APLIKASI CHATTING DENGAN FASILITAS FTP DAN IP ADDRESS BLOCKING UNTUK CLIENT-SERVER MULTIPLE CONNECTIONS

Makalah Seminar Tugas Akhir

TOMMY BUDIANTO
L2F 302 532
Jurusan Teknik Elektro
Fakultas Teknik Universitas Diponegoro

ABSTRAK

Komunikasi antar pengguna komputer dalam suatu jaringan sangat diperlukan. Dapat dibayangkan jika pengguna jaringan komputer yang berada dalam suatu gedung terpisahkan oleh dinding gedung, tentu komunikasinya akan menjadi terhambat. Sistem chatting adalah sebuah aplikasi client-server yang bisa digunakan untuk mengatasi hal ini. Sistem chatting merupakan sebuah sistem untuk "bercakap-cakap" antar komputer dalam sebuah jaringan. Dalam implementasinya, sistem ini akan sangat efisien dan sangat ekonomis. Pemrograman client-server pada dasarnya adalah pemrograman socket dengan memanfaatkan lapisan transport yang ada pada model referensi Internet.

Sebagai salah satu sistem client-server maka sistem chatting dapat dibuat dengan mengunakan bahasa pemrograman yang banyak beredar saat ini, salah satunya adalah Visual Basic 6.0. Bahasa Visual Basic 6.0 sudah bisa mengakses socket pada sistem operasi Windows yang dinamakan Winsock API. Winsock API sudah mendukung protokol TCP/IP, protokol TCP/IP sangat handal dan hampir diterapkan pada semua aplikasi client-server pada saat ini. Tugas akhir ini akan menjelaskan pemodelan sistem berorientasi objek, analisis dan desain sistem chatting berbasis grafis yang dilengkapi dengan fasilitas FTP dan IP address blocking.

Sistem ini dibagi menjadi dua subkelas, yaitu subkelas client dan subkelas server. Subkelas server mempunyai kewenangan dalam mengatur subkelas client, sehingga aplikasi server berperan sebagai administrator. Setelah dilakukan pengujian, aplikasi server dalam sistem ini dapat mem-blok IP address client dan dapat menendang client keluar dari sistem. Sedangkan kecepatan melakukan transfer file pada aplikasi FTP yang disisipkan tergantung dari ukuran file yang akan ditransfer.

Kata Kunci : Jaringan, client, server, protokol, socket, Winsock API, TCP/IP.

I. PENDAHULUAN

1.1 Latar Belakang

Manajemen suatu jaringan dapat dilakukan dengan membuat aplikasi sendiri. Aplikasi untuk pemrograman jaringan meliputi cara penulisan program yang dapat digunakan untuk berkomunikasi dengan program yang lain di dalam sebuah jaringan komputer. Logikanya, komputer satu berhubungan dengan komputer lain tidak peduli apa pun jenis medianya pasti memerlukan sebuah protokol sebagai bahasa komunikasi. Protokol yang dominan dan banyak digunakan untuk berbagai macam aplikasi saat ini adalah *TCP (Transmission Control Protocol).*

Secara normal ada dua program untuk berkomunikasi di sebuah jaringan komputer, salah satu program dinamakan *client* dan program yang satunya dikatakan server. Penulisan program untuk jaringan menggunakan *Application Programming Interface* atau *API.* API adalah spesifikasi yang menjelaskan bagaimana program aplikasi dapat berinteraksi dengan sistem manajemen jaringan[4].

API yang secara umum digunakan secara standart sekarang ini adalah *socket* yang sudah mendukung protokol UDP dan TCP/IP, dan untuk sistem operasi windows dinamakan *winsock API* . Dengan menggunakan bahasa program Visual Basic kita dapat mengakses *winsock API* yang memang telah disediakan oleh windows tetapi dalam format l*ow level*, sehingga dapat dibuat program aplikasi *chatting* untuk *client-server.*

1.2 Tujuan

1. Memberikan kemudahan komunikasi secara *on- line* dari clien*t* ke client maupun dari client ke server dan kemudahan pengaturan hak akses *client* dengan fasilitas *IP address blocking.*
2. Memberikan kemudahan dalam proses transfer file dengan penambahan fasilitas FTP

1.3 Pembatasan Masalah

1. Mendesain dan membuat software aplikasi *chatting* untuk komunikasi *client-server*.
2. Menyisipkan aplikasi FTP pada form client dan server sebagai media transfer data antara client-server.
3. Menyisipkan fasilitas *IP address blocking* pada form server sebagai media untuk memberikan kemudahan manajemen jaringan bagi seorang administrator.
4. Software aplikasi ini akan berjalan pada sistem operasi Windows.
5. Pembuatan software aplikasi tidak akan mempertimbangkan kelebihan software serupa yang telah ada saat ini. Pertimbangan hanya akan dijadikan sebagai masukan dan sebagai referensi.

