# MEMBANGUN APLIKASI KOMUNIKASI BERBASIS *TEXT* DENGAN TEKNOLOGI *WINSOCK* DAN *UDP*

**Bambang Sugiantoro**
Staf Pengajar Jurusan Teknik Informatika UPN "Veteran" Yogyakarta
Jl Babarsari No : 2 TB , Yogyakarta
email: edo_lapis@yahoo.com, bambang05@if.upnyk.ac.id

***Abstrak***

*Salah satu manfaat dari jaringan komputer yaitu sebagai media komunikasi antar pengguna dalam suatu jaringan komputer. Komunikasi berbasis text merupakan salah satu bentuk nyata dari komunikasi jenis ini yang sudah banyak dirasakan manfaatnya oleh pengguna komputer di seluruh dunia. Aplikasi pendukungnya pun sudah banyak beredar di pasaran, namun karena alasan kehandalan protokol, teknologi pengiriman paket datanya rata-rata menggunakan protokol TCP. Menurut model referensi OSI maupun TCP/IP Protocol Suite pada Transport Layer terdapat dua protokol, yaitu: TCP dan UDP. Maka, dirasa perlu untuk mencoba membangun aplikasi komunikasi berbasis text dengan memanfaatkan UDP sebagai protokolnya.*

*Metodologi pengembangan sistem yang digunakan adalah metode GRAPPLE. Perangkat lunak yang digunakan dalam membangun aplikasi ini adalah Microsoft Visual Basic 6.0 sebagai media implementasi dan bahasa pemrograman, perangkat lunak ini menyediakan semua references/library dan components yang dibutuhkan dalam pembangunan aplikasi ini. Inno Setup Compiler versi 4.20 sebagai media untuk pembuatan program instalasi aplikasi. Seperti kebanyakan aplikasi sejenis lainnya yang berjalan di sistem operasi Windows, aplikasi ini juga memanfaatkan teknologi WinSock, maka aplikasi ini diharapkan mampu beroperasi di semua sistem operasi Windows.*

*Input dan output textnya menggunakan RichTextBox, sehingga output textnya dapat disimpan ke dalam format rtf. Adanya tools Invite Friends dengan memanfaatkan Messenger Service dari Windows NT/2000/XP. Feature tambahan lainnya yaitu Pinger Tool sebagai salah satu implementasi dari ICMP, yang dapat digunakan untuk memeriksa apakah suatu komputer host dalam keadaan hidup atau tidak juga untuk membangkitkan traffic test antara dua komputer host.*

