputer & Informatika S-1

Dengan ini menyatakan bersedia / tidak bersedia *) Membimbing Skripsi dari mahasiswa tersebut, dengan judul :

DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR

Demikian surat pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang, Maret 2010

Kami yang membuat pernyataan,

I Komang Somawirata, ST MT

NIP.Y. 1030100361

Catatan :
Setelah disetujui formulir
Diserahkan mahasiswa/ I yang bersangkutan
Kepada Jurusan untuk diproses lebih lanjut
*) Coret yang tidak perlu

Form S-3 b

JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S1

KONSENTRASI	TEKNIK INFORMATIKA DAN KOMPUTER S-1

1.	Nama Mahasiswa	FEBRY DANANG EKA H		NIM	0512513
2.	Keterangan	Tanggal	Waktu		Tempat / Ruang
	Pelaksanaan	30 MARET 2010	09.45 - 10.30		RUANG SEMINAR LT. 4
3.	Spesifikasi Judul (berilah tanda silang) *)				
	a. Sistem Tenaga Elektrik	e. Embbeded System		i. Sistem Informasi	
	b. Konversi Energi	f. Antar Muka		j. Jaringan Komputer	
	c. Sistem Kendali	g. Elektronika Telekomunikasi		k. Web	
	d. Tegangan Tinggi	h. Elektronika Instrumentasi		l. Algoritma Cerdas	
4.	Judul Proposal yang diseminarkan Mahasiswa	DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR			
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian				
6.	Catatan :				
7.	Catatan :				

Persetujuan Judul Skripsi

Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II
FANNY NASTITI	SOTYOHADI

Mengetahui, Ketua Jurusan.	Disetujui, Calon Dosen Pembimbing	
	Pembimbing I	Pembimbing II
Ir. F. Yudi Limpraptono,MT NIP. Y. 1039500274		

Keterangan :
*) dilingkari a, b, c, ... sesuai dengan bidang keahlian

Form S-3c

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo 10 KM
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : FEBRY DANANG EKA HERMAWAN
NIM : 05.12.513
Masa Bimbingan : 30 Maret 2010 s/d 30 September 2010
Judul Skripsi : DESAIN APLIKASI UNTUK MENAMPILKAN SCREEN CAPTURE KOMPUTER CLIENT KE LCD PROJECTOR

No	Tanggal	Uraian	Paraf Pembimbing
1	19 Juli 2010	Konsultasi Bab I dan II	
2	3 Agustus 2010	Konsultasi Bab III dan IV	
3	9 Agustus 2010	ACC Bab I dan II	
4	12 Agustus 2010	Revisi Bab III dan IV	
5	14 Agustus 2010	Konsultasi Program	
6	18 Agustus 2010	ACC Bab III dan IV	
7	19 Agustus 2010	ACC Program	
8			

