Model Pengujian Komunikasi Socket dengan Protokol TCP/IP

Aji Supriyanto
Fakultas Teknologi Informasi, Universitas Stikubank Semarang
e-mail : ajisup@gmail.com

ABSTRAK: Pemrograman socket merupakan pemrograman yang digunakan untuk melakukan komunikasi proses (*process-to-process*) dalam sebuah jaringan. Selain komunikasi proses, dalam sebuah jaringan komputer juga melakukan komunikasi host (*host-to-host*). Dalam pengujian pemrograman socket dapat dilakukan dengan menggunakan jenis *socket networking sock_stream* pada client ke server dengan menggunakan alamat proses (IP dan Port) pada komputer yang sama dengan direktori yang sama dan yang berbeda serta dengan komputer lain yang berbeda. Untuk pengujian, akan dibuat server dan client yang akan melakukan komunikasi proses. Pada server dan client, socket yang digunakan adalah AF_INET. Pada komunikasi proses di server, mula-mula dilakukan proses pembuatan socket dengan fungsi call socket(), kemudian server mengikat socket yang dibuat tadi dengan jaringan berdasarkan nomor local address dan port dengan fungsi call bind().

Kata Kunci : socket, protocol, client-server

PENDAHULUAN

Komunikasi antar host dilakukan dengan menggunakan protocol IP yang bekerja pada lapis jaringan (*network layer*). Komunikasi pada host yang dilakukan protocol IP belum lengkap atau sempurna jika belum disertai proses yang benar. Oleh karena itu pesan (*message*) yang berpindah antar host ke host lain akan diproses lebih lanjut pada lapis transport (*transport* layer) diperlukan agar bisa berkomunikasi *client-server*.

Proses yang terjadi pada local host disebut *client*, yang membutuhkan layanan sebuah proses yang lain yang disebut *server*. Proses yang dilakukan oleh client dan server haruslah jenis dan proses yang bernama sama. Untuk mendefinisikan nama sebuah proses diperlukan sebuah identifier khusus yang disebut *port***.** Dalam protocol TCP/IP nomor port yang dapat digunakan adalah 0 hingga 65.535 (16 bit).

Pada protocol UDP (*user datagram protocol*) yang berada pada *transport layer*, dalam melakukan komunikasi membutuhkan dua identifier sekaligus yaitu alamat IP yang berada pada *network layer* dan alamat atau nomor port untuk melakukan pengalamatan socket (*socket address*).

Bagaimana pengujian pemrograman socket dilakukan dengan menggunakan jenis socket networking *sock_stream* pada client ke server dengan menggunakan alamat proses (IP dan Port) pada komputer yang sama dengan direktori yang sama dan yang berbeda serta dengan komputer lain yang berbeda (dua komputer atau lebih). Tujuan dari bahasan ini adalah :

- Menguji koneksi client dan server dengan komputer yang berbeda (dua komputer atau lebih)
- Menguji koneksi client dan server dengan komputer yang sama dengan IP :127.0.0.1 (*loopback*)
- Menguji koneksi client dan server dengan komputer yang sama tetapi dengan direktori yang berbeda

KONSEP SOCKET DAN IP ADDRESS

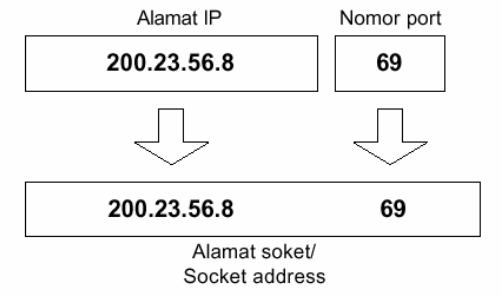
Socket adalah suatu mekanisme yang menyediakan komunikasi untuk (dua) proses yang berjalan (tidak pasti) pada host-host yang berbeda (P. Bergamo). Definisi lain

menyebutkan, Socket adalah mekanisme komunikasi yang memungkinkan terjadinya pertukaran data antar program atau proses baik dalam satu mesin maupun antar mesin. Sehingga pemrograman socket dapat didefinisikan sebagai pemrograman yang menyediakan komunikasi untuk komunikasi antar proses dalam jaringan (client-server).