***Kata Kunci :*** *Windows Sockets, WinSock, User Datagram Protocol, UDP.*

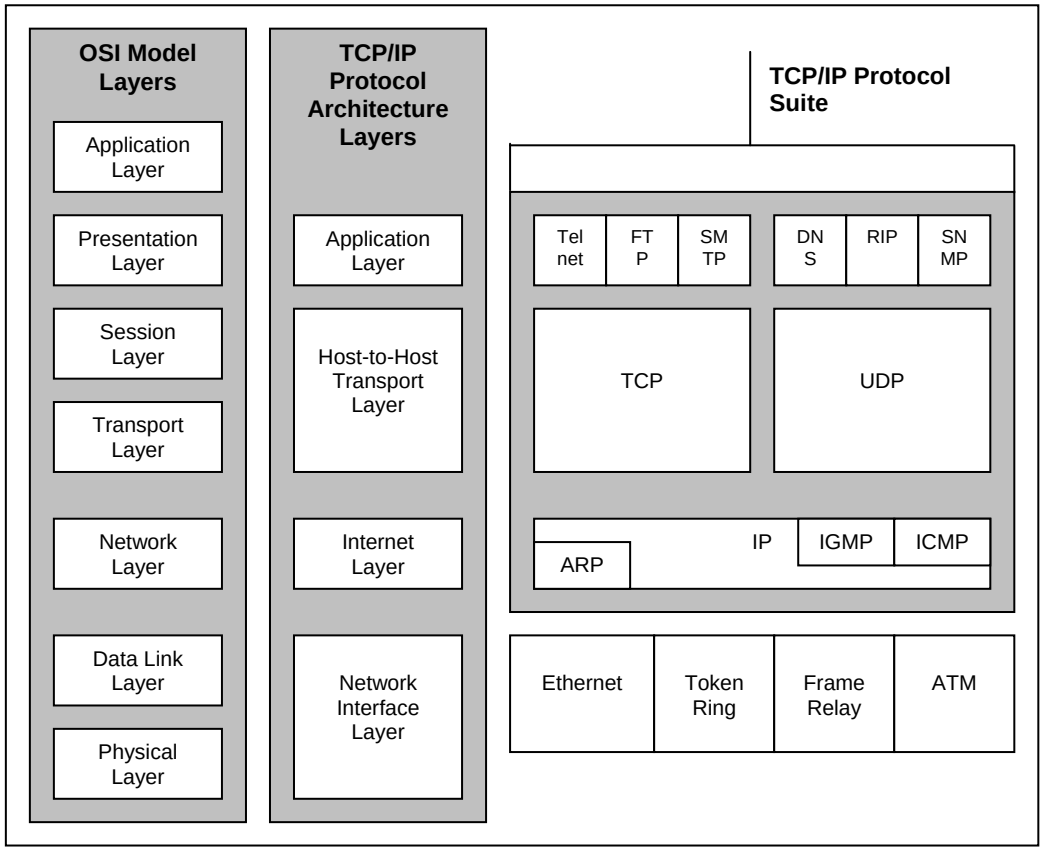
## 1. PENDAHULUAN

Jaringan komputer memberikan manfaat yang sangat besar bagi perkembangan teknologi informasi di dunia. Salah satu manfaat dari jaringan komputer yaitu sebagai media komunikasi antar pengguna dalam suatu jaringan komputer. Sistem komunikasi berbasis text merupakan salah satu bentuk nyata dari komunikasi jenis ini yang sudah banyak dirasakan manfaatnya sebagai media komunikasi oleh pengguna jaringan komputer di seluruh dunia. Aplikasi pendukungnya pun sudah banyak beredar di pasaran, namun karena alasan kehandalan protokol, teknologi pengiriman paket datanya rata-rata menggunakan protokol TCP. Padahal menurut model referensi OSI (*Open System Interconnection*) maupun *TCP/IP Protocol Suite* pada *Transport Layer* terdapat dua protokol, yaitu: *Transmission Control Protocol* (TCP) dan *User Datagram Protocol* (UDP). UDP jarang digunakan untuk aplikasi pengiriman paket data yang berkapasitas besar, karena protokol ini merupakan protokol yang "tidak andal" dalam hal ini. Jika suatu aplikasi diimplementasikan menggunakan UDP, maka aplikasi tersebut harus memiliki mekanisme *error recovery*-nya sendiri, walaupun hasil akhirnya masih juga belum sempurna.

## 2. LANDASAN TEORI

### 2.1. Model Referensi OSI dan TCP/IP

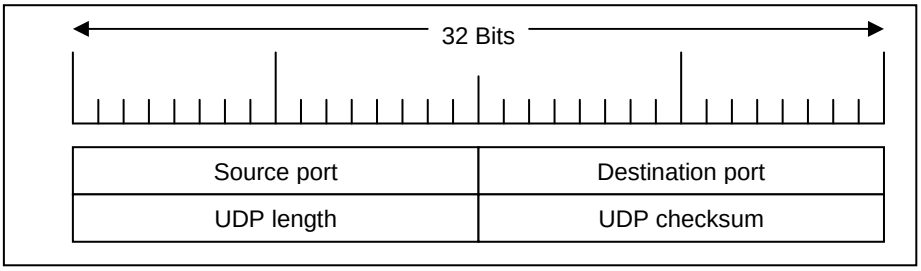
Hubungan antara Model Referensi OSI dengan TCP/IP dapat dilihat pada Gambar 1 (MSDN Library, 2004).




Gambar 1. Hubungan antara Model Referensi OSI dengan TCP/IP

### 2.2. User Datagram Protocol (UDP)

*User Datagram Protocol* (UDP) adalah protokol *transport* yang digunakan secara luas pada urutan kedua. Seperti *Transmission Control Protocol* (TCP), UDP menggunakan *port* dan menyediakan konektivitas *end-to-end* antara aplikasi *client* dan *server*. UDP merupakan protokol yang kecil dan efisien. Tetapi, berbeda dengan TCP, UDP tidak menjamin pengiriman datanya, aplikasi harus mengimplementasikan mekanisme *error recovery*-nya sendiri jika memerlukan mekanisme tersebut. Hal ini membuatnya cocok untuk beberapa aplikasi, tetapi tidak untuk beberapa yang lain (Mansfield, 2004). Segmen UDP terdiri dari *header 8 byte* yang diikuti oleh data, *header* UDP ini dapat dilihat pada Gambar 2 (Tanenbaum, 1997).




Gambar 2. Header UDP

UDP mirip dengan TCP dalam beberapa hal, diantaranya:

a. UDP adalah protokol *transport*: UDP hanya berhubungan dengan komunikasi antara dua *end-point* (misalnya aplikasi *client* pada komputer *host*, dan aplikasi *server* pada komputer *remote*). *Intermediate router* tidak berhubungan dengan data UDP dalam paket yang dikirimkannya, *router* hanya beroperasi pada layer IP atau *network lower-down*.
b. UDP menggunakan *port* untuk membedakan antara *traffic* dari banyak aplikasi UDP pada komputer yang sama, dan untuk mengirim paket yang tepat ke aplikasi yang sesuai (ini disebut *demultiplexing*). UDP dan *port*-nya menyediakan *interface* antara program aplikasi dan *layer networking IP*.