Malang,
Dosen Pembimbing

I Komang Sumawirata, ST MT
NIP.Y. 1030100361

FORM S-4b

UNIT CLIENT

```
unit ServerDlg;

interface

uses

  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,

  ExtCtrls, StdCtrls, WinSock, ScktComp,
Menus, TrayIcon, FormSettings,

  RemConMessages, ZLib, MsgSimulator,
ComCtrls, ShellAPI, sSkinManager,

  sSkinProvider;

type

  TClientForm = class(TForm)

    PageControl1: TPageControl;

    TabSheet1: TTabSheet;

    TabSheet2: TTabSheet;

    LogList: TListBox;

    ServerSocket1: TServerSocket;

    TrayIcon1: TTrayIcon;

    TrayMenu: TPopupMenu;

    RemoteControl1: TMenuItem;

    N1: TMenuItem;

    Client1: TMenuItem;

    N2: TMenuItem;

    Shutdown1: TMenuItem;

    FormSettings1: TFormSettings;

    MsgSimulator1: TMsgSimulator;

    Timer1: TTimer;

    sSkinProvider1: TsSkinProvider;

    sSkinManager1: TsSkinManager;

    Label1: TLabel;

    NameLabel: TLabel;

    Label2: TLabel;

    PortEdit: TEdit;

    Label5: TLabel;

    StartLab: TLabel;

    ConLab: TLabel;

    Label9: TLabel;

    LastRecLab: TLabel;

    NumErrLab: TLabel;

    Label4: TLabel;

    Label3: TLabel;

    StartBut: TButton;

    DisconBut: TButton;

    MinimizeBut: TButton;

    ClientBut: TButton;

    procedure StartButClick(Sender: TObject);

    procedure DisconButClick(Sender:
TObject);

    procedure FormShow(Sender: TObject);

    procedure MinimizeButClick(Sender:
TObject);

    procedure RemoteControl1Click(Sender:
TObject);

    procedure Shutdown1Click(Sender:
TObject);

    procedure FormClose(Sender: TObject; var
Action: TCloseAction);

    procedure ServerSocket1Listen(Sender:
TObject;

      Socket: TCustomWinSocket);

    procedure
ServerSocket1ClientRead(Sender: TObject;

      Socket: TCustomWinSocket);
```

```
UNIT CLIENT

unit ServerDlg;

interface

uses

  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs,

  ExtCtrls, StdCtrls, WinSock, ScktComp,
  Menus, TrayIcon, FormSettings,

  RemComMessages, Zlib, MsgSimulator,
  ComCtrls, ShellAPI, SkinManager,

  sSkinProvider;

type

  TClientForm = class(TForm)

    PageControl1: TPageControl;

    TabSheet1: TTabSheet;

    TabSheet2: TTabSheet;

    LogList: TListBox;

    ServerSocket1: TServerSocket;

    TrayIcon1: TTrayIcon;

    TrayMenu: TPopupMenu;

    RemoteControl1: TMenuItem;

    N1: TMenuItem;

    Client1: TMenuItem;

    N2: TMenuItem;

    Shutdown1: TMenuItem;

    FormSettings1: TFormSettings;

    MsgSimulator1: TMsgSimulator;

    Timer1: TTimer;

    sSkinProvider1: TsSkinProvider;

    sSkinManager1: TsSkinManager;

    Label1: TLabel;

    NameLabel: TLabel;

    Label2: TLabel;

    PortEdit: TEdit;

    Label5: TLabel;

    startLab: TLabel;

    ConLab: TLabel;

    Label9: TLabel;

    LastRecLab: TLabel;

    NumErrLab: TLabel;

    Label4: TLabel;

    Label3: TLabel;

    StartBut: TButton;

    DisconBut: TButton;

    MinimizeBut: TButton;

    ClientBut: TButton;

    procedure StartButClick(Sender: TObject);

    procedure DisconButClick(Sender:
    TObject);

    procedure FormShow(Sender: TObject);

    procedure MinimizeButClick(Sender:
    TObject);

    procedure RemoteControl1Click(Sender:
    TObject);

    procedure Shutdown1Click(Sender:
    TObject);

    procedure FormClose(Sender: TObject; var
    Action: TCloseAction);

    procedure ServerSocket1Listen(Sender:
    TObject;

      Socket: TCustomWinSocket);

    procedure
    ServerSocket1ClientRead(Sender: TObject;

      Socket: TCustomWinSocket);
```

```
  procedure
ServerSocket1ClientConnect(Sender:
TObject;
    Socket: TCustomWinSocket);
  procedure
ServerSocket1ClientDisconnect(Sender:
TObject;
    Socket: TCustomWinSocket);
  procedure
ServerSocket1ClientError(Sender: TObject;
    Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;
    var ErrorCode: Integer);
  procedure FormCreate(Sender: TObject);
  procedure FormDestroy(Sender: TObject);
  procedure Client1Click(Sender: TObject);
  procedure FormCloseQuery(Sender:
TObject; var CanClose: Boolean);
  procedure ClientButClick(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
 protected
  NumRec    : double;
  NumSend   : double;
  NumError  : integer;
  CurMsg    : string;
  LoggedOn  : boolean;
  CurBmp    : TBitmap;
  CurSocket : TCustomWinSocket;
  CurHandle : THandle;
  SleepTime : integer;
  ViewMode  : TViewMode;
  CompMode  : TCompressionLevel;
  procedure UpdateStats;
  procedure Log(const s: string);
  procedure ProcessClick(const Data:
string);
  procedure ProcessDrag(const Data:
string);
  procedure Send_Screen_Update(Socket:
TCustomWinSocket);
  procedure SleepDone(Sender: TObject);
  procedure ProcessKeys(const Data:
string);
  procedure CreateSleepThread;
  procedure GetHostNameAddr;
  procedure ParseComLine;
  function  Get_Process_List: string;
  procedure CloseWindow(const Data:
string);
  procedure KillWindow(const Data: string);
  function  Get_Drive_List: string;
  function  GetDirectory(const PathName:
string): string;
  function  GetFile(const PathName: string):
string;
 public
  procedure EnableButs;
  procedure ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);
  procedure SendMsg(MsgNum: integer;
const MsgData: string; Socket:
TCustomWinSocket);
 end;
var
 ClientForm: TClientForm;
implementation
uses ClientFrm;
{$R *.DFM}
```

```
    procedure OnClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure OnClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure OnClientError(Sender: TObject;
      Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
      var ErrorCode: Integer);
    procedure OnClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure OnClientWrite(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ClientClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject;
      var CanClose: Boolean);
    procedure ClientDblClick(Sender: TObject);
    procedure TimerTimer(Sender: TObject);
  protected
    UnitRec    : double;
    UnitSend   : double;
    UnitError  : Integer;
    CmdMsg     : string;
    logged0n   : boolean;
    CmdBmp     : TBitmap;
    CmdSocket  : TCustomWinSocket;
    CmdHandle  : THandle;
    SleepTime  : Integer;
    ViewMode   : TViewMode;
    CombMode   : TCompressionLevel;
    procedure UpdateStatus;
    procedure ProcessMessage(const Msg:
      string; Socket: TCustomWinSocket);
    procedure SendMsg(MsgNum: Integer;
      const MsgData: string; Socket:
      TCustomWinSocket);
    procedure ProcessMsg(const Msg:
      string; Socket: TCustomWinSocket);
    procedure EnableBtns;
  public
    procedure Log(const s: string);
    procedure ProcessClick(const Data:
      string);
    procedure ProcessTag(const Data:
      string);
    procedure Send_ScreenUpdate(Socket:
      TCustomWinSocket);
    procedure SleepDone(Sender: TObject);
    procedure ProcessKeys(const Data:
      string);
    procedure CreateJpegThread;
    procedure GetHostNameAndAdd;
    procedure ParseCommand;
    function  Get_Process_List: string;
    procedure CloseWindow(const Data:
      string);
    procedure KillWindow(const Data: string);
    function  Get_Drive_List: string;
    function  GetDirectory(const ParmName:
      string): string;
    function  GetFile(const ParmName: string):
      string;
  public
    procedure ProcessMessage(const Msg:
      string; Socket: TCustomWinSocket);
    procedure SendMsg(MsgNum: Integer;
      const MsgData: string; Socket:
      TCustomWinSocket);
    end;
var
  Form1: TForm1;
  Client: TClientForm;
implementation
uses ClientImg;
{$R *.DFM}
```

```
procedure TClientForm.StartButClick(Sender:
TObject);

begin

  with ServerSocket1 do begin

    Port := StrToInt(PortEdit.Text);

    Active := True;

  end;

  EnableButs;

end;

procedure
TClientForm.DisconButClick(Sender:
TObject);

begin

  ServerSocket1.Active := False;

  EnableButs;

end;

procedure TClientForm.EnableButs;

var

  b : boolean;

begin

  b := ServerSocket1.Active;

  StartBut.Enabled := not b;

  PortEdit.Enabled := not b;

  DisconBut.Enabled := b;

  // MinimizeBut.Enabled := b;

//agar strat button dan portedit tidak aktif

end;

procedure TClientForm.GetHostNameAddr;

var

  buf  : array[0..MAX_PATH] of char;

  he   : PHostEnt;

  buf2 : PChar;

  rc   : integer;

begin

  rc := GetHostName(buf, sizeof(buf));

  if rc<>SOCKET_ERROR then begin

    he := GetHostByName(buf);

    if he = nil then begin

      rc := WSAGetLastError;

      NameLabel.Caption := Format('Socket
Error %d = %s', [rc, SysErrorMessage(rc)]);

    end else begin

      buf2 :=
inet_ntoa(PInAddr(he.h_addr^)^);

      NameLabel.Caption := Format('%s
(%s)', [buf, buf2]);

    end;

  end else begin

    NameLabel.Caption := 'Unknown Host';

  end;

end;

procedure TClientForm.FormShow(Sender:
TObject);

begin

  EnableButs;

  GetHostNameAddr;

  StartBut.Click;

end;

procedure
TClientForm.MinimizeButClick(Sender:
TObject);

begin

  if ServerSocket1.Active then begin

    TrayIcon1.ToolTip := Application.Title + ' -
Port: ' + PortEdit.Text;
```

```
procedure TClientForm.StartButClick(Sender:
TObject);

begin

  with ServerSocket1 do begin

    Port := StrToInt(PortEdit.Text);

    Active := True;

  end;

  EnableButs;

end;

procedure
TClientForm.DisconnButClick(Sender:
TObject);

begin

  ServerSocket1.Active := False;

  EnableButs;

end;

procedure TClientForm.EnableButs;

var

  b : boolean;

begin

  b := ServerSocket1.Active;

  StartBut.Enabled := not b;

  PortEdit.Enabled := not b;

  DisconnBut.Enabled := b;

  \\ MinimizeBut.Enabled := b;

\\agar stat button dan portedit tidak aktif

end;

procedure TClientForm.GetHostNameAdd;

var

  buf  : array[0..MAX_PATH] of char;

  he   : PHostEnt;

  buf2 : PChar;

  rc   : integer;

begin

  rc := GetHostName(buf, sizeof(buf));

  if rc<>SOCKET_ERROR then begin

    he := GetHostByName(buf);

    if he = nil then begin

      rc := WSAGetLastError;

      NameLabel.Caption := Format('Socket
Error %d = %s', [rc, SysErrorMessage(rc)]);

    end else begin

      buf2 :=
inet_ntoa(PInAddr(he.h_addr^)^);

      NameLabel.Caption := Format('%s
(%s)', [buf, buf2]);

    end;

  end else begin

    NameLabel.Caption := 'Unknown Host';

  end;

end;

procedure TClientForm.FormShow(Sender:
TObject);

begin

  EnableButs;

  GetHostNameAdd;

  StartBut.Click;

end;

procedure
TClientForm.MinimizeButClick(Sender:
TObject);

begin

  if ServerSocket1.Active then begin

    TrayIcon1.ToolTip := Application.Title + ' -
Port: ' + PortEdit.Text;
```

```
  end else begin

    TrayIcon1.ToolTip := Application.Title + ' -
Inactive';

  end;

  TrayIcon1.Active := True;

  ShowWindow(Application.Handle,
SW_HIDE);

  Hide;

end;

//^^--->>> untuk minimaze program pada
taskbar

procedure
TClientForm.RemoteControl1Click(Sender:
TObject);

begin

  TrayIcon1.Active := False;

  ShowWindow(Application.Handle,
SW_SHOW);

  Application.Restore;

  Show;

  SetForegroundWindow(Handle);

end;

//^^-->> untuk menampilakn form Client
ketika diklik kanan pada taskbar

procedure
TClientForm.Shutdown1Click(Sender:
TObject);

begin

  RemoteControl1Click(nil);

  Close;

end;

//^^-->> untuk mengakhiri program ketika
diklik kanan pada posisi di taskbar

//-----------------------------mulai capture ---
-----------------//

procedure TClientForm.FormClose(Sender:
TObject; var Action: TCloseAction);

begin

  FormSettings1.SaveSettings;

end;

procedure
TClientForm.ServerSocket1Listen(Sender:
TObject;

  Socket: TCustomWinSocket);

begin

  StartLab.Caption := CurTime;

  NumRec  := 0;

  NumSend  := 0;

  CurMsg  := '';

  LoggedOn := False;

  UpdateStats;

  Log('Mulai Start capture: ' + CurTime);

end;

//^^-->> nilai awal pada form client

procedure TClientForm.UpdateStats;

begin

  ConLab.Caption :=
IntToStr(ServerSocket1.Socket.ActiveConnec
tions);

  //NumRecLab.Caption := Format('%1.0n',
[NumRec]);

  //NumSendLab.Caption := Format('%1.0n',
[NumSend]);

  //NumErrLab.Caption :=
IntToStr(NumError);

end;

//^^-->>untuk memberikan nilai pada conlab
dll..
```

```
//----------mulai capture ---
----------//

//^^-->> untuk mengakhiri program ketika
diklik kanan pada posisi di taskbar

procedure TClientForm.ShutdownClick(Sender:
TObject);

begin

  RemoteControlClick(nil);

  Close;

end;

//^^-->> untuk menampilkan form Client
ketika diklik kanan pada taskbar

procedure
TClientForm.RemoteControlClick(Sender:
TObject);

begin

  ShowWindow(Application.Handle,
SW_SHOW);

  Application.Restore;

  Show;

  SetForegroundWindow(Handle);

end;

//^^-->> untuk sembunyikan ke taskbar

procedure
TClientForm.RemoteControlClick(Sender:
TObject);

begin

  TrayIcon1.Active := True;

  ShowWindow(Application.Handle,
SW_HIDE);

  Hide;

end;

//^^-->>> untuk minimize program pada
taskbar

procedure
TClientForm.RemoteControlClick(Sender:
TObject);

begin

  TrayIcon1.Active := False;

  ShowWindow(Application.Handle,
SW_SHOW);

  Application.Restore;

  Show;

  SetForegroundWindow(Handle);

end;

//^^-->> untuk menampilkan form Client
ketika diklik kanan pada taskbar

procedure
TClientForm.ShutdownClick(Sender:
TObject);

begin

  RemoteControlClick(nil);

  Close;

end;

//^^-->> untuk mengakhiri program ketika
diklik kanan pada posisi di taskbar

  TrayIcon1.ToolTip := Application.Title +
'-
Inactive';

end else begin

  TrayIcon1.Active := True;

  ShowWindow(Application.Handle,
SW_HIDE);

  Hide;

end;

//^^-->>> untuk minimize program pada
taskbar

procedure
TClientForm.RemoteControlClick(Sender:
TObject);

begin

  TrayIcon1.Active := False;

  ShowWindow(Application.Handle,
SW_SHOW);

  Application.Restore;

  Show;

  SetForegroundWindow(Handle);

end;

//^^---->> mulai awal pada form client

procedure TClientForm.UpdateStats;

begin

  CountLab.Caption :=
IntToStr(ServerSocket1.Socket.ActiveConnec
tions);

  //NumRecLab.Caption := Format('%1.0n',
[NumRec]);

  //NumSendLab.Caption := Format('%1.0n',
[NumSend]);

  //NumErrLab.Caption :=
IntToStr(NumError);

end;

//^^-->>untuk memberikan nilai pada conlab
dll..

procedure TClientForm.FormClose(Sender:
TObject; var Action: TCloseAction);

begin

  FormSettings1.SaveSettings;

end;

procedure
TClientForm.ServerSocket1Listen(Sender:
TObject;

Socket: TCustomWinSocket);

begin

  StartTab.Caption := CurTime;

  NumRec  := 0;

  NumSend  := 0;

  CurMsg  := '';

  LoggedOn := False;

  UpdateStats;

  Log('Mulai Start Capture: ' + CurTime);

end;

//^^---->> mulai awal pada form client

procedure TClientForm.UpdateStats;

begin

  CountLab.Caption :=
IntToStr(ServerSocket1.Socket.ActiveConnec
tions);

  //NumRecLab.Caption := Format('%1.0n',
[NumRec]);

  //NumSendLab.Caption := Format('%1.0n',
[NumSend]);

  //NumErrLab.Caption :=
IntToStr(NumError);

end;

//^^-->>untuk memberikan nilai pada conlab
dll..
```

```
procedure
TClientForm.ServerSocket1ClientRead(Sende
r: TObject;

 Socket: TCustomWinSocket);

var

  s : string;

begin

  Log(Format('%-20s %s', ['Recv Data',
Socket.RemoteAddress]));

  LastRecLab.Caption := CurTime;

  s := Socket.ReceiveText;

  NumRec := NumRec + Length(s);

  UpdateStats;

  CurMsg := CurMsg + s;

  while IsValidMessage(CurMsg) do begin

    s := TrimFirstMsg(CurMsg);

    ProcessMessage(s, Socket);

  end;

end;

procedure
TClientForm.ServerSocket1ClientConnect(Se
nder: TObject;

 Socket: TCustomWinSocket);

begin

  Log(Format('%-20s %s', ['Connect',
Socket.RemoteAddress]));

  ViewMode := vmColor4;

  CompMode := clDefault;

  SetThreadPriority(GetCurrentThread,
THREAD_PRIORITY_NORMAL);

  UpdateStats;

end;

procedure
TClientForm.ServerSocket1ClientDisconnect(
Sender: TObject;

 Socket: TCustomWinSocket);

begin

  Log(Format('%-20s %s', ['Disconnect',
Socket.RemoteAddress]));

  UpdateStats;

end;

procedure
TClientForm.ServerSocket1ClientError(Sende
r: TObject;

 Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;

 var ErrorCode: Integer);

begin

  Log(Format('%-20s %d', ['Error',
ErrorCode]));

  ErrorCode := 0;

  Inc(NumError);

  UpdateStats;

end;

procedure TClientForm.Log(const s: string);

begin

  LogList.ItemIndex := LogList.Items.Add(s);

end;

procedure
TClientForm.ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);

var

  MsgNum, x: integer;

  rc     : integer;

  Data   : string;

  bmp    : TBitmap;
```

```
procedure
TClientForm.ServerSocketClientRead(Sende
r: TObject;

  Socket: TCustomWinSocket);

var

  s : string;

begin

  Log(Format('%-20s %s', ['Recv Data',
Socket.RemoteAddress]));

  LastRecLab.Caption := CurTime;

  s := Socket.ReceiveText;

  NumRec := NumRec + Length(s);

  UpdateStats;

  CurMsg := CurMsg + s;

  while IsValidMessage(CurMsg) do begin

    s := TrimFirstMsg(CurMsg);

    ProcessMessage(s, Socket);

  end;

end;

procedure
TClientForm.ServerSocketClientConnect(Se
nder: TObject;

  Socket: TCustomWinSocket);

begin

  Log(Format('%-20s %s', ['Connect',
Socket.RemoteAddress]));

  ViewMode := vmColor4;

  CompMode := cdDefault;

  SetThreadPriority(GetCurrentThread,
THREAD_PRIORITY_NORMAL);

  UpdateStats;

end;
```

```
procedure
TClientForm.ServerSocketClientDisconnect(
Sender: TObject;

  Socket: TCustomWinSocket);

begin

  Log(Format('%-20s %s', ['Disconnect',
Socket.RemoteAddress]));

  UpdateStats;

end;

procedure
TClientForm.ServerSocketClientError(Sende
r: TObject;

  Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;

  var ErrorCode: Integer);

begin

  Log(Format('%-20s %d', ['Error',
ErrorCode]));

  ErrorCode := 0;

  Inc(NumError);

  UpdateStats;

end;

procedure TClientForm.Log(const s: string);

begin

  LogList.ItemIndex := LogList.Items.Add(s);

end;

procedure
TClientForm.ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);

var

  MsgNum, x: Integer;

  rc   : Integer;

  Data : string;

  bmp  : TBitmap;
```

```
  tmp     : string;
begin
  CurSocket := Socket;
  Move(Msg[1], MsgNum, sizeof(integer));
  Data := Copy(Msg, 9, Length(Msg));
  Log(Format('%-20s %d', ['Message',
MsgNum]));
  if MsgNum = MSG_LOGON then begin
    //LoggedOn := (AnsiCompareText(Data,
PassEdit.Text) = 0);
    if LoggedOn then begin
      SendMsg(MSG_LOGON, '1', Socket)
    end else begin
      SendMsg(MSG_LOGON, '0', Socket);
    end;
    exit;
  end;
  if not LoggedOn then begin
    Log('Denied Access!');
    SendMsg(MSG_STAT_MSG, 'Invalid
Password', Socket);
    Socket.Close;
    exit;
  end;
  if MsgNum = MSG_REFRESH then begin
    Log('Screen Capture');
    SendMsg(MSG_STAT_MSG, 'Screen
Capture', Socket);
    GetScreen(bmp, ViewMode);
    Log('Compressing Bitmap');
    SendMsg(MSG_STAT_MSG, 'Screen
Compression', Socket);
    CompressBitmap(bmp, tmp);
    SaveString(tmp, 'Temp1.txt');
    SendMsg(MSG_REFRESH, tmp, Socket);
    CurBmp.Assign(bmp);
    bmp.Free;
  end;
  if MsgNum = MSG_SCREEN_UPDATE then
begin
    Send_Screen_Update(Socket);
  end;
  if MsgNum = MSG_CLICK then begin
    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);
    ProcessClick(Data);
    // SleepDone will be called when it is
finished
  end;
  if MsgNum = MSG_DRAG then begin
    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);
    ProcessDrag(Data);
    // SleepDone will be called when it is
finished
  end;
  if MsgNum = MSG_KEYS then begin
    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);
    ProcessKeys(Data);
    // SleepDone will be called when it is
finished
  end;
  if MsgNum = MSG_SEVER_DELAY then
begin
```

```
  tmp     : string;

begin

  CurSocket := Socket;

  Move(Msg[1], MsgNum, sizeof(integer));

  Data := Copy(Msg, 9, Length(Msg));

  Log(Format('%-20s %d', ['Message',
MsgNum]));

  if MsgNum = MSG_LOGON then begin

    //LoggedOn := (AnsiCompareTxt(Data,
PassEdit.Text) = 0);

    if LoggedOn then begin

      SendMsg(MSG_LOGON, '1', Socket)

    end else begin

      SendMsg(MSG_LOGON, '0', Socket);

    end;

    exit;

  end;

  if not LoggedOn then begin

    Log('Denied Access!');

    SendMsg(MSG_STAT_MSG, 'Invalid
Password', Socket);

    Socket.Close;

    exit;

  end;

  if MsgNum = MSG_REFRESH then begin

    Log('Screen Capture');

    SendMsg(MSG_STAT_MSG, 'Screen
Capture', Socket);

    GetScreen(bmp, ViewMode);

    Log('Compressing Bitmap');

    SendMsg(MSG_STAT_MSG, 'Screen
Compression', Socket);

    CompressBitmap(bmp, tmp);

    SaveString(tmp, 'Temp1.txt');

    SendMsg(MSG_REFRESH, tmp, Socket);

    CurBmp.Assign(bmp);

    bmp.Free;

  end;

  if MsgNum = MSG_SCREEN_UPDATE then
begin

    Send_Screen_Update(Socket);

  end;

  if MsgNum = MSG_CLICK then begin

    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);

    ProcessClick(Data);

    \\ SleepDone will be called when it is
finished

  end;

  if MsgNum = MSG_DRAG then begin

    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);

    ProcessDrag(Data);

    \\ SleepDone will be called when it is
finished

  end;

  if MsgNum = MSG_KEYS then begin

    SendMsg(MSG_STAT_MSG, 'Simulating
Input', Socket);

    ProcessKeys(Data);

    \\ SleepDone will be called when it is
finished

  end;

  if MsgNum = MSG_SEVER_DELAY then
begin
```

```
    Move(Data[1], SleepTime,
sizeof(integer));

    SendMsg(MSG_SEVER_DELAY, '', Socket);

  end;

  if MsgNum = MSG_VIEW_MODE then begin

    Move(Data[1], x, sizeof(integer));

    ViewMode := TViewMode(x);

    SendMsg(MSG_VIEW_MODE, '', Socket);

  end;

  if MsgNum = MSG_FOCUS_SERVER then
begin

    if TrayIcon1.Active then
RemoteControl1Click(nil);

    SetFocus;

    CreateSleepThread;

    // SleepDone will be called when it is
finished

  end;

  if MsgNum = MSG_COMP_MODE then
begin

    Move(Data[1], x, sizeof(integer));

    CompMode := TCompressionLevel(x);

    SendMsg(MSG_COMP_MODE, '', Socket);

  end;

  if MsgNum = MSG_PRIORITY_MODE then
begin

    Move(Data[1], x, sizeof(integer));

    SetThreadPriority(GetCurrentThread, x);

    SendMsg(MSG_PRIORITY_MODE, '',
Socket);

  end;

  if MsgNum = MSG_PROCESS_LIST then
begin

    SendMsg(MSG_PROCESS_LIST,
Get_Process_List, Socket);

  end;

  if MsgNum = MSG_CLOSE_WIN then begin

    CloseWindow(Data);

  end;

  if MsgNum = MSG_KILL_WIN then begin

    KillWindow(Data);

  end;

  if MsgNum = MSG_DRIVE_LIST then begin

    SendMsg(MSG_DRIVE_LIST,
Get_Drive_List, Socket);

  end;

  if MsgNum = MSG_DIRECTORY then begin

    SendMsg(MSG_DIRECTORY,
GetDirectory(Data), Socket);

  end;

  if MsgNum = MSG_FILE then begin

    SendMsg(MSG_FILE, GetFile(Data),
Socket);

  end;

  if MsgNum = MSG_REMOTE_LAUNCH then
begin

    SendMsg(MSG_STAT_MSG, 'Launching
File: ' + Data, Socket);

    rc := ShellExecute(Handle, 'open',
PChar(Data), nil, nil, SW_SHOWNORMAL);

    if rc <= 32 then begin

      Data := Format('ShellExecute Error #%d
Launching %s', [rc, Data]);

      SendMsg(MSG_REMOTE_LAUNCH,
Data, Socket);

    end else begin

      SendMsg(MSG_REMOTE_LAUNCH,
Data, Socket);
```

```
      Move(Data[1], SleepTime,
sizeof(integer));

      SendMsg(MSG_SERVER_DELAY, '', Socket);

    end;

    if MsgNum = MSG_VIEW_MODE then begin

      Move(Data[1], x, sizeof(integer));

      ViewMode := TViewMode(x);

      SendMsg(MSG_VIEW_MODE, '', Socket);

    end;

    if MsgNum = MSG_FOCUS_SERVER then
    begin

      if TrayIcon1.Active then
      RemoteControlClick(nil);

      SetFocus;

      CreateSleepThread;

      \\ SleepDone will be called when it is
      finished

    end;

    if MsgNum = MSG_COMP_MODE then
    begin

      Move(Data[1], x, sizeof(integer));

      CompMode := TCompressionLevel(x);

      SendMsg(MSG_COMP_MODE, '', Socket);

    end;

    if MsgNum = MSG_PRIORITY_MODE then
    begin

      Move(Data[1], x, sizeof(integer));

      SetThreadPriority(GetCurrentThread, x);

      SendMsg(MSG_PRIORITY_MODE, '',
      Socket);

    end;

    if MsgNum = MSG_PROCESS_LIST then
    begin

      SendMsg(MSG_PROCESS_LIST,
      Get_Process_List, Socket);

    end;

    if MsgNum = MSG_CLOSE_WIN then begin

      CloseWindow(Data);

    end;

    if MsgNum = MSG_KILL_WIN then begin

      KillWindow(Data);

    end;

    if MsgNum = MSG_DRIVE_LIST then begin

      SendMsg(MSG_DRIVE_LIST,
      Get_Drive_List, Socket);

    end;

    if MsgNum = MSG_DIRECTORY then begin

      SendMsg(MSG_DIRECTORY,
      GetDirectory(Data), Socket);

    end;

    if MsgNum = MSG_FILE then begin

      SendMsg(MSG_FILE, GetFile(Data),
      Socket);

    end;

    if MsgNum = MSG_REMOTE_LAUNCH then
    begin

      SendMsg(MSG_CHAT_MSG, 'Launching
      File: ' + Data, Socket);

      rc := ShellExecute(Handle, 'open',
      PChar(Data), nil, nil, SW_SHOWNORMAL);

      if rc <= 32 then begin

        Data := Format('ShellExecute Error #%d
        Launching %s', [rc, Data]);

        SendMsg(MSG_REMOTE_LAUNCH,
        Data, Socket);

      end else begin

        SendMsg(MSG_REMOTE_LAUNCH,
        Data, Socket);
```

```
    end;
  end;
end;
function EnumWinProc(hw: THandle; lp:
LParam): boolean; stdcall;
var
  sl   : TStringList;
  buf  : array[0..MAX_PATH] of char;
  s, iv : string;
begin
  sl := TStringList(lp);
  GetWindowText(hw, buf, sizeof(buf));
  if buf<>'' then begin
    if IsWindowVisible(hw) then iv := '' else iv
:= '(Invisible)';
    s := Format('%8.8x - %-32s  %s', [hw, buf,
iv]);
    sl.AddObject(s, TObject(hw));
  end;
  Result := True;
end;
function TClientForm.Get_Process_List:
string;
var
  sl : TStringList;
begin
  sl := TStringList.Create;
  EnumWindows(@EnumWinProc,
integer(sl));
  Result := sl.Text;
  sl.Free;
end;
function TClientForm.Get_Drive_List: string;
var
  DriveBits : integer;
  i       : integer;
begin
  Result := '';
  DriveBits := GetLogicalDrives;
  for i := 0 to 25 do begin
    if (DriveBits and (1 shl i)) <> 0 then
      Result := Result + Chr(Ord('A') + i) + ':\' +
#13#10;
  end;
end;
function TClientForm.GetDirectory(const
PathName: string): string;
var
  DirList   : TStringList;
  CommaList : TStringList;
  sr      : TSearchRec;
  s       : string;
  dt      : TDateTime;
begin
  DirList := TStringList.Create;
  CommaList := TStringList.Create;
  if FindFirst(PathName, faAnyFile, sr) = 0
then repeat
    CommaList.Clear;
    s := sr.Name;
    if (s = '.') or (s = '..') then continue;
    if (sr.Attr and faDirectory) <> 0 then s := s
+ '\';
    CommaList.Add(s);
```

```
end;

end;

end;

function EnumWinProc(hw: THandle; lp:
LParam): boolean; stdcall;

var

  sl  : TStringList;

  buf  : array[0..MAX_PATH] of char;

  s, iv : string;

begin

  sl := TStringList(lp);

  GetWindowText(hw, buf, sizeof(buf));

  if buf<>'' then begin

    if IsWindowVisible(hw) then iv := '' else iv
:= '(invisible)';

    s := Format('%8.8x - %-22s %s', [hw, buf,
iv]);

    sl.AddObject(s, TObject(hw));

  end;

  Result := True;

end;

function TClientForm.Get_Process_List:
string;

var

  sl : TStringList;

begin

  sl := TStringList.Create;

  EnumWindows(@EnumWinProc,
integer(sl));

  Result := sl.Text;

  sl.Free;

end;
```

```
function TClientForm.Get_Drive_List: string;

var

  DriveBits : integer;

  i     : integer;

begin

  Result := '';

  DriveBits := GetLogicalDrives;

  for i := 0 to 25 do begin

    if (DriveBits and (1 shl i)) <> 0 then

      Result := Result + Chr(Ord('A') + i) + ':/' +
#13#10;

  end;

end;

function TClientForm.GetDirectory(const
PathName: string): string;

var

  DirList  : TStringList;

  CommaList : TStringList;

  sr     : TSearchRec;

  s      : string;

  dt     : TDateTime;

begin

  DirList := TStringList.Create;

  CommaList := TStringList.Create;

  if FindFirst(PathName, faAnyFile, sr) = 0
then repeat

    CommaList.Clear;

    s := sr.Name;

    if (s = '.') or (s = '..') then continue;

    if (sr.Attr and faDirectory) <> 0 then s := s
+ '/';

    CommaList.Add(s);
```

```
    s := Format('%1.0n', [sr.Size+0.0]);
    CommaList.Add(s);
    dt := FileDateToDateTime(sr.Time);
    s := FormatDateTime('yyyy-mm-dd  hh:nn
ampm', dt);
    CommaList.Add(s);
    DirList.Add(CommaList.CommaText);
  until FindNext(sr) <> 0;
  FindClose(sr);
  Result := DirList.Text;
  CommaList.Free;
  DirList.Free;
end;
function TClientForm.GetFile(const
PathName: string): string;
var
  fs : TFileStream;
begin
  fs := TFileStream.Create(PathName,
fmOpenRead or fmShareDenyWrite);
  SetLength(Result, fs.Size);
  fs.Read(Result[1], fs.Size);
  fs.Free;
end;
procedure TClientForm.CloseWindow(const
Data: string);
var
  sl : TStringList;
  i  : integer;
  hw : THandle;
begin
  sl := TStringList.Create;
  EnumWindows(@EnumWinProc,
integer(sl));
  i := sl.IndexOf(Data);
  if i<>-1 then begin
    hw := THandle(sl.Objects[i]);
    SendMessage(hw, WM_CLOSE, 0, 0);
    Sleep(SleepTime);
    SendMsg(MSG_PROCESS_LIST,
Get_Process_List, CurSocket);
  end;
  sl.Free;
end;
procedure TClientForm.KillWindow(const
Data: string);
var
  sl    : TStringList;
  i     : integer;
  hw    : THandle;
  ProcID : integer;
  hProc  : THandle;
begin
  sl := TStringList.Create;
  EnumWindows(@EnumWinProc,
integer(sl));
  i := sl.IndexOf(Data);
  if i<>-1 then begin
    hw := THandle(sl.Objects[i]);
    GetWindowThreadProcessId(hw,
@ProcID);
    hProc :=
OpenProcess(PROCESS_ALL_ACCESS, False,
ProcID);
    TerminateProcess(hProc, DWORD(-1));
```

```
  sl := TStringList.Create;
begin
  hw : THandle;
  i : integer;
  sl : TStringList;
var
procedure TClientForm.CloseWindow(const
Data: string);
end;
  fs.Free;
  fs.Read(Result[1], fs.Size);
  SetLength(Result, fs.Size);
  fs := TFileStream.Create(PathName,
fmOpenRead or fmShareDenyWrite);
begin
var
function TClientForm.GetFile(const
PathName: string): string;
end;
  Commalist.Free;
  Result := DirList.Text;
  FindClose(sr);
until FindNext(sr) <> 0;
  DirList.Add(Commalist.CommaText);
  Commalist.Add(s);
  s := FormatDateTime('yyyy-mm-dd hh:nn:
ampm', dt);
  dt := FileDateToDateTime(sr.Time);
  Commalist.Add(s);
  s := Format('%d.0n', [sr.Size+0.0]);
```

```
  EnumWindows(@EnumWinProc,
integer(sl));
  i := sl.IndexOf(Data);
  if i<>-1 then begin
  hw := THandle(sl.Objects[i]);
  SendMessage(hw, WM_CLOSE, 0, 0);
  Sleep(SleepTime);
  SendMsg(MSG_PROCESS_LIST,
0, Proc_Process_List, CurSocket);
end;
  sl.Free;
end;
procedure TClientForm.KillWindow(const
Data: string);
var
  sl  : TStringList;
  i  : integer;
  hw  : THandle;
  ProcID : integer;
  hProc : THandle;
begin
  sl := TStringList.Create;
  EnumWindows(@EnumWinProc,
integer(sl));
  i := sl.IndexOf(Data);
  if i<>-1 then begin
  hw := THandle(sl.Objects[i]);
  GetWindowThreadProcessId(hw,
@ProcID);
  hProc :=
OpenProcess(PROCESS_ALL_ACCESS, False,
ProcID);
  TerminateProcess(hProc, DWORD(-1));
```

```
    CloseHandle(hProc);

    Sleep(SleepTime);

    SendMsg(MSG_PROCESS_LIST,
Get_Process_List, CurSocket);

  end;

  sl.Free;

end;

procedure TClientForm.SleepDone(Sender:
TObject);

begin

  Send_Screen_Update(CurSocket);

end;

procedure
TClientForm.Send_Screen_Update(Socket:
TCustomWinSocket);

var

  bmp, dif : TBitmap;

  R      : TRect;

  tmp    : string;

begin

  Log('Screen Capture');

  SendMsg(MSG_STAT_MSG, 'Screen
Capture', Socket);

  GetScreen(bmp, ViewMode);

  Log('Creating Diff Image');

  dif := TBitmap.Create;

  dif.Assign(bmp);

  R := Rect(0, 0, dif.Width, dif.Height);

  SendMsg(MSG_STAT_MSG, 'Screen
Difference', Socket);

  dif.Canvas.CopyMode := cmSrcInvert;

  dif.Canvas.CopyRect(R, CurBmp.Canvas, R);

  Log('Compressing Bitmap');

  SendMsg(MSG_STAT_MSG, 'Screen
Compression', Socket);

  CompressBitmap(dif, tmp);

  SendMsg(MSG_SCREEN_UPDATE, tmp,
Socket);

  CurBmp.Assign(bmp);

  dif.Free;

  bmp.Free;

end;

function GetMB(but: integer):
TMouseButton;

begin

  case but of

    1 : Result := mbLeft;

    2 : Result := mbRight;

    else Result := mbLeft;

  end;

end;

procedure TClientForm.ProcessClick(const
Data: string);

var

  x, y, i  : integer;

  num, but : integer;

  p      : TPoint;

begin

  Move(Data[1], x, sizeof(integer));

  Move(Data[1+4], y, sizeof(integer));

  Move(Data[1+8], num, sizeof(integer));

  Move(Data[1+12], but, sizeof(integer));

  // Find the Window Handle

  p := Point(x, y);

  CurHandle := WindowFromPoint(p);
```

```
  CloseHandle(hProc);

  Sleep(SleepTime);

  SendMsg(MSG_PROCESS_LIST,
Get_Process_List, CurSocket);

end;

sl.Free;

end;

procedure TClientForm.SleepDone(Sender:
TObject);

begin

  Send_Screen_Update(CurSocket);

end;

procedure
TClientForm.Send_Screen_Update(Socket:
TCustomWinSocket);

var

  bmp, dif : TBitmap;

  R      : TRect;

  tmp    : string;

begin

  Log('Screen Capture');

  SendMsg(MSG_STAT_MSG, 'Screen
Capture', Socket);

  GetScreen(bmp, ViewMode);

  Log('Creating Diff Image');

  dif := TBitmap.Create;

  dif.Assign(bmp);

  R := Rect(0, 0, dif.Width, dif.Height);

  SendMsg(MSG_STAT_MSG, 'Screen
Difference', Socket);

  dif.Canvas.CopyMode := cmSrcInvert;

  dif.Canvas.CopyRect(R, CurBmp.Canvas, R);

  Log('Compressing Bitmap');

  SendMsg(MSG_STAT_MSG, 'Screen
Compression', Socket);

  CompressBitmap(dif, tmp);

  SendMsg(MSG_SCREEN_UPDATE, tmp,
Socket);

  CurBmp.Assign(bmp);

  dif.Free;

  bmp.Free;

end;

function GetMB(but: integer):
TMouseButton;

begin

  case but of

    1 : Result := mbLeft;

    2 : Result := mbRight;

  else Result := mbLeft;

  end;

end;

procedure TClientForm.ProcessClick(const
Data: string);

var

  x, y, i : integer;

  num, but : integer;

  p      : TPoint;

begin

  Move(Data[1], x, sizeof(integer));

  Move(Data[1+4], y, sizeof(integer));

  Move(Data[1+8], num, sizeof(integer));

  Move(Data[1+12], but, sizeof(integer));

  \\ Find the Window Handle

  p := Point(x, y);

  CurHandle := WindowFromPoint(p);
```

```
  Assert(CurHandle<>0);
  SetCursorPos(x, y);
  // Create the Messages to send in the Hook
procedure
  with MsgSimulator1 do begin
    Messages.Clear;
    for i := 1 to num do
      Add_ClickEx(0, GetMB(but), [], x, y, 1);
    Play;
  end;
  CreateSleepThread;
end;
procedure TClientForm.ProcessDrag(const
Data: string);
var
  x, y     : integer;
  time     : integer;
  num, but  : integer;
  p        : TPoint;
  StartPt   : TPoint;
  StopPt    : TPoint;
begin
  Move(Data[1], but, sizeof(integer));
  Move(Data[1+4], num, sizeof(integer));
  Assert(num > 2);
  // Create the Messages to send in the Hook
procedure
  // Mouse Down
  Move(Data[(1-1)*12 + 9], x,
sizeof(integer));
  Move(Data[(1-1)*12 + 13], y,
sizeof(integer));
  Move(Data[(1-1)*12 + 17], time,
sizeof(integer));
  SetCursorPos(x, y);
  // Find the Window Handle
  p := Point(x, y);
  CurHandle := WindowFromPoint(p);
  Assert(CurHandle<>0);
  with MsgSimulator1 do begin
    Messages.Clear;
    StartPt.X := x;
    StartPt.Y := y;
    Windows.ScreenToClient(CurHandle,
StartPt);
    Move(Data[(num-1)*12 + 9], x,
sizeof(integer));
    Move(Data[(num-1)*12 + 13], y,
sizeof(integer));
    StopPt.X := x;
    StopPt.Y := y;
    Windows.ScreenToClient(CurHandle,
StopPt);
    Add_Window_Drag(CurHandle, StartPt.X,
StartPt.Y, StopPt.X, StopPt.Y);
    Play;
  end;
  CreateSleepThread;
end;
procedure TClientForm.ProcessKeys(const
Data: string);
begin
  with MsgSimulator1 do begin
    Messages.Clear;
    Add_ASCII_Keys(Data);
```

```
Assert(CurHandle<>0);

SetCursorPos(x, y);

\\ Create the Messages to send in the Hook
procedure

with MsgSimulator1 do begin

Messages.Clear;

for i := 1 to num do

Add_ClickEx(0, GetMB(but), [], x, y, 1);

Play;

end;

CreateSleepThread;

end;

procedure TClientForm.ProcessDrag(const
Data: string);

var

x, y     : integer;

time     : integer;

num, but : integer;

p        : TPoint;

StartPt  : TPoint;

StopPt   : TPoint;

begin

Move(Data[1], but, sizeof(integer));

Move(Data[1+4], num, sizeof(integer));

Assert(num > 2);

\\ Create the Messages to send in the Hook
procedure

\\ Mouse Down

Move(Data[(1-1)*12 + 9], x,
sizeof(integer));

Move(Data[(1-1)*12 + 13], y,
sizeof(integer));

Move(Data[(1-1)*12 + 17], time,
sizeof(integer));

SetCursorPos(x, y);

\\ Find the Window Handle

p := Point(x, y);

CurHandle := WindowFromPoint(p);

Assert(CurHandle<>0);

with MsgSimulator1 do begin

Messages.Clear;

StartPt.X := x;

StartPt.Y := y;

Windows.ScreenToClient(CurHandle,
StartPt);

Move(Data[(num-1)*12 + 9], x,
sizeof(integer));

Move(Data[(num-1)*12 + 13], y,
sizeof(integer));

StopPt.X := x;

StopPt.Y := y;

Windows.ScreenToClient(CurHandle,
StopPt);

Add_Window_Drag(CurHandle, StartPt.X,
StartPt.Y, StopPt.X, StopPt.Y);

Play;

end;

CreateSleepThread;

end;

procedure TClientForm.ProcessKeys(const
Data: string);

begin

with MsgSimulator1 do begin

Messages.Clear;

Add_ASCII_Keys(Data);
```

```
    Play;
  end;
  CreateSleepThread;
end;
procedure TClientForm.SendMsg(MsgNum:
integer; const MsgData: string; Socket:
TCustomWinSocket);
var
  s : string;
begin
  s := IntToByteStr(MsgNum) +
IntToByteStr(Length(MsgData)) + MsgData;

  Log(Format('%-20s %-4d %1.0n', ['Send',
MsgNum, Length(s)+0.0]));
  Socket.SendText(s);
  NumSend := NumSend + Length(s);
  UpdateStats;
end;
procedure TClientForm.FormCreate(Sender:
TObject);
begin
  CurBmp   := TBitmap.Create;
  SleepTime := 50;
  ParseComLine;
  Timer1.Enabled :=True;
end;
procedure TClientForm.FormDestroy(Sender:
TObject);
begin
  CurBmp.Free;
end;
type
 TSleepThread = class(TThread)
 public
   SleepTime   : integer;
   procedure   Execute; override;
 end;
procedure TSleepThread.Execute;
begin
 Sleep(SleepTime);
end;
procedure TClientForm.CreateSleepThread;
var
  st : TSleepThread;
begin
  st := TSleepThread.Create(True);
  st.SleepTime := SleepTime;
  st.OnTerminate := SleepDone;
  st.Resume;
end;
procedure TClientForm.Client1Click(Sender:
TObject);
begin
  ServerForm.Show;
end;
procedure
TClientForm.FormCloseQuery(Sender:
TObject;
 var CanClose: Boolean);
var
  rc : integer;
begin
  if ServerSocket1.Socket.ActiveConnections
> 0 then begin
```

```
  Play;

end;

CreateSleepThread;

end;

procedure TClientForm.SendMsg(MsgNum:
integer; const MsgData: string; Socket:
TCustomWinSocket);

var

  s : string;

begin

  s := IntToByteStr(MsgNum) +
IntToByteStr(Length(MsgData)) + MsgData;


  Log(Format('%-20s %-4d %d.0n', ['Send',
MsgNum, Length(s)+0.0]));

  Socket.SendText(s);

  NumSend := NumSend + Length(s);

  UpdateStats;

end;

procedure TClientForm.FormCreate(Sender:
TObject);

begin

  CurBmp    := TBitmap.Create;

  SleepTime := 50;

  ParseCmdLine;

  Timer1.Enabled :=True;

end;

procedure TClientForm.FormDestroy(Sender:
TObject);

begin

  CurBmp.Free;

end;

type
```

```
  TSleepThread = class(TThread)

  public

    SleepTime  : integer;

    procedure  Execute; override;

  end;

procedure TSleepThread.Execute;

begin

  Sleep(SleepTime);

end;

procedure TClientForm.CreateSleepThread;

var

  st : TSleepThread;

begin

  st := TSleepThread.Create(True);

  st.SleepTime := SleepTime;

  st.OnTerminate := SleepDone;

  st.Resume;

end;

procedure TClientForm.Client1Click(Sender:
TObject);

begin

  ServerForm.Show;

end;

procedure
TClientForm.FormCloseQuery(Sender:
TObject;

  var CanClose: Boolean);

var

  rc : integer;

begin

  if ServerSocket1.Socket.ActiveConnections
> 0 then begin
```

```
    rc := MessageDlg('Client Masih
Terhubung ,Apakah Ingin Keluar Sekarang?',
      mtWarning, mbYesNoCancel, 0);
    CanClose := (rc = mrYes);
  end;
end;
procedure TClientForm.ParseComLine;
var
  i       : integer;
  s       : string;
  AutoStart  : boolean;
begin
  AutoStart := False;
  for i := 1 to ParamCount do begin
    s := UpperCase(ParamStr(i));
    if Copy(s, 1, 6) = '/PORT:' then begin
      PortEdit.Text := Copy(s, 7, Length(s));
      AutoStart := True;
      StartButClick(nil);
      MinimizeButClick(nil);
    end;
    if s = '/CLIENT' then begin
      MinimizeButClick(nil);
      AutoStart := True;
    end;
  end;
  if not AutoStart then
    Visible := True;
end;
procedure
TClientForm.ClientButClick(Sender: TObject);
begin
  ServerForm.Show;
end;
procedure TClientForm.Timer1Timer(Sender:
TObject);
begin
  Timer1.Enabled :=False;
  MinimizeBut.Click;
end;
end.
```