Komunikasi socket terutama diciptakan untuk tujuan menjembatani komunikasi antara dua buah program yang dijalankan pada mesin yang berbeda, tentu saja komunikasi mesin yang sama juga dapat dilakukan. Kelebihan lain dari komunikasi socket adalah mampu menangani banyak klien sekaligus (*multiple clients*).

Pengalamatan Socket

Dalam melakukan komunikasi proses antar host diperlukan dua protocol utama yang penting yaitu IP (*Internet Protocol*) yang memberikan nomor alamat IP yang berada pada lapis jaringan (*network layer*), dan protocol UDP (*User Datagram Protocol*) yang memberikan nomor alamat port yang berada pada lapis transport (*transport layer*). Sehingga alamat sebuah socket dapat digambarkan sebagai berikut :



Gambar 1. Pengalamatan socket

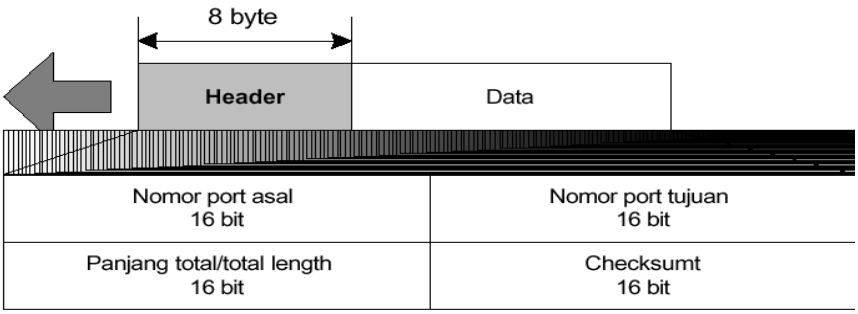
IP Address

Adanya IP Address merupakan konsekuensi dari penerapan Internet Protocol untuk mengintegrasikan jaringan komputer Internet di dunia. Seluruh host (komputer) yang terhubung ke Internet dan ingin berkomunikasi memakai TCP/IP harus memiliki IP Address sebagai alat pengenal host pada network. Secara logika, Internet merupakan suatu network besar yang terdiri dari berbagai sub network yang terintegrasi. Oleh karena itu, suatu IP Address harus bersifat unik untuk seluruh dunia. Tidak boleh ada satu IP Address yang sama dipakai oleh dua host yang berbeda.

IP Address terdiri dari bilangan biner sepanjang 32 bit yang dibagi atas 4 segmen. Tiap segmen terdiri atas 8 bit yang berarti memiliki nilai desimal dari 0 - 255. Range address yang bisa digunakan adalah dari 00000000.00000000.00000000.00000000 sampai dengan 11111111.11111111.11111111. 11111111. Untuk memudahkan pembacaan dan penulisan, IP Address biasanya direpresentasikan dalam bilangan desimal. Jadi, range address di atas dapat diubah menjadi address 0.0.0.0 sampai address 255.255.255.255. Nilai desimal dari IP Address inilah yang dikenal dalam pemakaian sehari-hari. Beberapa contoh IP Address adalah : 44.132.1.20 dan 167.205.9.35

IP Address dapat dipisahkan menjadi 2 bagian, yakni bagian network (bit-bit network/network bit) dan bagian host (bit-bit host/host bit). Bit network berperan dalam identifikasi suatu network dari network yang lain, sedangkan bit host berperan dalam identifikasi host dalam suatu network. Jadi, seluruh host yang tersambung dalam jaringan yang sama memiliki bit network yang sama. Sebagian dari bit-bit bagian awal dari IP Address merupakan network bit/network number, sedangkan sisanya untuk host. Garis pemisah antara bagian network dan host tidak tetap, bergantung kepada kelas network. Ada 3 kelas address yang utama dalam TCP/IP, yakni kelas A, kelas B dan kelas C. Perangkat lunak Internet Protocol menentukan pembagian jenis kelas ini dengan menguji beberapa bit pertama dari IP Address.