UDP berbeda dari TCP dalam beberapa hal penting, karena:

a. UDP adalah "*datagram-oriented*", TCP adalah "*session-oriented*". *Datagram* adalah paket informasi *self-contained*; UDP berhubungan dengan *datagram* atau paket individu yang dikirim dari *client* ke *server*, atau sebaliknya.
b. UDP adalah *connection-less*, *client* tidak membangun koneksi ke *server* sebelum mengirim data, *client* hanya mengirim data secara langsung.
c. UDP "tidak andal" dalam pengertian jaringan formal; paket yang dikirim bisa hilang dan paket dapat mengalami kerusakan. Kata-istilah yang tidak begitu merendahkan daripada "tidak andal" adalah "*best-effort*". UDP melakukan yang terbaik yang dapat dilakukannya untuk mengirimkan paket tersebut tetapi pengirimannya tidak terjamin.
d. Karena UDP adalah *datagram-oriented* dan pada level protokol setiap paket berdiri sendiri, maka UDP tidak memiliki konsep paket sesuai urutan, yang selanjutnya berarti tidak memerlukan nomor urut pada paket tersebut.
e. Sejak pertama kali dikembangkan, TCP telah diperlengkapi dengan mekanisme yang sangat canggih untuk mengendalikan kecepatan aliran dalam koneksinya, untuk menghindari kemacetan dan kehilangan paket yang berlebihan. Karena UDP hanya mengirim paket tunggal, yang berdiri sendiri, maka UDP tidak memerlukan mekanisme kontrol yang rumit. Hal itu membuat UDP lebih mudah dan lebih kecil (dalam hal baris data dan memori) untuk diimplementasikan, tetapi juga membuatnya tidak cocok untuk sejumlah besar data.

Walaupun dengan segala kekurangannya yang nyata, ada beberapa aplikasi yang menggunakan UDP sebagai protokolnya. Padahal ada TCP yang andal dan tampaknya mampu melakukan semua tuntutan aplikasi tersebut. Aplikasi yang menggunakan UDP antara lain: *Domain Name System* (DNS), *Dynamic Host Configuration Protocol* (DHCP), *Microsoft Windows Networking*, *Trivial File Transfer Protocol* (TFTP), *System Logger* (Syslog), *Network File System* (NFS), dan lain-lain. Hal ini disebabkan karena UDP lebih murah dari TCP dalam hampir setiap hal yaitu:

a. UDP adalah protokol yang kecil dan mudah diimplementasikan, berbeda dengan TCP, yang sekarang sangat besar. UDP demikian kecil karena hanya menambahkan sangat sedikit, selain *port*, ke protokol IP yang mendasarinya.
b. Karena UDP kecil, maka dapat masuk dalam peralatan *special-purposes* dengan memori terbatas, atau dalam *flash* yang mahal atau *programmable read-only memory* (PROM) yang digunakan untuk *boot up* peralatan dalam jaringan. (Sekarang hal ini tidak begitu penting dibanding beberapa tahun yang lalu pada saat memori masih sangat mahal).
c. Karena UDP tidak sekompleks TCP, karena itu tidak banyak menggunakan CPU (*Central Processing Unit*).
d. Karena UDP *connection-less*, maka *client* dapat mengirim informasi ke server dengan sangat sedikit *overhead*. Sebaliknya membangun koneksi TCP menghabiskan waktu dan *resource* jaringan. UDP ideal jika *client* mengirimkan sejumlah kecil informasi dalam frekuensi yang jarang ke satu *server* atau lebih.
e. *Datagram* UDP dapat dikirim ke alamat *broadcast* untuk *request service* yang lokasi *server*-nya tidak diketahui *client* tersebut. TCP hanya dapat melakukan koneksi antar alamat individu yang spesifik (Mansfield, 2004).

### 2.3. Windows Sockets (WinSock)

Perkembangan *Windows Sockets* (WinSock), pada versi pertamanya, yaitu WinSock versi 1.1 telah menjadi standar sejak peluncurannya pada bulan Januari 1993, dan telah

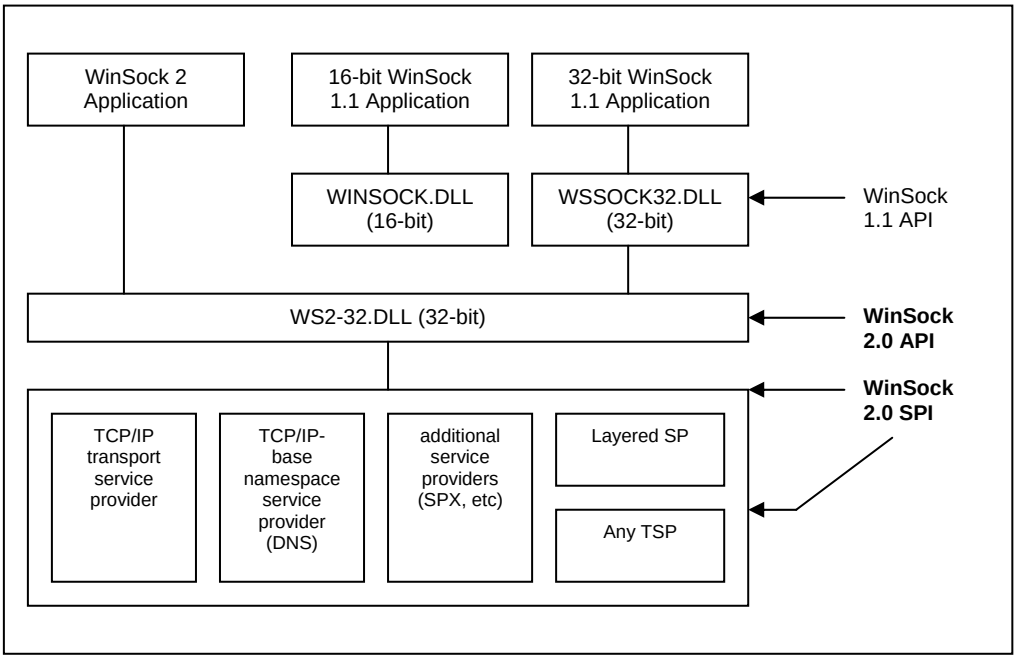
memenuhi tujuan asli *programmer*-nya yaitu untuk menyediakan suatu *Application Programming Interface* (API) pada sistem operasi Windows, yang bersifat fleksibel dan tangguh untuk menciptakan aplikasi TCP/IP yang universal.