UNIT SERVER

```
unit ClientFrm;
interface
uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  ScktComp, ExtCtrls, ComCtrls, FormSettings,
Menus, StdCtrls, Buttons,
  RemConMessages, ZLib;
const
  DEFAULT_SERVER_DELAY = 500;
  DEFAULT_VIEW_MODE   = vmColor4;
  DEFAULT_COMP_MODE   = clDefault;
  DEFAULT_SVR_PRIORITY =
THREAD_PRIORITY_HIGHEST;
type
  TMoveObj = class
    X, Y  : integer;
    Time  : integer;
  end;
  TServerForm = class(TForm)
```

```
rc := MessageDlg('Client Masih
Terhubung, Apakah Ingin Keluar Sekarang?',

  mtWarning, mbYesNoCancel, 0);

  CanClose := (rc = mrYes);

end;

end;

procedure TClientForm.ParseCmdLine;

var

  i      : integer;

  s      : string;

  AutoStart : boolean;

begin

  AutoStart := false;

  for i := 1 to ParamCount do begin

    s := UpperCase(ParamStr(i));

    if Copy(s, 1, 6) = '/PORT:' then begin

      PortEdit.Text := Copy(s, 7, length(s));

      AutoStart := True;

      StartButClick(nil);

      MinimizeButClick(nil);

    end;

    if s = '/CLIENT' then begin

      MinimizeButClick(nil);

      AutoStart := True;

    end;

  end;

  if not AutoStart then

    Visible := True;

end;

procedure
TClientForm.ClientButClick(Sender: TObject);

begin

  ServerForm.Show;

end;

procedure TClientForm.TimerITimer(Sender:
TObject);

begin

  Timer1.Enabled :=false;

  MinimizeBut.Click;

end;

end.
```