II. WINSOCK API SEBAGAI KOMPONEN UTAMA APLIKASI CHATTING MULTIPLE CONNECTIONS

2.1 Dasar Aplikasi Client-Server

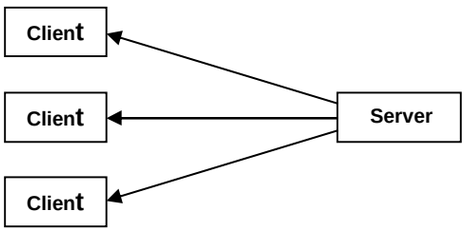
Koneksi antar komputer yang terbentuk adalah koneksi dari *socket* ke *socket. Socket* adalah dua buah nilai yang mengidentifikasi setiap *endpoint* sebuah alamat IP dan sebuah nomor port[4]. Analogikan socket sebagai pintu, untuk bisa berkomunikasi maka pintu di kedua komputer harus dalam keadaan terbuka. *Client*-lah yang dalam hal ini harus lebih pro-aktif untuk membuka koneksi terlebih dahulu, client harus "mengetuk pintu" dahulu pada *server*. Server memonitor "tiap ketukan" atau permohonan koneksi client, setiap koneksi memiliki ID yang unik. Setelah server menerima atau "membuka pintu", maka sebuah koneksi baru bisa terbentuk

sehingga baik client maupun server sudah bisa berkomunikasi dan saling bertukar data.

Konsep sederhana untuk memahami pembentukan koneksi client-server dapat diterangkan sebagai berikut :

1. Server membuat sebuah socket dengan menggunakan karakter unik (misalnya dengan penentuan alamat IP dan nomor port), yang dapat diidentifikasi dan ditemukan oleh client, pada saat ini server telah memasuki kondisi *listening*. Kondisi listening adalah keadaan di mana server dalam kondisi siap untuk menerima permintaan servis dari client.
2. Client membuat socket, mencari nama atau alamat socket server dan kemudian "menyambungkannya" untuk menginisialisasi sebuah komunikasi.
3. Setelah inisialisasi dilakukan maka client dan server sudah bisa saling mengirimkan data dan menerima data.

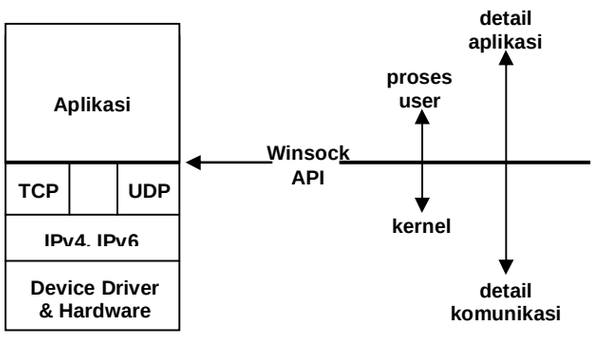
Secara sepintas, hubungan antara server dengan client-nya diperlihatkan pada gambar 2.1 :



Gambar 2.1 Hubungan suatu server aplikasi dengan beberapa client

Pada prinsipnya, server sebagai penyedia servis harus selalu siap sewaktu-waktu ketika ada sebuah client yang meminta servis. Namun dalam prakteknya, keterbatasan penyediaan servis juga disebabkan oleh keterbasan sistem itu sendiri, tepatnya kernel di dalam sistem tersebut. Misalnya ketika beberapa client secara bersamaan meminta servis, maupun karena aturan jaringan yang ditentukan oleh server itu sendiri, maka server akan menolak permintaan servis dari satu atau beberapa client. Contoh dari aturan ini adalah adanya alamat IP yang diblok oleh server sehingga client tidak bisa terkoneksi atau server bisa memaksa sebuah user untuk keluar dari sistem dengan cara meng-*kick*.

Sebenarnya detail proses dari pemrograman socket akan mengarah ke lapisan kernel dan komunikasi seperti pada model protokol Internet yang diperlihatkan pada gambar 2.2 berikut[4] :



Gambar 2.2 Winsock API menjembatani dua orientasi yang berbeda, satu ke arah detail aplikasi dan proses user sedang yang lain ke arah detail komunikasi dan kernel.

Dari gambar 2.2 di atas dapat kita lihat bahwa untuk menjembatani dua bagian yang berbeda orientasi maka digunakan API *(Application Programming Interface)*, dan dalam sistem operasi Windows dinamakan *Winsock API*.