Windows Socket versi 2.0 (WinSock 2) menyusun API untuk sejumlah deretan protokol yang lain seperti ATM, IPX/SPX, dan DECnet dan mengijinkan mereka untuk beroperasi secara bersama-sama. Hal yang paling penting yaitu WinSock 2 mampu mengerjakan semua tuntutan itu dan masih mempertahankan kompatibilitas dengan versi 1.1. Aplikasi WinSock dapat terus berjalan tanpa modifikasi (hanya saja pengecualian untuk aplikasi Winsock 1.1 yang masih menggunakan *blocking hooks*, tetapi pada WinSock 2 hal itu tidak ditulis ulang lagi (Quinn, 1998).

### 2.4. Arsitektur Windows Sockets

Arsitektur yang sederhana dari WinSock 1.1. yaitu berupa file WINSOCK.DLL tunggal (atau WSOCK32.DLL) yang menyediakan WinSock API. Sejak versi 1.1-nya, Winsock hanya mendukung satu *suite TCP/IP* protokol karena kebanyakan komputer Windows hanya mempunyai alat penghubung jaringan yang tunggal. Arsitektur ini membatasi sistem untuk mengaktifkan hanya satu WinSock DLL pada waktu yang sama pada sistem tersebut.

WinSock 2 mempunyai arsitektur yang semuanya serba baru dan menyediakan fleksibilitas yang jauh lebih baik. Arsitektur WinSock 2 yang baru, memberikan dukungan untuk *multiple stack protocol*, *interface*, dan *service provider*. WinSock 2 mengadopsi model *Windows Open Systems Architecture* (WOSA), yang memisahkan API dari protokol *service provider*. Di model ini WINSOCK DLL menyediakan API yang baku, dan masing-masing *vendor* harus menginstal sendiri *service provider layer*-nya pada bagian bawah, dan *API layer*-nya merupakan standarisasi dari *Service Provider Interface* (SPI). Arsitektur WinSock 2.0 seperti pada Gambar 3 (Quinn, 1998).




Gambar 3. Arsitektur WinSock 2

## 3. ANALISIS DAN PERANCANGAN

### 3.1. Analisis Kebutuhan Pengguna

Aplikasi komunikasi berbasis text yang akan dibangun ini, memberikan layanan dan kemudahan-kemudahan bagi pengguna, diantaranya:

a. Komunikasi bisa dilakukan oleh banyak pengguna pada waktu yang bersamaan.
b. Tidak membutuhkan layanan *internet* dalam menjalankan aplikasi.

c. Tidak membutuhkan spesifikasi komputer yang tinggi dalam menjalankan aplikasi.
d. Aplikasi komunikasi yang murah dari segi ekonomi.
e. Tidak membutuhkan aplikasi *server* untuk menangani lalu-lintas data saat proses komunikasi terjadi.
f. Mampu berjalan di semua sistem operasi Windows.
g. Mudah dalam instalasi.
h. Seting dan konfigurasi yang tidak rumit.
i. Adanya pemeliharaan serta monitoring jaringan komputer.
j. Kemudahan dalam berkomunikasi.
k. Memerlukan hanya sedikit sumber daya jaringan kommputer dalam menjalankan aplikasi.

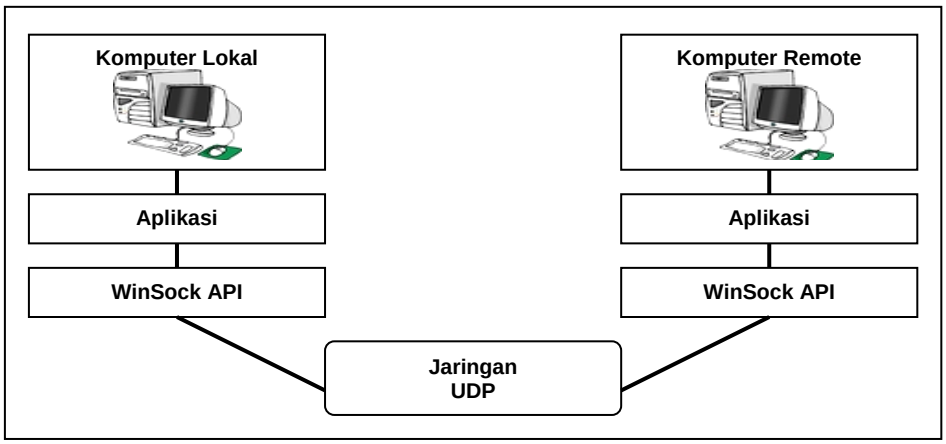
### 3.2. Analisis Sistem

Aplikasi komunikasi berbasis text yang akan dibangun ini memiliki spesifikasi sistem sebagai berikut:

a. Layanan komunikasi yang diberikan bersifat *multicast*.
b. Tampilan program dibuat semenarik mungkin dan sangat *user friendly*.
c. Proses *connecting* dan *disconnecting* yang sangat cepat.
1. Membutuhkan sumber daya komputer maupun jaringan komputer yang sangat minimum.
d. Terdapat *Pinger Tool* yang merupakan visualisasi dari *ECHO REQUEST* dan *ECHO REPLY* dari *Internet Control Message Protocol* (ICMP) untuk meningkatkan *traffic test* serta pemeliharaan jaringan komputer.
e. Terdapat *tools Invite Friends* yang merupakan visualisasi dari *Messenger Service* Windows untuk kemudahan komunikasi.
f. *Text input* maupun *text output* berupa *Rich Text Format*.
g. Dapat menyimpan *output text* ke dalam file dokumen dengan format *Rich Text Format*.
h. Terdapat *Smiley* yang telah disempurnakan ke dalam model grafis.
i. Dapat menjalankan aplikasi lebih dari satu aplikasi pada komputer yang sama.
j. Terdapat *help document* yang dapat digunakan sebagai *manual book* dalam menggunakan aplikasi ini.
k. Aplikasi yang akan dibangun didasari oleh teknologi *Windows Sockets* (WinSock) sebagai *Application Programming Interface* (API) dan *User Datagram Protocol* (UDP).

### 3.3. Rancangan Proses

Arsitektur aplikasi yang akan dibangun dapat dilihat pada Gambar 4.

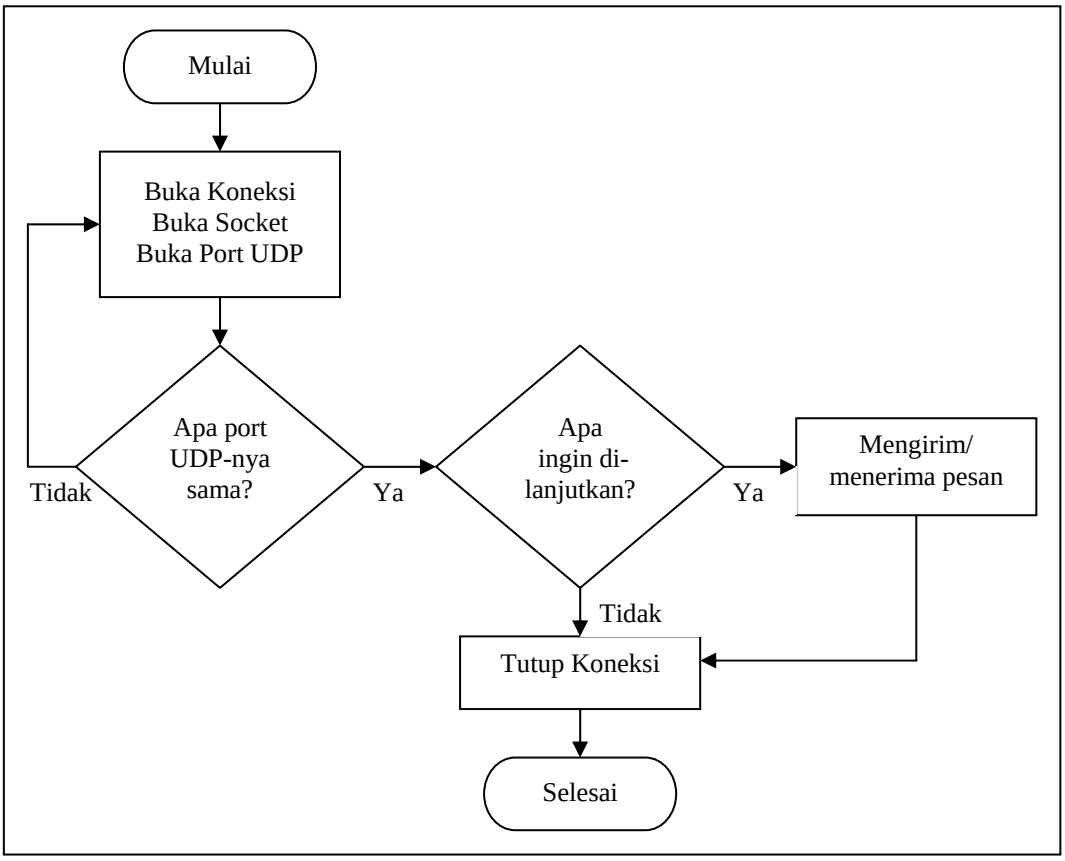



Gambar 4. Arsitektur Aplikasi

#### 3.3.1. *Flowchart* untuk *Main Program*

Pada *flowchart* ini digambarkan aliran arus logika dan langkah-langkah proses yang terjadi pada *Main Program*. Proses yang digambarkan terjadi antara aplikasi pada komputer lokal dan komputer *remote* yang terpisah tetapi dengan prosedur aliran arus logika dan langkah-langkah proses yang sama. Proses komunikasi (mengirim/menerima pesan) tidak akan

terjadi apabila komputer lokal maupun komputer *remote* sama-sama menjalankan aplikasi tetapi menggunakan nomor *port* UDP yang berbeda-beda. Jadi untuk bisa melakukan komunikasi (mengirim/menerima pesan), komputer lokal maupun komputer *remote* harus sama-sama menjalankan aplikasi dan menggunakan nomor *port* UDP yang sama. *Flowchart* untuk *Main Program* untuk komputer lokal maupun komputer *remote* dapat dilihat pada Gambar 5.