UNIT SERVER

```
unit ClientFrm;

interface

uses

  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,

  StdCtrls, ExtCtrls, ComCtrls, FormSettings,
Menus, StdCtrls, Buttons,

  RemConMessages, Calib;

const

  DEFAULT_SERVER_DELAY = 500;

  DEFAULT_VIEW_MODE    = vmColor4;

  DEFAULT_COMP_MODE    = clDefault;

  DEFAULT_SVR_PRIORITY =
THREAD_PRIORITY_HIGHEST;

type

  TMoveObj = class

    X, Y : integer;

    Time : integer;

  end;

  TServerForm = class(TForm)
```

StatPanel: TPanel;

StatusBar1: TStatusBar;

ScrollBox1: TScrollBox;

Image1: TImage;

ClientSocket1: TClientSocket;

Timer1: TTimer;

FormSettings1: TFormSettings;

MainMenu1: TMainMenu;

File1: TMenuItem;

Connect1: TMenuItem;

N1: TMenuItem;

Exit1: TMenuItem;

Disconnect1: TMenuItem;

View1: TMenuItem;

RefreshComplete1: TMenuItem;

UpdateChanges1: TMenuItem;

ResponseTimer: TTimer;

ClickTimer: TTimer;

Options1: TMenuItem;

ServerPause1: TMenuItem;

N005sec1: TMenuItem;

N010sec1: TMenuItem;

N050sec1: TMenuItem;

N100sec1: TMenuItem;

N200sec1: TMenuItem;

N500sec1: TMenuItem;

LogList: TListBox;

Splitter1: TSplitter;

N2: TMenuItem;