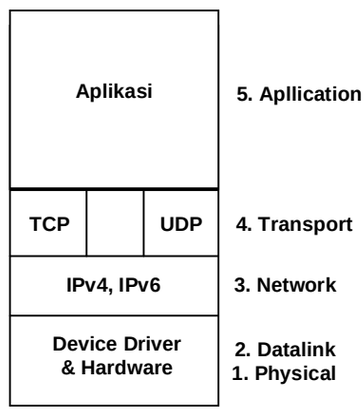
2.2 Winsock API

Windows memiliki API untuk berkomunikasi melalui TCP/IP yang terkenal dengan nama Winsock API. Winsock *(Windows Socket)* adalah antarmuka (interface) pemrograman jaringan yang digunakan untuk komunikasi data pada berbagai macam protokol jaringan komputer. Sebagai sebuah API, Winsock bekerja berdasarkan konsep socket. Socket merupakan sarana transformasi data yang memungkinkan sebuah aplikasi jaringan mengakses data pada sebuah jaringan komputer. Winsock API berperan sebagai tangan untuk menyediakan transportasi ke arah *low-level*. Windows tidak mengijinkan kita untuk secara langsung mengakses low-level, namun Windows menyediakan API sendiri. Alamat IP komputer yang akan dituju didefinisikan sebagai **RemoteIP** dan port yang akan dituju didefinisikan sebagai **RemotePort**. Jadi intinya adalah sepanjang kedua buah program aplikasi "berbicara" pada protokol yang sama dan protokol aplikasi yang sama, maka komunikasi antara keduanya dapat dilakukan.

Winsock API sebenarnya mendukung dua buah protokol transport Internet yang ada saat ini yaitu TCP dan UDP. Tetapi karena banyak faktor yang menyebabkan TCP lebih unggul dari pada UDP, maka jika diinginkan membuat sebuah program aplikasi client-server yang handal protokol TCP adalah sebuah pilihan yang tepat. Perbedaan mendasar antara protokol TCP dan UDP bisa dilihat pada sub bab 2.3, bab ini akan menjelaskan perbandingan di antara kedua protokol tersebut.

2.3 Protokol-protokol Transport Internet

Model referensi protokol Internet yang dijadikan acuan sebagai dasar pembuatan program aplikasi client-server adalah sebagai berikut[4] :



Gambar 2.3 Model referensi protokol Internet

Dari gambar 2.3 tersebut dapat dilihat bahwa protokol *transport* (pengangkut) yang didukung oleh model referensi protokol Internet ada dua buah yaitu TCP *(Transmission Control Protocol)* dan *UDP (User Datagram Protocol)*. Berikut ini dapat dilihat perbedaan perbedaan protokol TCP dengan UDP pada tabel 2-1 :

Tabel 2-1 Perbedaan TCP dengan UDP

TCP	UDP
- *Connection oriented*	- *Connectionless*
- *Reliable*	- *Unreliable*
- Ada RTT *(round-trip-time) estimation*	- Tidak ada RTT
- Ada antrian data	- Tidak ada antrian data
- Ada konfirmasi pengiriman data	- Tidak ada konfirmasi data
- Ada aliran kontrol data	- Tidak ada aliran kontrol data

Dari tabel perbandingan di atas jelas terlihat bahwa protokol TCP jauh lebih unggul dari pada protokol UDP.

III. PERANCANGAN DAN DESAIN SISTEM APLIKASI

Pada bab ini akan dijelaskan mengenai langkah-langkah dalam mendesain sekaligus membuat perangkat lunak aplikasi chatting dengan fasilitas FTP dan *IP address blocking* untuk *client-server multiple connections*. Perancangan tugas akhir ini terbagi menjadi dua bagian besar atau sub bab yaitu :

1. Analisis sistem menggunakan metode pemodelan berorientasi objek.
2. Desain sistem aplikasi berorientasi objek.

Sebuah perangkat lunak yang menganut *Object Oriented (OO)* akan menggunakan prinsip pendekatan terhadap tiga buah elemen untuk memodelkan sebuah sistem. Elemen tersebuat adalah adalah analisis berorientasi objek *(Object Oriented Analisis / OOA)*, desain berorientasi objek *(Object Oriented Design / OOD)* dan pemrograman berorientasi objek *(Object Oriented Programming / OOP)*.