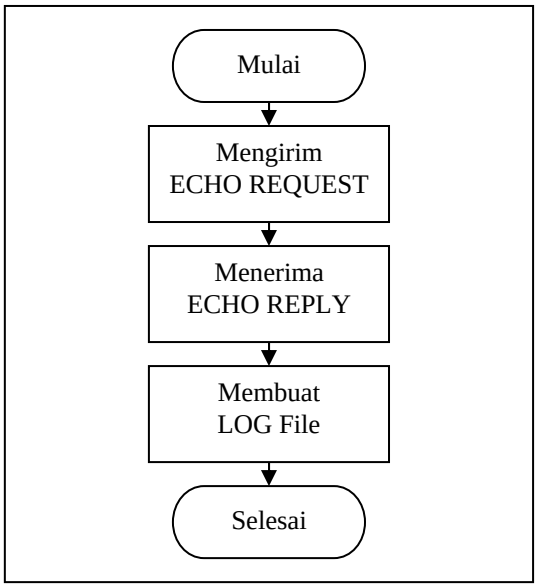

Gambar 5. *Flowchart Main Program* pada Komputer Lokal dan Komputer *Remote*

### 3.3.2 *Flowchart* untuk *Pinger Tool*

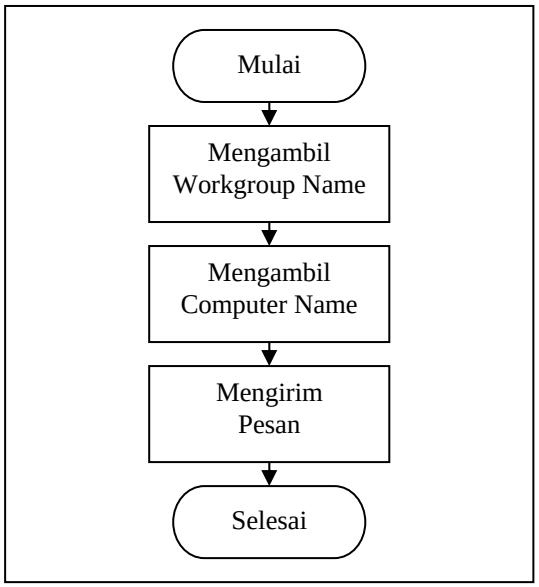
Pada *flowchart* ini digambarkan aliran arus logika dan langkah-langkah proses yang terjadi pada *Pinger Tool*. *Tool* ini merupakan visualisasi dari *ECHO REQUEST* dan *ECHO REPLY* dari *Internet Control Message Protocol* (ICMP). Sebenarnya *tool* ini merupakan *tool* bawaan dari sistem operasi Windows dan untuk menjalankan *tool* ini harus melalui *Command Prompt*-nya Windows, jadi proses detailnya di-*handle* langsung oleh sistem operasi Windows. Proses tambahannya hanya berupa: Membuat *LOG File* yang fungsinya untuk menyimpan hasil dari jawaban *ECHO REPLY* setelah *ECHO REQUEST* dikirimkan. *Flowchart* untuk *Pinger Tool* dapat dilihat pada Gambar 6.

### 3.3.3 *Flowchart* untuk *Invite Friends*

Pada *flowchart* ini digambarkan aliran arus logika dan langkah-langkah proses yang terjadi pada *Invite Friends*. *Tool* ini merupakan visualisasi dari *Messenger Service* Windows NT, 2000 dan XP. Sebenarnya *tool* ini merupakan *tool* bawaan dari sistem operasi Windows dan untuk menjalankan *tool* ini harus melalui *Command Prompt*-nya Windows, jadi proses detailnya di-*handle* langsung oleh sistem operasi Windows. *Flowchart* untuk *Invite Friends* dapat dilihat pada Gambar 7.




Gambar 6. *Flowchart Pinger Tool*




Gambar 7. *Flowchart Invite Friends*

## 4. IMPLEMENTASI

### 4.1 Perangkat Lunak yang Digunakan

Perangkat lunak yang digunakan dalam tahap *Development* dan *Deployment* yaitu:

a. Microsoft Visual Basic 6.0 merupakan perangkat lunak utama yang digunakan untuk meng-*compile* kode program dan pembuatan *user interface* aplikasi.
b. Inno Setup Compiler versi 4.20 merupakan perangkat lunak yang digunakan untuk pembuatan media *installer* aplikasi.
c. IconCool Editor versi 2.8 *build* 20304 merupakan perangkat lunak yang digunakan untuk pembuatan *icon*.
d. Macromedia Flash 5 merupakan perangkat lunak yang digunakan untuk pembuatan *Manual Book*.
e. Xara Webstyle versi 3.1 merupakan perangkat lunak yang digunakan untuk pembuatan teks 3D.
f. Microsoft Windows XP Professional Edition SP1 sebagai sistem operasi pada saat pembangunan aplikasi.