Selain ke tiga kelas di atas, ada 2 kelas lagi yang ditujukan untuk pemakaian khusus, yakni kelas D dan kelas E. Jika 4 bit pertama adalah 1110, IP Address merupakan kelas D yang digunakan untuk multicast address, yakni sejumlah komputer yang memakai bersama suatu aplikasi (bedakan dengan pengertian network address yang mengacu kepada sejumlah komputer yang memakai bersama suatu network). Kelas terakhir adalah kelas E (4 bit pertama adalah 1111 atau sisa dari seluruh



Gambar 2. Format UDP

kelas). Pemakaiannya dicadangkan untuk kegiatan eksperimental.

Dari alamat-alamat IP tersebut di atas terdapat alamat tertentu yang memiliki fungsi khusus yaitu IP dengan nomor 127.0.0.1 untuk kegiatan *loopback*, yaitu melakukan koneksi pada host (komputer) dirinya sendiri. Selain itu ada nomor IP 255.255.255.255 yang berfungsi untuk melakukan *bradcast*.

UDP Protocol

UDP merupakan protocol *process-to-process* yang menambahkan hanya alamat *port*, *chec-sum error control*, dan panjang informasi dari lapis yang ada diatasnya. Alamat *port* yang dapat digunakan adalah 0 hingga 65.535 (16 bit), yang dapat digolongkan menjadi beberapa sebagai berikut ;

1. *Well-known ports* : yang berisi nomor port antara 0 hingga 1.023
2. *Registered Port* : yang berisi nomor port antara 1.024 hingga 49.151
3. *Dynamic Port* : yang berisi nomor port antara 49.152 hingga 65.535

Suatu paket UDP disebut sebagai user datagram, yang memiliki *header* yang tetap yaitu sebesar 8 byte, yang dapat diperlihatkan dalam gambar 2.

Fungsi UDP adalah :

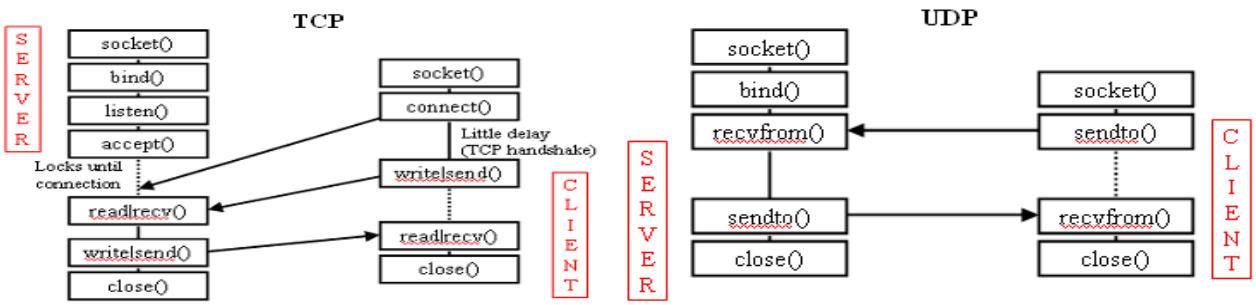
1. UDP sangat cocok untuk proses yang memerlukan *request-respon communication* dan sedikit sekali yang memperhatikan masalah *flow control* dan *error control*
2. UDP yang melakukan proses dengan mekanisme internal *flow control* dan *error control* hanya untuk proses TFTP (*Trivial File Transfer Protocol*).
3. UDP cocok untuk *multicasting* dan *bradcasting* pada lapisan transport.
4. UDP digunakan untuk manajemen proses seperti aplikasi SNMP.
5. UDP digunakan sebagai peng-*update* protocol *routing* seperti pada RIP (*Routing Information Protocol*).

MODEL DAN BAHASAN

Model Sistem

Pengujian kali ini akan menggunakan jenis socket networking *Sock_Stream* atau *Socket Stream*. *Socket Stream* adalah socket komunikasi *full-duplex* berbasis aliran (*stream*) data. Pada model komunikasi *Socket Stream*, koneksi dua aplikasi harus dalam kondisi tersambung dengan benar untuk dapat bertukar data. Koneksi model seperti ini akan menjamin data dapat dipertukarkan dengan baik, namun memiliki kelemahan dalam hal penggunaan jalur data yang relatif besar dan tidak boleh terputus.