Log1: TMenuItem;

CommStat1: TMenuItem;

N3: TMenuItem;

Shutdown1: TMenuItem;

Special1: TMenuItem;

FocusServerWindow1: TMenuItem;

BitmapFormat1: TMenuItem;

Color4: TMenuItem;

Gray4: TMenuItem;

Gray8: TMenuItem;

Color24: TMenuItem;

Default1: TMenuItem;

WaitImage: TImage;

CompressionLevel1: TMenuItem;

HighSlow1: TMenuItem;

Medium1: TMenuItem;

LowFast1: TMenuItem;

ServerPriority1: TMenuItem;

Critical1: TMenuItem;

Highest1: TMenuItem;

AboveNormal1: TMenuItem;

Normal1: TMenuItem;

BelowNormal1: TMenuItem;

Lowest1: TMenuItem;

Idle1: TMenuItem;

N4: TMenuItem;

ScaleImage1: TMenuItem;

N5: TMenuItem;

FileList1: TMenuItem;

Panel1: TPanel;

SendPanel: TPanel;

```
  Log1: TMenuItem;
  N2: TMenuItem;
  Splitter1: TSplitter;
  LogList: TListBox;
  N500sec1: TMenuItem;
  N200sec1: TMenuItem;
  N100sec1: TMenuItem;
  N50sec1: TMenuItem;
  N10sec1: TMenuItem;
  N005sec1: TMenuItem;
  ServerPause1: TMenuItem;
  Options1: TMenuItem;
  ClickTimer: TTimer;
  ResponseTimer: TTimer;
  UpdateChanges1: TMenuItem;
  RefreshComplete1: TMenuItem;
  View1: TMenuItem;
  Disconnect1: TMenuItem;
  Exit1: TMenuItem;
  N1: TMenuItem;
  Connect1: TMenuItem;
  File1: TMenuItem;
  MainMenu1: TMainMenu;
  FormSettings1: TFormSettings;
  Timer1: TTimer;
  ClientSocket1: TClientSocket;
  Image1: TImage;
  ScrollBox1: TScrollBox;
  StatusBar1: TStatusBar;
  StatPanel: TPanel;
```

```
  SendPanel: TPanel;
  Panel1: TPanel;
  FileList1: TMenuItem;
  N5: TMenuItem;
  ScaleImage1: TMenuItem;
  N4: TMenuItem;
  Idle1: TMenuItem;
  Lowest1: TMenuItem;
  BelowNormal1: TMenuItem;
  Normal1: TMenuItem;
  AboveNormal1: TMenuItem;
  High1: TMenuItem;
  Critical1: TMenuItem;
  ServerPriority1: TMenuItem;
  Low1: TMenuItem;
  Medium1: TMenuItem;
  HighSlow1: TMenuItem;
  CompressionLevel1: TMenuItem;
  WaitImage: TImage;
  Default1: TMenuItem;
  Color24: TMenuItem;
  Gray8: TMenuItem;
  Gray4: TMenuItem;
  Color4: TMenuItem;
  Bitmapformat1: TMenuItem;
  FocusServerWindow1: TMenuItem;
  Special1: TMenuItem;
  Shutdown1: TMenuItem;
  N3: TMenuItem;
  Comm5tat1: TMenuItem;
```

```
    Help1: TMenuItem;
    About1: TMenuItem;
    StatBarMenu: TMenuItem;
    FullScreen1: TMenuItem;
    timerset: TTimer;
    procedure FormShow(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var
Action: TCloseAction);
    procedure ClientSocket1Lookup(Sender:
TObject;
      Socket: TCustomWinSocket);
    procedure
ClientSocket1Connecting(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket1Connect(Sender:
TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocket1Error(Sender:
TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode:
Integer);
    procedure Exit1Click(Sender: TObject);
    procedure Connect1Click(Sender:
TObject);
    procedure ClientSocket1Read(Sender:
TObject; Socket: TCustomWinSocket);
    procedure
ClientSocket1Disconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure Disconnect1Click(Sender:
TObject);
    procedure RefreshComplete1Click(Sender:
TObject);
    procedure UpdateChanges1Click(Sender:
TObject);
    procedure Image1MouseMove(Sender:
TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure ResponseTimerTimer(Sender:
TObject);
    procedure Image1MouseDown(Sender:
TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image1Click(Sender: TObject);
    procedure Image1MouseUp(Sender:
TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image1DblClick(Sender:
TObject);
    procedure ClickTimerTimer(Sender:
TObject);
    procedure PauseChange(Sender: TObject);
    procedure Log1Click(Sender: TObject);
    procedure CommStat1Click(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
    procedure Shutdown1Click(Sender:
TObject);
    procedure FormDestroy(Sender: TObject);
    procedure
FocusServerWindow1Click(Sender: TObject);
    procedure ColorClick(Sender: TObject);
    procedure CompClick(Sender: TObject);
    procedure PriorityClick(Sender: TObject);
    procedure ScaleImage1Click(Sender:
TObject);
    procedure ProcessList1Click(Sender:
TObject);
    procedure FileList1Click(Sender: TObject);
    procedure SendPanelResize(Sender:
TObject);
```

```
    Help1: TMenuItem;
    About1: TMenuItem;
    StatusBarMenu: TMenuItem;
    FullScreen1: TMenuItem;
    timerset: TTimer;
    procedure FormShow(Sender: TObject);
    procedure TimerTimer(Sender: TObject);
    procedure FormClose(Sender: TObject; var
Action: TCloseAction);
    procedure ClientSocket1Lookup(Sender:
TObject;
    Socket: TCustomWinSocket);
    procedure
ClientSocket1Connecting(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocket1Connect(Sender:
TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocket1Error(Sender:
TObject; Socket: TCustomWinSocket;
    ErrorEvent: TErrorEvent; var errorCode:
Integer);
    procedure Exit1Click(Sender: TObject);
    procedure Connect1Click(Sender:
TObject);
    procedure ClientSocket1Read(Sender:
TObject; Socket: TCustomWinSocket);
    procedure
ClientSocket1Disconnect(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure Disconnect1Click(Sender:
TObject);
    procedure RefreshComplete1Click(Sender:
TObject);
    procedure UpdateChanges1Click(Sender:
TObject);
```

```
    procedure SendPanelSize(Sender:
TObject);
    procedure FileList1Click(Sender: TObject);
    procedure ProcessList1Click(Sender:
TObject);
    procedure ScaleImage1Click(Sender:
TObject);
    procedure PriorityClick(Sender: TObject);
    procedure CompClick(Sender: TObject);
    procedure ColorClick(Sender: TObject);
    procedure
FocusServerWindow1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Shutdown1Click(Sender:
TObject);
    procedure FormCreate(Sender: TObject);
    procedure Comm1Start1Click(Sender:
TObject);
    procedure Log1Click(Sender: TObject);
    procedure PauseChange(Sender: TObject);
    procedure ClickTimerTimer(Sender:
TObject);
    procedure ImageDblClick(Sender:
TObject);
    Shift: TShiftState; X, Y: Integer);
    procedure ImageMouseUp(Sender:
TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
    procedure ImageClick(Sender: TObject);
    procedure ImageMouseDown(Sender:
TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
    procedure ResponseTimerTimer(Sender:
TObject);
    procedure ImageMouseMove(Sender:
TObject; Shift: TShiftState; X,
    Y: Integer);
```

```
  procedure About1Click(Sender: TObject);
  procedure StatBarMenuClick(Sender:
TObject);
  procedure FullScreen1Click(Sender:
TObject);
  procedure FormKeyDown(Sender:
TObject; var Key: Word;
   Shift: TShiftState);
  procedure timersetTimer(Sender:
TObject);
 protected
  NumRec    : double;
  NumSend   : double;
  CurMsg    : string;
  NeedReply : integer;
  LastX     : integer;
  LastY     : integer;
  t1        : DWORD;
  but       : integer;
  NumClick  : integer;
  MoveList  : TList;
  Anim      : integer;
  LastRec   : DWORD;
  ServerDelay: integer;
  ViewMode  : TViewMode;
  CompMode  : TCompressionLevel;
  SvrPriority: integer;
  ProcForm  : TForm;
  FileForm  : TForm;
  LastCPS   : string;
  BeforeFull : TRect;
  procedure SetStat(i: integer; s: string);
  procedure UpdateStats;
  procedure SendText(const Text: string);
  procedure Log(const s: string);
  procedure EnableButs;
  procedure ClearMoveList;
  procedure AddMove(x, y: integer);
  procedure ParseComLine;
  procedure StopAnim;
  procedure StartAnim;
  procedure EnableInput;
  procedure WMSysCommand(var
Message: TWMSysCommand); message
WM_SYSCOMMAND;
  function CanSendMenuMsg: boolean;
  procedure Send_Current_Settings;
  procedure ScaleXY(var X, Y: integer);
  procedure UpdateLogVis;
 public
  procedure SendMsg(MsgNum: integer;
const MsgData: string; Socket:
TCustomWinSocket);
  procedure ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);
  property Stat[i: integer]: string write
SetStat;
 end;
var
 ServerForm: TServerForm;
 step:integer;
 autorefresh:boolean=false;
implementation
uses ConnectDlg, ProcListDlg, FilesDlg,
About, FsTopDlg;
```

```
    procedure AboutClick(Sender: TObject);
    procedure SaveAsClick(Sender: TObject);

    procedure EditCopyClick(Sender:
TObject);

    procedure EditCutClick(Sender:
TObject);

    procedure FormKeyDown(Sender:
TObject; var Key: Word;
      Shift: TShiftState);

    procedure TimerTimer(Sender:
TObject);

  protected
    NumRec   : double;
    NumSend  : double;
    ComMsg   : string;
    MessageBy : integer;
    LastX    : integer;
    LastY    : integer;
    ft       : DWORD;
    pnt      : integer;
    LnmClick : integer;
    MdvsList : TList;
    Anim     : integer;
    LastRec  : DWORD;
    SavTextDelay: integer;
    ViewMode : TViewMode;
    ComboMode : TCombinationMode;
    SavTextIdx: integer;
    procTerm : TTerm;
    EndTerm  : TTerm;
    LastCPS  : string;
    BytesTell : TRect;
    procedure SetSize(i: integer; s: string);

    procedure WMSysCommand(var
Message: TWMSysCommand); message
WM_SYSCOMMAND;
    function  CanSendMemMsg: boolean;
    procedure SendTextMsg(Msg: string;
Socket:
      TCustomWinSocket);
    procedure ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);
      property Size[i: integer]: string
write
    procedure SetSize;
    end;
    var
      SavTextDelay: integer;
      ViewMode : TViewMode;
      ComboMode : TCombinationMode;
      SavTextIdx: integer;
      procTerm : TTerm;
      EndTerm  : TTerm;
      LastCPS  : string;
      BytesTell : TRect;
      procedure SetSize(i: integer; s: string);

      procedure AboutClick(Sender: TObject);
      procedure SaveAsClick(Sender: TObject);

      procedure EditCopyClick(Sender:
TObject);

      procedure EditCutClick(Sender:
TObject);

      procedure FormKeyDown(Sender:
TObject; var Key: Word;
        Shift: TShiftState);

      procedure TimerTimer(Sender:
TObject);

    implementation
    uses ConnectDlg, ProcListDlg, FileDlg,
AboutDlg;
    procedure TMainForm.UpdateSpeed;

    procedure TMainForm.SendText(const Text: string);

    procedure TMainForm.Log(const s: string);

    procedure TMainForm.EnableBtns;

    procedure TMainForm.CreateMovelist;

    procedure TMainForm.AddMove(x, y: integer);

    procedure TMainForm.ParseCmdLine;

    procedure TMainForm.StopAnim;

    procedure TMainForm.StartAnim;

    procedure TMainForm.EnableMenuItem;

    procedure TMainForm.WMSysCommand(var
Message: TWMSysCommand); message
WM_SYSCOMMAND;
    function  TMainForm.CanSendMemMsg: boolean;
    procedure TMainForm.SendTextMsg(Msg: string;
Socket:
      TCustomWinSocket);
    procedure TMainForm.ProcessMessage(const Msg:
string; Socket: TCustomWinSocket);
      property TMainForm.Size[i: integer]: string
write
    procedure TMainForm.SetSize;
    end;
    var
      SavTextDelay: integer;
      ViewMode : TViewMode;
      ComboMode : TCombinationMode;
      SavTextIdx: integer;
      procTerm : TTerm;
      EndTerm  : TTerm;
      LastCPS  : string;
      BytesTell : TRect;
      procedure SetSize(i: integer; s: string);
```

```
{$R *.DFM}

procedure TServerForm.FormShow(Sender:
TObject);

begin

  UpdateLogVis;

  if not ClientSocket1.Active then

    Timer1.Enabled := True;

end;

function IsDotAddress(const s: string):
boolean;

var

  i : integer;

begin

  Result := True;

  for i := 1 to Length(s) do

    if not (s[i] in ['0'..'9', '.']) then Result :=
False;

end;


procedure
TServerForm.Timer1Timer(Sender: TObject);

var

  f : TForm;

begin

  Timer1.Enabled := False;

  f := Self;

  with ClientConnectForm do begin

    Left := (f.Left + f.Width div 2) - Width div
2;

    Top  := (f.Top + f.Height div 2) - Height div
2;

    if ShowModal = mrOK then with
ClientSocket1 do begin

      if IsDotAddress(ServerCombo.Text) then
begin

        Host    := '';

        Address := ServerCombo.Text;

      end else begin

        Address := '';

        Host    := ServerCombo.Text;

      end;

      Port := StrToInt(PortEdit.Text);

      StartAnim;

      timerset.Enabled:=true;

      step:=step+1;

      Active := True;

    end;

  end;

end;

procedure TServerForm.FormClose(Sender:
TObject; var Action: TCloseAction);

begin

  if BorderStyle<>bsNone then
FormSettings1.SaveSettings;

  Disconnect1Click(nil);

end;

procedure
TServerForm.ClientSocket1Lookup(Sender:
TObject;

 Socket: TCustomWinSocket);

begin

  Stat[0] := ('Looking up: ' +
ClientSocket1.Host);

end;

procedure TServerForm.SetStat(I: integer; s:
string);
```

```
{$R *.DFM}

procedure TServerForm.FormShow(Sender:
TObject);

begin

  UpdateLogVis;

  if not ClientSocket1.Active then

    Timer1.Enabled := True;

end;

function IsDotAddress(const s: string):
boolean;

var

  i : integer;

begin

  Result := True;

  for i := 1 to Length(s) do

    if not (s[i] in ['0'..'9', '.']) then Result :=
False;

end;


procedure
TServerForm.Timer1Timer(Sender: TObject);

var

  f : TForm;

begin

  Timer1.Enabled := False;

  f := Self;

  with ClientConnectForm do begin

    Left := (f.Left + f.Width div 2) - Width div
2;

    Top  := (f.Top + f.Height div 2) - Height div
2;

    if ShowModal = mrOK then with
ClientSocket1 do begin
      if IsDotAddress(ServerCombo.Text) then
begin
        Host    := '';

        Address := ServerCombo.Text;

      end else begin

        Address := '';

        Host    := ServerCombo.Text;

      end;

      Port := StrToInt(PortEdit.Text);

      StartAdmin;

      timerset.Enabled:=True;

      step:=step+1;

      Active := True;

    end;

  end;

end;

procedure TServerForm.FormClose(Sender:
TObject; var Action: TCloseAction);

begin

  if BorderStyle<>bsNone then
FormSettings1.SaveSettings;

  Disconnect1Click(nil);

end;

procedure
TServerForm.ClientSocket1Lookup(Sender:
TObject;

  Socket: TCustomWinSocket);

begin

  Stat[0] := ('Looking up: ' +
ClientSocket1.Host);

end;

procedure TServerForm.SetStat(i: integer; s:
string);
```

```
begin

  FSTopForm.StatLabel.Caption := s;

  StatusBar1.Panels[i].Text := s;

  Update;

end;

procedure
TServerForm.ClientSocket1Connecting(Send
er: TObject;

 Socket: TCustomWinSocket);

begin

  Stat[0] := ('Connecting: ' +
ClientSocket1.Host);

end;

procedure
TServerForm.ClientSocket1Connect(Sender:
TObject;

 Socket: TCustomWinSocket);

begin

  Log(Format('%-7s %s', ['LogOn',
DateTimeToStr(Now)]));



  EnableButs;

  Stat[0] := ('Terhubung Dengan: ' +
Socket.RemoteHost);

  Caption := 'Aplikasi Screen Capture : ' +
Socket.RemoteHost;

  NumSend  := 0;

  NumRec   := 0;

  NeedReply := 0;

  StopAnim;

  EnableInput;

  //SendMsg(MSG_LOGON,
ClientConnectForm.PassEdit.Text,
ClientSocket1.Socket);

  Send_Current_Settings;

end;

procedure
TServerForm.ClientSocket1Error(Sender:
TObject;

 Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;

 var ErrorCode: Integer);

begin

  Stat[0] := ('Error: ' + IntToStr(ErrorCode));

  ErrorCode := 0;

  if not Socket.Connected then StopAnim;

end;

procedure TServerForm.Exit1Click(Sender:
TObject);

begin

  Close;

end;

procedure
TServerForm.Connect1Click(Sender:
TObject);

begin

  Image1.Picture.Bitmap := nil;

  Timer1Timer(nil);

end;

procedure TServerForm.SendMsg(MsgNum:
integer; const MsgData: string; Socket:
TCustomWinSocket);

var

  s : string;

begin

  Log(Format('%-7s #%2.2d', ['Send',
MsgNum]));

  Stat[0] := Format('Sending Message (Len =
%1.0n)', [Length(MsgData)+0.0]);
```

```
begin

  FSTopForm.StatLabel.Caption := s;

  StatusBar1.Panels[i].Text := s;

Update;

end;

procedure
TServerForm.ClientSocket1Connecting(Send
er: TObject;

  Socket: TCustomWinSocket);

begin

  Stat[0] := ('Connecting: ' +
ClientSocket1.Host);

end;

procedure
TServerForm.ClientSocket1Connect(Sender:
TObject;

  Socket: TCustomWinSocket);

begin

  Log(Format('%-7s %s', ['LogOn',
DateTimeToStr(Now)]));


  EnableBtns;

  Stat[0] := ('Terhubung Dengan: ' +
Socket.RemoteHost);

  Caption := 'Aplikasi Screen Capture : ' +
Socket.RemoteHost;

  NumSend  := 0;

  NumRec   := 0;

  NeedReply := 0;

  StopAnim;

  EnableInput;

  \\SendMsg(MSG_LOGON,
ClientConnectForm.PassEdit.Text +
ClientSocket1.Socket);

  Send_Current_Settings;

end;

procedure
TServerForm.ClientSocket1Error(Sender:
TObject;

  Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent;

  var ErrorCode: Integer);

begin

  Stat[0] := ('Error: ' + IntToStr(ErrorCode));

  ErrorCode := 0;

  if not Socket.Connected then StopAnim;

end;

procedure TServerForm.Exit1Click(Sender:
TObject);

begin

  Close;

end;

procedure
TServerForm.Connect1Click(Sender:
TObject);

begin

  Image1.Picture.Bitmap := nil;

  Timer1.Timer(nil);

end;

procedure TServerForm.SendMsg(MsgNum:
Integer; const MsgData: string; Socket:
TCustomWinSocket);

var

  s : string;

begin

  Log(Format('%-7s %2.2d', ['Send',
MsgNum]));

  Stat[0] := Format('Sending Message (Len =
%d.0n)', [Length(MsgData)+0.0]);
```

```
  s := IntToByteStr(MsgNum) +
IntToByteStr(Length(MsgData)) + MsgData;

  Socket.SendText(s);

  NumSend := NumSend + Length(s);

  UpdateStats;

  Inc(NeedReply);

  StartAnim;

end;

procedure TServerForm.UpdateStats;

begin

  // Stat[0] := Format('Sent: %1.0n',
[NumSend]);

  // Stat[1] := Format('Recv: %1.0n',
[NumRec]);

end;

procedure
TServerForm.ClientSocket1Read(Sender:
TObject;

  Socket: TCustomWinSocket);

var

  s    : string;

  msg   : integer;

  len   : integer;

  PerStr : string;

  tdif   : double;

  cps   : string;

begin

  // WaitImage.Hint := 'Data Last Received:' +
#13#10 + CurTime;

  s := Socket.ReceiveText;

  NumRec := NumRec + Length(s);

  UpdateStats;

  if CurMsg = '' then LastRec := GetTickCount;

  CurMsg := CurMsg + s;

  if Length(CurMsg) >= 8 then begin

    Move(CurMsg[1], msg, sizeof(integer));

    Move(CurMsg[5], len, sizeof(integer));

    PerStr := Format('(%1.0n%%)',
[Length(CurMsg) / (len + 8.0) * 100.0]);

    tdif := (GetTickCount - LastRec) / 1000.0;

    if tdif > 0.5 then cps := Format('%1.0n
cps', [Length(CurMsg) / tdif])

      else cps := '';

    Stat[0] := Format('Received: %1.0n of
%1.0n  %s  %s',

      [Length(CurMsg) + 0.0, len + 8.0, PerStr,
cps]);

    LastCPS := cps;

  end else begin

    if Length(s) > 0 then

      Stat[0] := 'Received: ' +
IntToStr(Length(CurMsg));

  end;

  while IsValidMessage(CurMsg) do

  begin

    s := TrimFirstMsg(CurMsg);

    ProcessMessage(s, Socket);

  end;

 if autorefresh=true then
SendMsg(MSG_SCREEN_UPDATE, '',
ClientSocket1.Socket);

end;

procedure
TServerForm.ClientSocket1Disconnect(Sende
r: TObject;

  Socket: TCustomWinSocket);

begin
```

```
  s := IntToByteStr(MsgNum) +
IntToByteStr(Length(MsgData)) + MsgData;

  Socket.SendText(s);

  NumSend := NumSend + Length(s);

  UpdateStats;

  Inc(NeedReply);

  StartAnim;

end;

procedure TServerForm.UpdateStats;

begin

  // Stat[0] := Format('Sent: %1.0n',
[NumSend]);

  // Stat[1] := Format('Recv: %1.0n',
[NumRec]);

end;

procedure
TServerForm.ClientSocketRead(Sender:
TObject;

  Socket: TCustomWinSocket);

var

  s   : string;

  msg  : integer;

  len  : integer;

  PerStr : string;

  tdif  : double;

  cps  : string;

begin

  // WaitMsg.Hint := 'Data Last Received: ' +
#13#10 + CurTime;

  s := Socket.ReceiveText;

  NumRec := NumRec + Length(s);

  UpdateStats;

  if CurMsg = '' then LastRec := GetTickCount;

  CurMsg := CurMsg + s;

  while IsValidMessage(CurMsg) do

  begin

    s := TrimFirstMsg(CurMsg);

    ProcessMessage(s, Socket);

  end;

  if autorefresh=true then
    SendMsg(MSG_SCREEN_UPDATE, '',
ClientSocket.Socket);

end;

procedure
TServerForm.ClientSocketDisconnect(Sender
: TObject;

  Socket: TCustomWinSocket);

begin

  Stat[0] := 'Received: ' +
IntToStr(Length(CurMsg));

  if Length(s) > 0 then

  begin

    Move(s[1], msg, sizeof(integer));

    Move(CurMsg[5], len, sizeof(integer));

    PerStr := Format('%1.0n%%',
[Length(CurMsg) / (len + 8.0) * 100.0]);

    tdif := (GetTickCount - LastRec) / 1000.0;

    if tdif > 0.5 then cps := Format('%1.0n
cps', [Length(CurMsg) / tdif])

    else cps := '';

    Stat[0] := Format('Received: %1.0n of
%1.0n  %s  %s',

      [Length(CurMsg) + 0.0, len + 8.0, PerStr,
cps]);

    LastCPS := cps;

  end else begin

    Stat[0] := 'Received: ' +
IntToStr(Length(CurMsg));

  end;

  while IsValidMessage(CurMsg) do

  begin

    s := TrimFirstMsg(CurMsg);

    ProcessMessage(s, Socket);

  end;

  if autorefresh=true then
    SendMsg(MSG_SCREEN_UPDATE, '',
ClientSocket.Socket);

end;

procedure
TServerForm.ClientSocketDisconnect(Sende
r: TObject;

  Socket: TCustomWinSocket);

begin
```

```