### 4.2 Perangkat Keras yang Digunakan

Perangkat keras yang digunakan pada saat pembangunan aplikasi yaitu sebuah komputer dengan spesifikasi:

a. *Processor* Intel Celeron 850 MHz
b. *Memory* SDRAM 320 MB PC133
c. *Motherboard* Microstar MS-6309 Lite!
d. *Harddisk* 20 GB 7200 rpm
e. VGA GeForce 2MX 200 32 MB
f. NIC D-Link DFE-530TX PCI 10/100 Mbps

Pada tahap pengujiannya menggunakan jaringan komputer lokal dengan 5 komputer yang terhubung pada *hub* dengan kecepatan 10 Mbps.

### 4.3 *Components* dan *Library* yang Digunakan

Dalam membangun aplikasi ini, menggunakan *components* serta *library* yang disediakan oleh Microsoft Visual Basic 6.0.

*Components* yang digunakan yaitu:

a. *Microsoft Common Dialog Control 6.0 (SP3)*

b. *Microsoft Rich Textbox Control 6.0 (SP4)*
c. *Microsoft Windows Common Control 6.0 (SP4)*
d. *Microsoft Winsock Control 6.0*

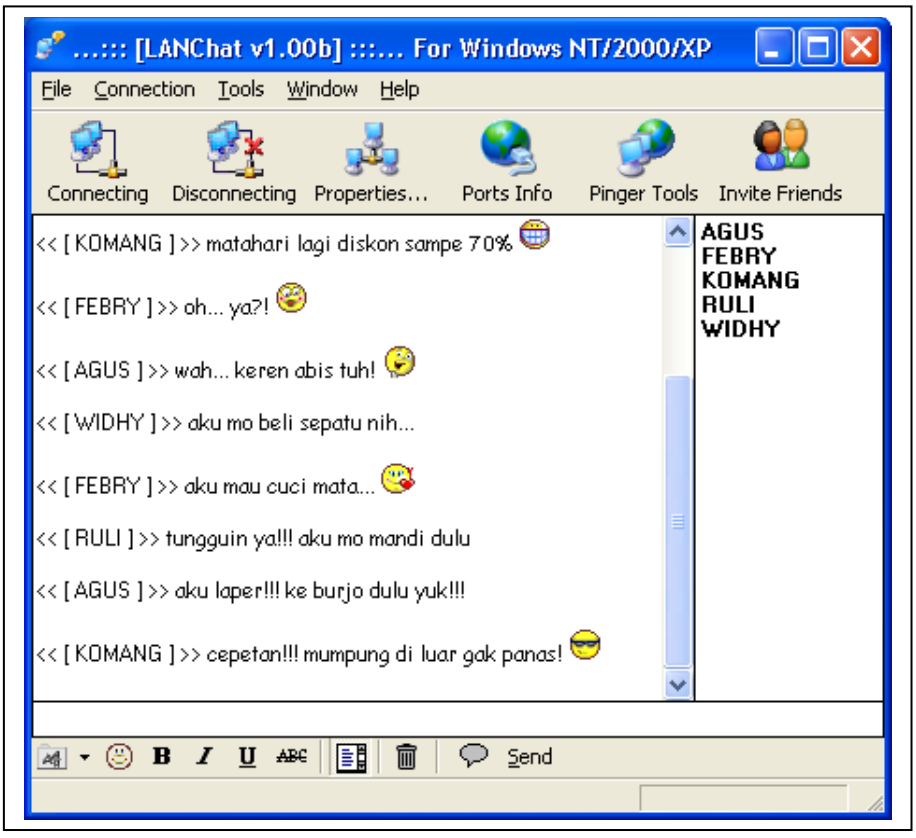
*Library* yang digunakan yaitu:

a. *Visual Basic For Applications*
b. *Visual Basic Runtime Objects and Procedures*
c. *Visual Basic Objects and Procedures*
d. *OLE Automation*
e. *Active DS Type Library*
f. *Microsoft Connection Designer Instance 1.0*
g. *Microsoft Remote Data Object 2.0*
h. *Microsoft Shell Controls and Automation*