Untuk pengujian, akan dibuat server dan client yang akan melakukan komunikasi proses. Pada server dan client, socket yang digunakan adalah AF_INET yang akan mendefinisikan empat buah komponen, yaitu :



Gambar 3. Sesi-sesi pada TCP (kiri) dan UDP (kanan) untuk komunikasi proses

1. Identifikasi Nomor atau Alamat Host Remote
2. Nomor Port Host Remote
3. Identifikasi Nomor atau Alamat Host Lokal
4. Nomor Port Host Lokal

Pada server, yang penting didefinisikan diawal adalah poin 3 dan poin 4. Poin 1 dan poin 2 didapat saat ada client yang masuk menghubungi server.

Terdapat 5 langkah dasar untuk server pada pemrograman socket yang melalui protocol TCP, seperti yang terdapat pada gambar 3, yaitu

1. Membuat socket dengan perintah socket()
2. Mengikatkan socket kepada sebuah alamat network dengan perintah bind()
3. Menyiapkan socket untuk menerima koneksi yang masuk dengan perintah listen()
4. Menerima koneksi yang masuk ke server dengan perintah accept()
5. Melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah write() atau send() dan read() atau recv().

Untuk membuat proses pada client, langkah-langkahnya adalah sabagai berikut :

1. Membuat socket dengan perintah socket()
2. Menghubungi server dengan connect()
3. Melakukan komunikasi (mengirim data dan menerima data), dengan menggunakan perintah send() atau write() dan read() atau recv()

Sedangkan untuk proses server dan client yang melalui protocol UDP, hanya terdapat 2 fungsi baru yaitu sendto() dan recvfrom yang memiliki paramater yang sama dengan fungsi send() dan recv(). Perbedaannya adalah baik fungsi sendto() dan recvfrom memiliki 2 paramater lagi yaitu pointer sockaddr ke tujuan dan ukuran pointer sockaddr tersebut.

Pada komunikasi proses di server, mula-mula dilakukan proses pembuatan socket dengan fungsi call socket(), kemudian server mengikat socket yang dibuat tadi dengan jaringan berdasarkan nomor local address dan port dengan fungsi call bind(). Setelah local address dan port ditentukan, langkah selanjutnya, server akan menunggu datangnya request dari client. Pada langkah ini server menggunakan fungsi call listen() untuk menangani antrian dari request yang datang. Setelah itu proses akan diblok sampai ada koneksi antara client ke server dengan menggunakan fungsi call accept() dan kemudian server akan menerima data dari socket dengan menggunakan fungsi call recv(). Bila proses telah selesai dijalankan maka koneksi ditutup dengan proses close(). Untuk proses di client, sistemnya mula-mula juga melakukan proses pembuatan socket. Setelah socket tercipta maka aplikasi berusaha melakukan proses connect pada server dengan port dan address dari server. Jika client sudah terhubung dengan server, maka dapat dilakukan proses send data yang diinginkan. Proses menutup koneksi socket dilakukan setelah semua proses selesai dikerjakan.

HASIL PENGUJIAN

Kompilasi Program Client

Pada pengkompilasian proses client menggunakan file stream_client.c untuk programnya. Port pada client harus sama dengan port pada server agar keduanya dapat saling berkomunikasi. Misalkan saja port didefinisikan dengan nilai :

```
...
#define PORT 25563
...
```

Kompilasi Program Server

Setelah port didefinisikan dan file di-save, langkah selanjutnya adalah proses kompilasi dengan cara:

```
[bayu@mkom22 bayu]$ gcc -o client
stream_client.c
```

Kemudian dihasilkan file kompilasi untuk menjalankan proses client. Disini terdapat 3 argumen yang harus diisikan yaitu:

- argumen file kompilasi
- argumen alamat IP atau bisa dengan hostname
- argumen nama file

Untuk menguji penggunaan argumen pada hasil kompilasi pada client, akan dicoba dengan menjalankan program sebagai berikut :
Proses mengalami error karena argumen ke 2 dan ke 3 tidak diisikan.

```
[bayu@mkom22 bayu]$ ./client
Usage Error: ./client node_name
file_name
```