  Log(Format('%-7s %s', ['LogOff',
DateTimeToStr(Now)]));

  ClientSocket1.Active := False;

  EnableButs;

  Stat[0] := ('Disconnected: ' +
Socket.RemoteHost);

  Caption := 'Remote Control Client';

  StopAnim;

end;

procedure
TServerForm.Disconnect1Click(Sender:
TObject);

begin

  Stat[0] := 'Disconnecting...';

  ClientSocket1.Active := False;

  EnableButs;

  StopAnim;

end;

procedure
TServerForm.RefreshComplete1Click(Sender:
TObject);

begin

  SendMsg(MSG_REFRESH, '',
ClientSocket1.Socket);

end;


procedure
TServerForm.UpdateChanges1Click(Sender:
TObject);

begin

  SendMsg(MSG_SCREEN_UPDATE, '',
ClientSocket1.Socket);

end;

procedure
TServerForm.Image1MouseMove(Sender:
TObject; Shift: TShiftState;
 X, Y: Integer);

begin

  ScaleXY(X, Y);

  LastX := X;

  LastY := Y;


  AddMove(X, Y);

end;

procedure TServerForm.AddMove(x, y:
integer);

var

  MoveObj  : TMoveObj;

begin

  MoveObj := TMoveObj.Create;

  MoveObj.X := X;

  MoveObj.Y := Y;

  MoveObj.Time := GetTickCount;

  MoveList.Add(MoveObj);

end;

procedure
TServerForm.ResponseTimerTimer(Sender:
TObject);

var

  bm : TBitmap;

  x, y  : integer;

begin

  WaitImage.Hint := Format('Wait: %3.1n
seconds', [(GetTickCount-t1)/1000.0]);

  bm := TBitmap.Create;

  bm.Width := WaitImage.Width;

  bm.Height := WaitImage.Height;

  Anim := Anim + 1;
```

```
Log(Format('%-7s %s', ['LogOff',
DateTimeToStr(Now)]));

ClientSocket1.Active := False;

EnableButs;

Stat[0] := ('Disconnected: ' +
Socket.RemoteHost);

Caption := 'Remote Control Client';

StopAnim;

end;

procedure
TServerForm.Disconnect1Click(Sender:
TObject);

begin

Stat[0] := 'Disconnecting...';

ClientSocket1.Active := False;

EnableButs;

StopAnim;

end;

procedure
TServerForm.RefreshComplete1Click(Sender:
TObject);

begin

SendMsg(MSG_REFRESH, '',
ClientSocket1.Socket);

end;



procedure
TServerForm.UpdateChanges1Click(Sender:
TObject);

begin

SendMsg(MSG_SCREEN_UPDATE, '',
ClientSocket1.Socket);

end;

procedure
TServerForm.Image1MouseMove(Sender:
TObject; Shift: TShiftState;
X, Y: Integer);

begin

ScaleXY(X, Y);

LastX := X;

LastY := Y;

.

AddMove(X, Y);

end;

procedure TServerForm.AddMove(x, y:
integer);

var

MoveObj : TMoveObj;

begin

MoveObj := TMoveObj.Create;

MoveObj.X := X;

MoveObj.Y := Y;

MoveObj.Time := GetTickCount;

MoveList.Add(MoveObj);

end;

procedure
TServerForm.ResponseTimerTimer(Sender:
TObject);

var

bm : TBitmap;

x, y : integer;

begin

WaitImage.Hint := Format('Wait: %3.1n
seconds', [(GetTickCount-t1)/1000.0]);

bm := TBitmap.Create;

bm.Width := WaitImage.Width;

bm.Height := WaitImage.Height;

Anim := Anim + 1;
```

```
  Anim := Anim and 31;

  for x := -1 to 1 do

    for y := -1 to 1 do

      bm.Canvas.Draw(Anim + x*32, Anim +
y*32, Application.Icon);

  WaitImage.Picture.Assign(bm);

  bm.Free;

end;

procedure
TServerForm.Image1MouseDown(Sender:
TObject;

 Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);

begin

  ScaleXY(X, Y);

  but := 1;

  if Button = mbRight then but := 2;

  ClearMoveList;

  AddMove(x, y);

end;

procedure TServerForm.Image1Click(Sender:
TObject);

begin

  NumClick := 1;

  ClickTimer.Enabled := True;

end;

procedure
TServerForm.Image1MouseUp(Sender:
TObject; Button: TMouseButton;

 Shift: TShiftState; X, Y: Integer);

begin

  ScaleXY(X, Y);

  if but = 2 then begin

    // Only do this for Right Clicks

    SendMsg(MSG_CLICK, IntToByteStr(LastX)
+ IntToByteStr(LastY) +

      IntToByteStr(1 {Single}) +
IntToByteStr(but), ClientSocket1.Socket);

  end;

  AddMove(x, y);

end;

procedure
TServerForm.Image1DblClick(Sender:
TObject);

begin

  NumClick := 2;

  ClickTimer.Enabled := True;

end;

procedure
TServerForm.ClickTimerTimer(Sender:
TObject);

var

  s      : string;

  MoveObj  : TMoveObj;

  i      : integer;

begin

  ClickTimer.Enabled := False;

  if (MoveList.Count < 5) or (NumClick = 2)
then begin

    // This is a Click or Double-click

    SendMsg(MSG_CLICK, IntToByteStr(LastX)
+ IntToByteStr(LastY) +

      IntToByteStr(NumClick) +
IntToByteStr(but), ClientSocket1.Socket);

  end else begin

    // This is a "drag" operation

    s := IntToByteStr(but) +
IntToByteStr(MoveList.Count);
```

```
Anim := Anim and 31;

for x := -1 to 1 do

  for y := -1 to 1 do

    bm.Canvas.Draw(Anim + x*32, Anim +
y*32, Application.Icon);

WallImage.Picture.Assign(bm);

bm.Free;

end;

procedure
TServerForm.Image1MouseDown(Sender:
TObject;

Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);

begin

  ScaleXY(X, Y);

  but := 1;

  if Button = mbRight then but := 2;

  ClearMoveList;

  AddMove(x, y);

end;

procedure TServerForm.Image1Click(Sender:
TObject);

begin

  NumClick := 1;

  ClickTimer.Enabled := True;

end;

procedure
TServerForm.Image1MouseUp(Sender:
TObject; Button: TMouseButton;

  Shift: TShiftState; X, Y: Integer);

begin

  ScaleXY(X, Y);

  if but = 2 then begin

    \\ Only do this for Right Clicks

    SendMsg(MSG_CLICK, IntToBytestr(LastX)
+ IntToBytestr(LastY) +

    IntToBytestr(1 {Single}) +
IntToBytestr(but), ClientSocket1.Socket);

  end;

  AddMove(x, y);

end;

procedure
TServerForm.Image1DblClick(Sender:
TObject);

begin

  NumClick := 2;

  ClickTimer.Enabled := True;

end;

procedure
TServerForm.ClickTimerTimer(Sender:
TObject);

var

  s     : string;

  MoveObj  : TMoveObj;

  i     : integer;

begin

  ClickTimer.Enabled := False;

  if (MoveList.Count < 5) or (NumClick = 2)
then begin

    \\ This is a Click or Double-click

    SendMsg(MSG_CLICK, IntToBytestr(LastX)
+ IntToBytestr(LastY) +

    IntToBytestr(NumClick) +
IntToBytestr(but), ClientSocket1.Socket);

  end else begin

    \\ This is a "drag" operation

    s := IntToBytestr(but) +
IntToBytestr(MoveList.Count);
```

```
    for i := 0 to MoveList.Count-1 do begin
      MoveObj := MoveList[i];
      s := s + IntToByteStr(MoveObj.X) +
IntToByteStr(MoveObj.Y)
        + IntToByteStr(MoveObj.time);
    end;
    SendMsg(MSG_DRAG, s,
ClientSocket1.Socket);
  end;
end;
procedure TServerForm.Log1Click(Sender:
TObject);
begin
  Log1.Checked := not Log1.Checked;
  UpdateLogVis;
end;
procedure TServerForm.UpdateLogVis;
begin
  LogList.Visible := Log1.Checked;
  Splitter1.Visible := Log1.Checked;
  if Log1.Checked then
    LogList.Left := Splitter1.Left - 1;
end;
procedure TServerForm.Log(const s: string);
begin
  LogList.ItemIndex := LogList.Items.Add(s);
end;
procedure
TServerForm.CommStat1Click(Sender:
TObject);
begin
  CommStat1.Checked := not
CommStat1.Checked;
  StatPanel.Visible := CommStat1.Checked;
end;
procedure TServerForm.EnableButs;
var
  b : boolean;
begin
  b := ClientSocket1.Active;
  Connect1.Enabled := not b;
  Disconnect1.Enabled := b;
end;
procedure TServerForm.FormCreate(Sender:
TObject);
begin
  EnableButs;
  MoveList := TList.Create;
  ParseComLine;
  StopAnim;
  EnableInput;
  ServerDelay := DEFAULT_SERVER_DELAY;
  ViewMode   := DEFAULT_VIEW_MODE;
  CompMode   := DEFAULT_COMP_MODE;
  SvrPriority := DEFAULT_SVR_PRIORITY;
end;
procedure
TServerForm.Shutdown1Click(Sender:
TObject);
begin
  Close;
  Application.MainForm.Close;
end;
procedure
TServerForm.FormDestroy(Sender: TObject);
```

```
for i := 0 to MoveList.Count-1 do begin
  MoveObj := MoveList[i];
  s := s + IntToBytesStr(MoveObj.X) +
IntToBytesStr(MoveObj.Y)
    + IntToBytesStr(MoveObj.time);
end;
SendMsg(MSG_DRAG, s,
ClientSocket1.Socket);
end;
end;
procedure TServerForm.Log1Click(Sender:
TObject);
begin
  Log1.Checked := not Log1.Checked;
  UpdateLogVis;
end;
procedure TServerForm.UpdateLogVis;
begin
  LogList.Visible := Log1.Checked;
  Splitter1.Visible := Log1.Checked;
  if Log1.Checked then
    LogList.Left := Splitter1.Left - 1;
end;
procedure TServerForm.Log(const s: string);
begin
  LogList.ItemIndex := LogList.Items.Add(s);
end;
procedure
TServerForm.Comm Start1Click(Sender:
TObject);
begin
  Comm Start1.Checked := not
Comm Start1.Checked;
  StartPanel.Visible := Comm Start1.Checked;
end;
procedure TServerForm.EnableBuns;
var
  b : boolean;
begin
  b := ClientSocket1.Active;
  Connect1.Enabled := not b;
  Disconnect1.Enabled := b;
end;
procedure TServerForm.FormCreate(Sender:
TObject);
begin
  EnableBuns;
  MoveList := TList.Create;
  ParseCommLine;
  StopAnim;
  EnableInput;
  ServerDelay := DEFAULT_SERVER_DELAY;
  ViewMode   := DEFAULT_VIEW_MODE;
  CompMode   := DEFAULT_COMP_MODE;
  SvrPriority := DEFAULT_SVR_PRIORITY;
end;
procedure
TServerForm.Shutdown1Click(Sender:
TObject);
begin
  Close;
  Application.MainForm.Close;
end;
procedure
TServerForm.FormDestroy(Sender: TObject);
```

```
begin
  ClearMoveList;
  MoveList.Free;
end;
procedure TServerForm.ClearMoveList;
var
  i : integer;
begin
  for i := 0 to MoveList.Count-1 do
    TObject(MoveList[i]).Free;
  MoveList.Clear;
end;
procedure
TServerForm.FocusServerWindow1Click(Sen
der: TObject);
begin
  SendMsg(MSG_FOCUS_SERVER, '',
ClientSocket1.Socket);
end;
procedure TServerForm.ParseComLine;
var
  i       : integer;
  s       : string;
begin
  for i := 1 to ParamCount do begin
    s := UpperCase(ParamStr(i));

    if s = '/CLIENT' then begin
      Visible := True;
    end;
  end;
end;
procedure TServerForm.EnableInput;
var
  b : boolean;
begin
  b := (NeedReply = 0) and
ClientSocket1.Active;
  Image1.Enabled := b;
  Special1.Enabled := b;
  // Options1.Enabled := b;
end;
procedure TServerForm.StopAnim;
var
  bmp : TBitmap;
begin
  Screen.Cursor := crDefault;
  ResponseTimer.Enabled := False;
  // Stat[2] := 'Not Waiting';
  bmp := TBitmap.Create;
  bmp.Width := WaitImage.Width;
  bmp.Height := WaitImage.Height;
  bmp.Canvas.Draw(2, 2, Application.Icon);
  WaitImage.Picture.Assign(bmp);
  bmp.Free;
    EnableInput;
end;
procedure TServerForm.StartAnim;
begin
  Anim := 2;
  ResponseTimer.Enabled := True;
  // Stat[2] := 'Waiting';
  t1 := GetTickCount;
```

```
begin
  ClearMoveList;
  MoveList.Free;
end;

procedure TServerForm.ClearMoveList;
var
  i : integer;
begin
  for i := 0 to MoveList.Count-1 do
    TObject(MoveList[i]).Free;
  MoveList.Clear;
end;

procedure
TServerForm.FocusServerWindow1Click(Sen
der: TObject);
begin
  SendMsg(MSG_FOCUS_SERVER, '',
ClientSocket1.Socket);
end;

procedure TServerForm.ParseComLine;
var
  i     : integer;
  s     : string;
begin
  for i := 1 to ParamCount do begin
    s := UpperCase(ParamStr(i));

    If s = '/CLIENT' then begin
      Visible := True;
    end;
  end;
end;

procedure TServerForm.EnableInput;
var
  b : boolean;
begin
  b := (NeedReply = 0) and
ClientSocket1.Active;
  Image1.Enabled := b;
  Special1.Enabled := b;
  // Options1.Enabled := b;
end;

procedure TServerForm.StopAnim;
var
  bmp : TBitmap;
begin
  Screen.Cursor := crDefault;
  ResponseTimer.Enabled := False;
  // Stat[2] := 'Not Waiting';
  bmp := TBitmap.Create;
  bmp.Width := WaitImage.Width;
  bmp.Height := WaitImage.Height;
  bmp.Canvas.Draw(2, 2, Application.Icon);
  WaitImage.Picture.Assign(bmp);
  bmp.Free;
  EnableInput;
end;

procedure TServerForm.StartAnim;
begin
  Anim := 2;
  ResponseTimer.Enabled := True;
  // Stat[2] := 'Waiting';
  t1 := GetTickCount;
```

```
  // Screen.Cursor := crAppStart;

  EnableInput;

end;

procedure
TServerForm.WMSysCommand(var Message:
TWMSysCommand);

begin

  if (Message.CmdType and $FFF0 =
SC_MINIMIZE) then

    Application.Minimize

  else

    inherited;

end;

function TServerForm.CanSendMenuMsg:
boolean;

begin

  Result := ClientSocket1.Active;

end;

procedure
TServerForm.PauseChange(Sender: TObject);

var

  d : integer;

begin

  d := 0;

  (Sender as TMenuItem).Checked := True;

  if Sender = N005sec1 then d :=  50;

  if Sender = N010sec1 then d :=  100;

  if Sender = N050sec1 then d :=  500;

  if Sender = N100sec1 then d := 1000;

  if Sender = N200sec1 then d := 2000;

  if Sender = N500sec1 then d := 5000;

  ServerDelay := d;

  if CanSendMenuMsg then

    SendMsg(MSG_SEVER_DELAY,
IntToByteStr(d), ClientSocket1.Socket);

end;

procedure TServerForm.ColorClick(Sender:
TObject);

var

  vm : TViewMode;

  x  : integer;

begin

  (Sender as TMenuItem).Checked := True;

  vm := vmDefault;

  if Sender = Color4   then vm := vmColor4;

  if Sender = Gray4    then vm := vmGray4;

  if Sender = Gray8    then vm := vmGray8;

  if Sender = Color24  then vm := vmColor24;

  if Sender = Default1 then vm := vmDefault;

  ViewMode := vm;

  if CanSendMenuMsg then begin

    x := integer(vm);

    SendMsg(MSG_VIEW_MODE,
IntToByteStr(x), ClientSocket1.Socket);

    SendMsg(MSG_REFRESH, '',
ClientSocket1.Socket);

  end;

end;

procedure TServerForm.CompClick(Sender:
TObject);

var

  cm : TCompressionLevel;

begin

  (Sender as TMenuItem).Checked := True;
```

```
\\ Screen.Cursor := crAppStart;

EnableInput;

end;

procedure
TServerForm.WMSysCommand(var Message:
TWMSysCommand);

begin

  if (Message.CmdType and $FFF0 =
SC_MINIMIZE) then

    Application.Minimize

  else

    inherited;

end;

function TServerForm.CanSendMenuMsg:
boolean;

begin

  Result := ClientSocket1.Active;

end;

procedure
TServerForm.PauseChange(Sender: TObject);

var

  d : integer;

begin

  d := 0;

  (Sender as TMenuItem).Checked := True;

  if Sender = N50Sec1 then d :=  50;

  if Sender = N100Sec1 then d :=  100;

  if Sender = N250Sec1 then d :=  500;

  if Sender = N1000Sec1 then d := 1000;

  if Sender = N2000Sec1 then d := 2000;

  if Sender = N5000Sec1 then d := 5000;

  ServerDelay := d;

  if CanSendMenuMsg then

    SendMsg(MSG_SEVER_DELAY,
IntToByteStr(d), ClientSocket1.Socket);

end;

procedure TServerForm.ColorClick(Sender:
TObject);

var

  vm : TViewMode;

  x : integer;

begin

  (Sender as TMenuItem).Checked := True;

  vm := vmDefault;

  if Sender = Color4  then vm := vmColor4;

  if Sender = Gray4   then vm := vmGray4;

  if Sender = Gray8   then vm := vmGray8;

  if Sender = Color24 then vm := vmColor24;

  if Sender = Default1 then vm := vmDefault;

  ViewMode := vm;

  if CanSendMenuMsg then begin

    x := integer(vm);

    SendMsg(MSG_VIEW_MODE,
IntToByteStr(x), ClientSocket1.Socket);

    SendMsg(MSG_REFRESH, '',
ClientSocket1.Socket);

  end;

end;

procedure TServerForm.CompClick(Sender:
TObject);

var

  cm : TCompressionLevel;

begin

  (Sender as TMenuItem).Checked := True;
```

```
  cm := clDefault;

  if Sender = HighSlow1 then cm := clMax;

  if Sender = Medium1  then cm := clDefault;

  if Sender = LowFast1  then cm := clFastest;

  CompMode := cm;

  if CanSendMenuMsg then

    SendMsg(MSG_COMP_MODE,
IntToByteStr(integer(cm)),
ClientSocket1.Socket);

end;

procedure
TServerForm.ScaleImage1Click(Sender:
TObject);

begin

  ScaleImage1.Checked := not
ScaleImage1.Checked;

  if ScaleImage1.Checked then begin

    Image1.AutoSize := False;

    Image1.Stretch  := True;

    Image1.Align    := alClient;

  end else begin

    Image1.AutoSize := True;

    Image1.Stretch  := False;

    Image1.Align    := alNone;


Image1.Picture.Assign(Image1.Picture.Graphi
c); // To trigger the Autosize property

  end;

end;

procedure TServerForm.ScaleXY(var X, Y:
integer);

begin

  if not ScaleImage1.Checked then exit;

  with Image1 do begin

    X := X * Picture.Width div Width;

    Y := Y * Picture.Height div Height;

  end;

end;

procedure
TServerForm.ProcessList1Click(Sender:
TObject);

begin

  SendMsg(MSG_PROCESS_LIST, '',
ClientSocket1.Socket);

end;

procedure
TServerForm.FileList1Click(Sender: TObject);

begin

  SendMsg(MSG_DRIVE_LIST, '',
ClientSocket1.Socket);

end;

procedure TServerForm.About1Click(Sender:
TObject);

begin

  AboutBox.ShowModal;

end;

procedure
TServerForm.StatBarMenuClick(Sender:
TObject);

begin

  StatBarMenu.Checked := not
StatBarMenu.Checked;

  StatusBar1.Visible :=
StatBarMenu.Checked;

end;

procedure
TServerForm.FullScreen1Click(Sender:
TObject);

begin

  if BorderStyle = bsSizeable then begin
```

```
  cm := clDefault;

  if Sender = High5low1 then cm := clMax;

  if Sender = Medium1  then cm := clDefault;

  if Sender = Low1ast1  then cm := clFastest;

  CompMode := cm;

  if CanSendMenuMsg then

    SendMsg(MSG_COMP_MODE,
  IntToByteStr(integer(cm)),
  Client5ocket1.Socket);

end;

procedure
TServerForm.ScaleImage1Click(Sender:
TObject);

begin

  ScaleImage1.Checked := not
  ScaleImage1.Checked;

  if ScaleImage1.Checked then begin

    Image1.AutoSize := False;

    Image1.Stretch  := True;

    Image1.Align    := alClient;

  end else begin

    Image1.AutoSize := True;

    Image1.Stretch  := False;

    Image1.Align    := alNone;


  Image1.Picture.Assign(Image1.Picture.Graphi
  c); \\ To trigger the Autosize property

  end;

end;

procedure TServerForm.ScaleXY(var X, Y:
integer);

begin

  if not ScaleImage1.Checked then exit;

  with Image1 do begin

    X := X * Picture.Width div Width;

    Y := Y * Picture.Height div Height;

  end;

end;

procedure
TServerForm.ProcessList1Click(Sender:
TObject);

begin

  SendMsg(MSG_PROCESS_LIST, '',
  Client5ocket1.Socket);

end;

procedure
TServerForm.FileList1Click(Sender: TObject);

begin

  SendMsg(MSG_DRIVE_LIST, '',
  Client5ocket1.Socket);

end;

procedure TServerForm.About1Click(Sender:
TObject);

begin

  AboutBox.ShowModal;

end;

procedure
TServerForm.StatBarMenuClick(Sender:
TObject);

begin

  StatBarMenu.Checked := not
  StatBarMenu.Checked;

  StatusBar1.Visible :=
  StatBarMenu.Checked;

end;

procedure
TServerForm.FullScreen1Click(Sender:
TObject);

begin

  if BorderStyle = bsSizeable then begin
```

```
    BeforeFull := BoundsRect;
    Menu       := nil;
    Left       := 0;
    Top        := 0;
    Width      := Screen.Width;
    Height     := Screen.Height;
    BorderStyle := bsNone;
    StatPanel.Visible := False;
    StatusBar1.Visible := False;
    ScrollBox1.BorderStyle := bsNone;
    FSTopForm.Show;
  end else begin
    BoundsRect := BeforeFull;
    Menu := MainMenu1;
    BorderStyle := bsSizeable;
    StatPanel.Visible := True;
    StatusBar1.Visible := True;
    ScrollBox1.BorderStyle := bsSingle;
    FSTopForm.Hide;
  end;
end;

procedure
TServerForm.FormKeyDown(Sender:
TObject; var Key: Word;
 Shift: TShiftState);
begin
  // If in Full-Screen mode, do an extra check
for Hot-Keys on the popup menu
  if BorderStyle = bsNone then begin
    FSTopForm.CheckShortCut(Key, Shift);
  end;
end;

procedure
TServerForm.timersetTimer(Sender:
TObject);

var   cm : TCompressionLevel;
  vm : TViewMode;
  x:integer;
begin
 if step=1 then
SendMsg(MSG_SEVER_DELAY,
IntToByteStr(50), ClientSocket1.Socket);
//respon
 if step=2 then
 begin
  cm := clfastest;
  CompMode := cm;
  x := integer(cm);
  SendMsg(MSG_COMP_MODE,
IntToByteStr(integer(x)),
ClientSocket1.Socket); //kompresi
 end;
 if step=3 then
 begin
    x := integer(vmcolor16);
   SendMsg(MSG_VIEW_MODE,
IntToByteStr(x), ClientSocket1.Socket);
   SendMsg(MSG_REFRESH, '',
ClientSocket1.Socket);
 end;
 if step=4 then
 begin
  autorefresh:=true;
  RefreshComplete1.Click;
  screen.Cursor:=crarrow;
 end;
```

```
  BeforeFull  := BoundsRect;
  Menu        := nil;
  Left        := 0;
  Top         := 0;
  Width       := Screen.Width;
  Height      := Screen.Height;
  BorderStyle := bsNone;
  StatPanel.Visible := False;
  StatusBar1.Visible := False;
  ScrollBox1.BorderStyle := bsNone;
  FSTopForm.Show;
end else begin
  BoundsRect := BeforeFull;
  Menu := MainMenu1;
  BorderStyle := bsSizeable;
  StatPanel.Visible := True;
  StatusBar1.Visible := True;
  ScrollBox1.BorderStyle := bsSingle;
  FSTopForm.Hide;
end;
end;

procedure
TServerForm.FormKeyDown(Sender:
TObject; var Key: Word;
  Shift: TShiftState);

begin
  \\ If in Full-Screen mode, do an extra check
  for Hot-Keys on the popup menu
  if BorderStyle = bsNone then begin
    FSTopForm.CheckShortCut(Key, Shift);
  end;
end;

procedure
TServerForm.timerestTimer(Sender:
TObject);

var  cm : TCompressionLevel;
  vm : TViewMode;
  x:integer;
begin
  if step=1 then
  SendMsg(MSG_SEVER_DELAY,
  IntToBytestr(50), ClientSocket1.Socket);
  \\response
  if step=2 then
  begin
    cm := clFastest;
    CompMode := cm;
    x := integer(cm);
    SendMsg(MSG_COMP_MODE,
    IntToBytestr(integer(x)),
    ClientSocket1.Socket); \\compressi
  end;
  if step=3 then
  begin
    x := integer(vmcolor16);
    SendMsg(MSG_VIEW_MODE,
    IntToBytestr(x), ClientSocket1.Socket);
    SendMsg(MSG_REFRESH, '',
    ClientSocket1.Socket);
  end;
  if step=4 then
  begin
    autorefresh:=true;
    RefreshComplete1.Click;
    screen.Cursor:=crarrow;
  end;
```

```
 if step=5 then FullScreen1.Click;
 if step=6 then timerset.Enabled:=false;
 step:=step+1;
end;
```