### 4.4. Implementasi Program

#### 4.4.1 Main Program

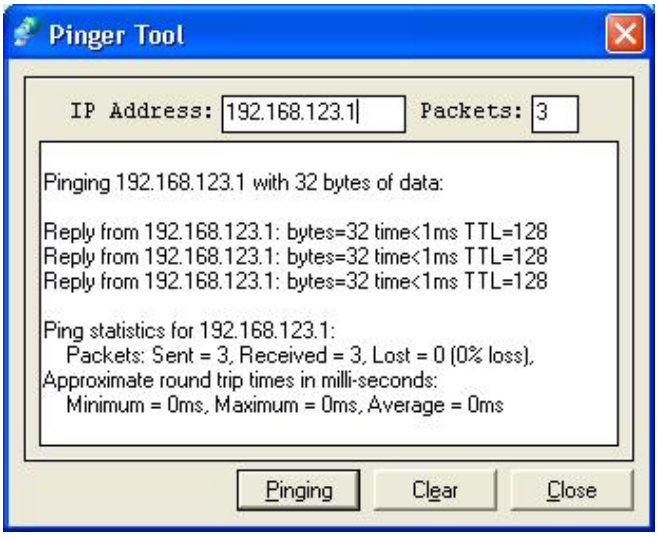
*Main Program* merupakan inti masalah pada penelitian ini. Penggunaan teknologi *Windows Sockets* (WinSock) dan *User Datagram Protocol* (UDP) terdapat pada *Main Program*. Proses komunikasi juga terdapat pada *Main Program*. Tampilan *Main Program* dapat dilihat pada Gambar 8.




Gambar 8. *Main Program*

#### 4.4.2 *Pinger Tool*

*Pinger Tool* merupakan tampilan aplikasi yang dapat digunakan untuk memeriksa apakah sebuah komputer *host* dalam keadaan hidup atau tidak, juga dapat digunakan untuk meningkatkan *traffic test* antara 2 komputer *host*. Tampilannya dapat dilihat pada Gambar 9.



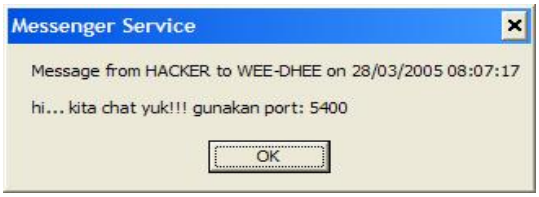

Gambar 9. *Pinger Tool*

### 4.4.3 *Invite Friends*

*Invite Friends* merupakan tampilan aplikasi yang berfungsi untuk mengirim pesan singkat kepada pengguna lain dalam suatu jaringan komputer. *Invite Friends* merupakan visualisasi dari *Messenger Service* Windows yang tersedia pada Windows NT, 2000 dan XP. Manfaat utama dari *Invite Friends* ini yaitu dapat digunakan untuk mengundang pengguna lain agar ikut bergabung dalam *chat room* yang sedang aktif. Tampilan *Invite Friends* dapat dilihat pada Gambar 10 dan 11.


Gambar 10. *Invite Friends*

Tampilan aplikasi untuk komputer penerima pesan dapat dilihat pada Gambar 11.




Gambar 11. *Messenger Service*

## 5. PENUTUP

### 5.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan maka dapat disimpulkan:

1. Telah berhasil dibangun suatu aplikasi komunikasi berbasis *text* menggunakan teknologi *Windows Sockets* (WinSock) dan *User Datagram Protocol* (UDP).
2. *User Datagram Protokol* (UDP) merupakan protokol yang "tidak andal" dalam hal pengiriman paket data yang berkapasitas besar.
3. *User Datagram Protokol* (UDP) sangat cocok diimplementasikan pada aplikasi yang transfer datanya kecil dan aplikasi *client-server* dimana *client* mengontak *server* dengan sangat jarang.

### 5.2 Saran

Beberapa saran yang dapat digunakan sehingga aplikasi ini diharapkan nantinya dapat disempurnakan, antara lain:

1. Aplikasi ini dapat dikembangkan lagi agar bisa digunakan melalui *internet* dan berjalan di semua sistem operasi komputer (diimplementasikan menggunakan java).
2. Bisa juga ditambahkan fasilitas *transfer file*, tetapi harus diimplementasikan menggunakan protokol TCP (*Transmission Control Protocol*).
3. Sebaiknya menggunakan protokol TCP (*Transmission Control Protocol*) untuk membangun aplikasi komunikasi data, agar hasilnya lebih sempurna.

## DAFTAR PUSTAKA

[ 1] Microsoft, "***Microsoft Developer Network (MSDN) Library***" 2004, Microsoft Corporation.

[ 2] Mansfield, N., "***Pratical TCP/IP Mendesain, Menggunakan, dan Troubleshooting Jaringan TCP/IP di Linux dan Windows***" Penerbit Andi, Yogyakarta, 2004.

[ 3] Fahrial, J., "***Teknik Konfigurasi LAN di Windows***" 2003. <http://www.ilmukomputer.com>, (accessed 01 Feb 2005).

[ 4] Postel, J., "***RFC 768 User Datagram Protocol***" 1980. <ftp://ftp.rfc-editor.org/in-notes/rfc768.txt>, (Aug 1980, accessed 01 Feb 2005).

[ 5] Prihanto, H., "***Membangun Jaringan Komputer: Mengenal Hardware dan Topologi Jaringan"*** 2003. <http://www.ilmukomputer.com>, (Jun 2003, accessed 01 Feb 2005).

[ 6] Quinn, B., "***Windows Sockets Network Programming*** **Addison-Wesley Publishing Company**" Massachusetts, 1995.

[ 7] Quinn, B., "***WinSock 2 Information***" 1998. <http://www.sockets.com/winsock2.htm>, (accessed 01 Feb 2005).