Proses mengalami error karena argumen ke 3 tidak diisikan.

```
[bayu@mkom22 bayu]$ ./client
192.168.10.29
Usage Error: ./client node_name
file_name
```

Proses client berjalan sebagaimana mestinya dengan 3 argumen

```
[bayu@mkom22 bayu]$ ./client
192.168.10.29 Keamanan_Jaringan
Connection from address
192.168.10.24 port 25563
server read Keamanan_Jaringan
file do not exist, get it from
client
File sent
copied 0 bytes
```

Untuk proses kompilasi pada program server, akan dipakai salah satu file stream_server dari Bergamo. Untuk pengujian ini, akan digunakan file stream_server_seq.c. Kompilasi file tersebut dengan cara:

```
[aji@mkom22 aji]$ gcc -o server
stream_server_seq.c
```

Jika sudah tidak terdapat error pada program server tersebut, lalu akan menjalankan program, agar server berjalan sebagai background maka dalam perintahnya ditambahkan huruf ‘ & ’.

```
[aji@mkom22 aji]$ ./server &
[1] 940
```

Pada kompilasi server dibawah ini menunjukkan bahwa definisi port sudah digunakan oleh proses lain, dan sedang berjalan. Sehingga server tidak dapat berjalan karena alamat yang akan digunakan sudah terpakai.

```
[aji@mkom22 aji]$ gcc -o server
stream_server_seq.c
[aji@mkom22 aji]$ ./server &
[1] 1939
[aji@mkom22 aji]$ bind: Address
already in use
[1]+  Exit 1 ./server
[aji@mkom22 aji]$
```

Komunikasi Client dengan Server

Setelah server dan client dikompilasi dengan benar, langkah selanjutnya adalah menguji komunikasi antara keduanya dengan cara:

Menguji koneksi server dan client dengan komputer yang berbeda (dua komputer atau lebih)
Untuk pengujian pada 2 komputer

Di komputer client

```
M.Kom Server
_____________

login: aji
```

```
Password:
Last login: Wed Apr 27 09:15:38 from 192.168.10.12
[aji@mkom22 aji]$ ls
Keamanan
[aji@mkom22 aji]$ cd Keamanan
[aji@mkom22 Keamanan]$ cd Bergamo
[aji@mkom22 Bergamo]$ cd SockCode
[aji@mkom22 SockCode]$ cd SOCK_STREAM/
[aji@mkom22 SOCK_STREAM]$ gcc -o client stream_client.c
[aji@mkom22 SOCK_STREAM]$ ./client 192.168.10.26 KirimDariAji
open file: No such file or directory
[aji@mkom22 SOCK_STREAM]$
```

Keterangan di komputer server

```
[aji@mkom22 server]$ Connection from address 192.168.10.26 port 32778
server read KirimDariAji
file do not exist, get it from client
copied 0 bytes

[aji@mkom22 server]$ ls -l
total 20
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLain
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLainUlang
-r--------  1 aji    jarkom-a     0  May  3 10:05 KirimDariAji
-rwxr-xr-x  1 aji    jarkom-a 13902  May  3 09:54 stream_server_seq
-rw-r--r--  1 aji    jarkom-a  2308  Mar  2 08:32 stream_server_seq.c
[aji@mkom22 server]$
```

Untuk pengujian pada lebih dari 2 komputer
Client yang mengirim ke server dari komputer yang berbeda.

Pada komputer client kesatu

```
M.Kom Server
_____________

login: aji
Password:
Last login: Wed Apr 27 07:44:31 from 192.168.10.13
[aji@mkom22 aji]$ cd jarkomc
[aji@mkom22 jarkomc]$ gcc -o client stream_client.c
[aji@mkom22 jarkomc]$ ./client 192.168.10.26 MasukKompAji
open file: No such file or directory
[aji@mkom22 jarkomc]$
```