FORM CONNECT

```
unit ConnectDlg;
interface
uses
 Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
 FormSettings, StdCtrls, Buttons, Registry;
type
 TClientConnectForm = class(TForm)
   Label1: TLabel;
   Label2: TLabel;
   PortEdit: TEdit;
   ConnectBut: TBitBtn;
   CancelBut: TBitBtn;
   FormSettings1: TFormSettings;
   ServerCombo: TComboBox;
   Label3: TLabel;
   PassEdit: TEdit;
   StartScreenBox: TCheckBox;
   procedure FormClose(Sender: TObject; var
Action: TCloseAction);
   procedure FormShow(Sender: TObject);
 private
   { Private declarations }
   procedure  LoadCombo;
   procedure  SaveCombo;
 public
   { Public declarations }
 end;
var
 ClientConnectForm: TClientConnectForm;
implementation
{$R *.DFM}
procedure
TClientConnectForm.FormClose(Sender:
TObject;
 var Action: TCloseAction);
var
  s : string;
  i : integer;
begin
  s := ServerCombo.Text;
  i := ServerCombo.Items.IndexOf(s);
  if i <> -1 then ServerCombo.Items.Delete(i);
  ServerCombo.Items.Insert(0, s);
  while ServerCombo.Items.Count > 8 do
    ServerCombo.Items.Delete(8);
  ServerCombo.Text := s;
  SaveCombo;
  FormSettings1.SaveSettings;
end;
procedure
TClientConnectForm.FormShow(Sender:
TObject);
begin
  LoadCombo;
  ServerCombo.SetFocus;
end;
```

```
if step=5 then FullScreen1.Click;

if step=6 then timerset.Enabled:=false;

step:=step+1;

end;
```

FORM CONNECT

```
unit ConnectDlg;

interface

uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs,
  FormSettings, StdCtrls, Buttons, Registry;

type
  TClientConnectForm = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    PortEdit: TEdit;
    ConnectBut: TBitBtn;
    CancelBut: TBitBtn;
    FormSettings1: TFormSettings;
    ServerCombo: TComboBox;
    Label3: TLabel;
    PassEdit: TEdit;
    StartScreenBox: TCheckBox;
    procedure FormClose(Sender: TObject; var
  Action: TCloseAction);
    procedure FormShow(Sender: TObject);
    private
    { Private declarations }
    procedure LoadCombo;
    procedure SaveCombo;
  public
    { Public declarations }
  end;
  var
    ClientConnectForm: TClientConnectForm;

implementation

{$R *.DFM}

procedure
  TClientConnectForm.FormClose(Sender:
  TObject);

var Action: TCloseAction);

var
  s : string;
  i : integer;
begin
  s := ServerCombo.Text;
  i := ServerCombo.Items.IndexOf(s);
  if i <> -1 then ServerCombo.Items.Delete(i);
  ServerCombo.Items.Insert(0, s);
  while ServerCombo.Items.Count > 8 do
    ServerCombo.Items.Delete(8);
  ServerCombo.Text := s;
  SaveCombo;
  FormSettings1.SaveSettings;
end;

procedure
  TClientConnectForm.FormShow(Sender:
  TObject);
begin
  LoadCombo;
  ServerCombo.SetFocus;
end;
```

```
procedure TClientConnectForm.LoadCombo;

var

  ini : TRegIniFile;

begin

  ini := TRegIniFile.Create('Software\' +
Application.Title);

  ServerCombo.Items.CommaText :=
ini.ReadString('Servers', 'ServerCombo', '');

  ServerCombo.ItemIndex := 0;

  ini.Free;

end;

procedure TClientConnectForm.SaveCombo;

var

  ini : TRegIniFile;

begin

  ini := TRegIniFile.Create('Software\' +
Application.Title);

  ini.WriteString('Servers', 'ServerCombo',
ServerCombo.Items.CommaText);

  ini.Free;

end;

end.
```

```
procedure TClientConnectForm.LoadCombo;
var
  ini : TRegIniFile;
begin
  ini := TRegIniFile.Create('Software\' +
    Application.Title);
  ini.ReadStrings('Servers', 'ServerCombo', '');
  ServerCombo.ItemIndex := 0;
  ini.Free;
end;

procedure TClientConnectForm.SaveCombo;
var
  ini : TRegIniFile;
begin
  ini := TRegIniFile.Create('Software\' +
    Application.Title);
  ini.WriteStrings('Servers', 'ServerCombo',
    ServerCombo.Items.CommaText);
  ini.Free;
end;
```