Cek file yang masuk ke server :

```
[aji@mkom22 server]$

[aji@mkom22 server]$ ls -l
total 20
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLain
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLainUlang
-r--------  1 aji    jarkom-a     0  May  3 10:05 KirimDariAji
-r--------  1 aji    jarkom-a     0  May  3 10:38 MasukKompAji
-rwxr-xr-x  1 aji    jarkom-a 13902  May  3 09:54 stream_server_seq
-rw-r--r--  1 aji    jarkom-a  2308  Mar  2 08:32 stream_server_seq.c
[aji@mkom22 server]$
```

Pada komputer client kedua

```
M.Kom Server
```

```
_____________

login: rita
Password:
Last login: Wed Apr 27 08:23:56 from 192.168.10.3
[rita@mkom22 rita]$ cd client
[rita@mkom22 client]$ gcc -o client stream_client.c
[rita@mkom22 client]$ ./client 192.168.10.26 MasukKompRita
open file: No such file or directory
[rita@mkom22 client]$
```

Cek file yang masuk ke server :

```
[aji@mkom22 server]$

[aji@mkom22 server]$ ls -l
total 20
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLain
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLainUlang
-r--------  1 aji    jarkom-a     0  May  3 10:05 KirimDariAji
-r--------  1 aji    jarkom-a     0  May  3 10:38 MasukKompAji
-r--------  1 aji    jarkom-a     0  May  3 10:40 MasukKompRita
-rwxr-xr-x  1 aji    jarkom-a 13902  May  3 09:54 stream_server_seq
-rw-r--r--  1 aji    jarkom-a  2308  Mar  2 08:32 stream_server_seq.c
[aji@mkom22 server]$
```

Pada komputer client ketiga

```
M.Kom Server
_____________

login: ilham
Password:
Last login: Wed Apr 27 08:50:35 from 192.168.10.11
[ilham@mkom22 ilham]$ cd Bergamo
[ilham@mkom22 Bergamo]$ cd SockCode
[ilham@mkom22 SockCode]$ cd SOCK_STREAM/
[ilham@mkom22 SOCK_STREAM]$ gcc -o client stream_client.c
[ilham@mkom22 SOCK_STREAM]$ ./client 192.168.10.26 MasukKompIlham
open file: No such file or directory
[ilham@mkom22 SOCK_STREAM]$
```

Cek file yang masuk ke server :

```
[aji@mkom22 server]$

[aji@mkom22 server]$ ls -l
total 20
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLain
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLainUlang
-r--------  1 aji    jarkom-a     0  May  3 10:05 KirimDariAji
-r--------  1 aji    jarkom-a     0  May  3 10:38 MasukKompAji
-r--------  1 aji    jarkom-a     0  May  3 10:40 MasukKompRita
-r--------  1 aji    jarkom-a     0  May  3 10:41 MasukKompIlham
-rwxr-xr-x  1 aji    jarkom-a 13902  May  3 09:54 stream_server_seq
-rw-r--r--  1 aji    jarkom-a  2308  Mar  2 08:32 stream_server_seq.c
[aji@mkom22 server]$
```

Pada komputer client keempat

```
M.Kom Server
_____________

login: nurdin
```

```
Password:
Last login: Wed Apr 27 08:01:31 from 192.168.10.10
[nurdin@mkom22 nurdin]$ cd asep
[nurdin@mkom22 asep]$ cd Bergamo
[nurdin@mkom22 Bergamo]$ cd SockCode
[nurdin@mkom22 SockCode]$ cd SOCK_STREAM/
[nurdin@mkom22 SOCK_STREAM]$ gcc -o client stream_client.c
[nurdin@mkom22 SOCK_STREAM]$ ./client 192.168.10.26 MasukKompAsep
open file: No such file or directory
[nurdin@mkom22 SOCK_STREAM]$
```

Di komputer server

```
[aji@mkom22 server]$ Connection from address 192.168.10.26 port 32836
server read MasukKompAji
file do not exist, get it from client
copied 0 bytes
Connection from address 192.168.10.26 port 32837
server read MasukKompRita
file do not exist, get it from client
copied 0 bytes
Connection from address 192.168.10.26 port 32838
server read MasukKompIlham
file do not exist, get it from client
copied 0 bytes
Connection from address 192.168.10.26 port 32839
server read MasukKompNurdin
file do not exist, get it from client
copied 0 bytes
```

Cek file yang masuk ke server :

```
[aji@mkom22 server]$ ls -l
total 20
.
.
-r--------  1 aji    jarkom-a     0  May  3 10:38  MasukKompAji
-r--------  1 aji    jarkom-a     0  May  3 10:40  MasukKompRita
-r--------  1 aji    jarkom-a     0  May  3 10:41  MasukKompIlham
-r--------  1 aji    jarkom-a     0  May  3 10:43  MasukKompNurdin
.
.
```

1. Menguji koneksi client dan server dengan komputer yang sama dengan IP :127.0.0.1 (*loopback*)

Di Komputer server

```
[aji@mkom22 aji]$ ./server &
```

Menjalankan program server.

```
[aji@mkom22 aji]$ ./client 127.0.0.1 peserta.dpr
```

Menjalankan program client dengan masukan local address 127.0.0.1 dan nama file peserta.dpr

Output:

```
Connection from address 127.0.0.1 port 32884
server read peserta.dpr
file do not exist, get it from client
copied 0 bytes
```

Koneksi ke address 127.0.0.1 dengan port 32884.Server akan mencari file peserta.dpr. Karena file tidak ada maka akan diterima dari client.

2. Menguji koneksi client dan server dengan komputer yang sama tetapi dengan direktori yang berbeda

Di direktori (folder) client

```
[aji@mkom22 SOCK_STREAM]$ gcc -o client stream_client.c
[aji@mkom22 SOCK_STREAM]$ ./client 192.168.10.26 KirimDariFolderLain
Connection from address 192.168.10.26 port 32774
server read KirimDariFolderLain
file do not exist, get it from client
open file: No such file or directory
copied 0 bytes
[aji@mkom22 SOCK_STREAM]$ ./client 127.0.0.1 KirimDariFolderLainUlang
Connection from address 127.0.0.1 port 32775
server read KirimDariFolderLainUlang
file do not exist, get it from client
open file: No such file or directory
copied 0 bytes
```

Di direktori (folder) server

Kemudian dicek file yang masuk ke server :

```
[aji@mkom22 SOCK_STREAM]$ cd server
[aji@mkom22 server]$ ls -l
total 20
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLain
-r--------  1 aji    jarkom-a     0  May  3 09:56 KirimDariFolderLainUlang
-rwxr-xr-x  1 aji    jarkom-a 13902  May  3 09:54 stream_server_seq
-rw-r--r--  1 aji    jarkom-a  2308  Mar  2 08:32 stream_server_seq.c
[aji@mkom22 server]$
```

KESIMPULAN

Dari hasil bahasan dan pengujian maka didapatkan beberapa kesimpulan sebagai berikut :

1. Model sistem yang dibuat adalah sebuah aplikasi client-server, dengan jenis socket yang dipakai adalah stream socket. Sehingga program yang dibuat ada dua, yaitu program untuk bagian server dan bagian client.
2. Dalam melakukan komunikasi pertama dilakukan proses creating socket, yang dilanjutkan dengan proses bind dengan tujuan mengikat jaringan dengan port dan address tertentu.
3. Selanjutnya, server melakukan proses listen dan dilanjutkan proses perulangan yang bertujuan untuk mendengarkan adanya koneksi dari client yang masuk serta untuk menerima data dari socket yang terhubung dengan proses *receive*.
4. Hasil koneksi akan menunjukkan bahwa jika berhasil maka akan terdapat respon pengiriman dan penerimaan berhasil, sedangkan jika gagal akan diberikoan respon kesalahan (*error*).

DAFTAR PUSTAKA

1. Bergamo, P., 2004, "*Socket and Network Programming : Introduction and Definitions*", UCLA.
2. Irawan, I., 2003 "*Pemrograman Socket dengan C*", www. ilmukomputer. com, akses : 20 April 2005.
3. Petillot, Y., , "*Computer Network*", www.ece.eps.hw.ac.uk/~ceeyrp/home/ Pages/Teaching/B34LA1, akses : 2 Maret 2005.
4. Prasimax, 2002, "*Protocol TCP/IP*", Prasimax, Yakarta.
5. Stalling, W., 2000, "*Jaringan Komputer*", Salemba, Jakarta.