OOA digunakan untuk merumuskan sebuah sistem yang baru akan dibuat. OOA menggambarkan objek-objek apa saja yang relevan, bagaimana objek-objek berhubungan satu dengan yang lainnya dan bagaimana objek-objek akan bertingkah laku di dalam konteks sistem tersebut. Dalam terminologi berorientasi objek, objek sebenarnya adalah anggota dari kelas. Dengan kata lain, kelas adalah kumpulan dari beberapa objek yang sama. Sasaran OOA adalah mengembangkan sederetan model yang menggambarkan cara kerja perangkat lunak di dalam sebuah sistem seperti yang dikehendaki oleh user / pemakai[3]. OOD digunakan untuk mentransformasi model analisis yang dibuat dengan OOA ke dalam suatu model desain yang berfungsi sebagai cetak biru perangkat lunak tersebut. OOD memungkinkan pembuat perangkat lunak untuk menunjukkan objek yang ditarik dari masing-masing kelas dan bagaimana objek itu saling berhubungan satu dengan yang lainnya. Sedangkan OOP digunakan untuk menerjemahkan kelas, atribut, operasi dan pesan yang dapat dieksekusi dalam sebuah mesin komputer.

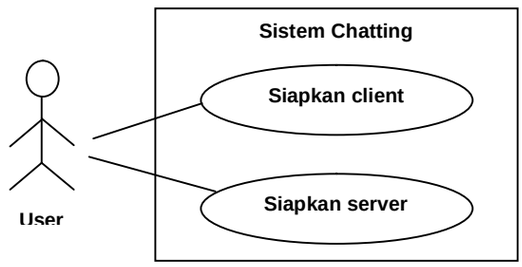
3.1 Analisis Sistem Berorientasi Objek

3.1.1 Use Case Sistem Aplikasi

Use case adalah serangkaian skenario yang masing-masing mengidentifikasi urutan pemakaian bagi sistem yang akan dibangun dan memberikan deskripsi bagaimana sistem tersebut akan digunakan[3]. Untuk membuat use case harus diidentifikasi tipe manusia (atau perangkat) yang berbeda yang menggunakan sistem (atau perangkat) yang berbeda yang menggunakan sistem atau produk tersebut. Aktor-aktor tersebut merepresentasikan peran yang dimainkan oleh manusia (atau perangkat pada saat sistem beroperasi). Aktor adalah sesuatu yang berkomunikasi dengan sistem atau produk eksternal terhadap sistem itu sendiri[3]. Aktor dan pemakai tidaklah sama, aktor merepresentasikan suatu kelas dari entitas eksternal yang hanya memainkan satu peran saja. Contoh dari aktor adalah sebuah tombol di sebuah sistem aplikasi, tombol tersebut akan menampilkan output tertentu pada saat ditekan. Sedangkan pemakai dapat memainkan sejumlah peranan yang berbeda pada saat menjalankan suatu sistem, misalnya pemakai dapat menekan beberapa tombol yang berbeda-beda fungsinya.

1. Use Case Sistem Chatting

Use case untuk sistem aplikasi chatting client-server ini bisa digambarkan pada Gambar 3.1 sebagai berikut :



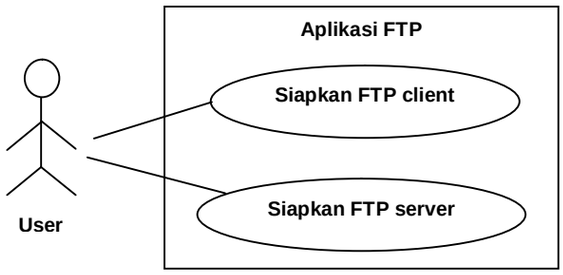
Gambar 3.1 Use case sistem chatting

Skenario normal untuk sistem aplikasi chatting client-server ini adalah sebagai berikut :

1. Server dijalankan pertama kali, pada saat ini server dikatakan dalam kondisi *listening* dan siap untuk menerima permintaan koneksi dari client.
2. Client siap melakukan koneksi ke server dengan menggunakan perintah `connect` , server lalu akan memproses permintaan koneksi dari client dengan menggunakan perintah `accept`.
3. Server mengirimkan balasan / notifikasi atas permintaan koneksi dari client, isinya adalah informasi bahwa client sudah terkoneksi dengan server. Bila notifikasi dari server sudah diterima maka client sudah terkoneksi dengan server, pada saat ini terjadi maka client dan server sudah bisa berkomunikasi saling mengirimkan pesan.
4. Bila ada permintaan koneksi dari client yang lain maka server akan melakukan hal yang sama pada point (1), (2) dan (3) di atas.
5. Masalah akan timbul pada client jika IP address-nya di *blocking* oleh server, jika ini terjadi maka client tidak akan bisa terkoneksi dengan server, yang terlihat dengan adanya notifikasi dari server.
6. Server berhak untuk meng-*kick* (menendang) client yang dikehendaki dari sistem.
7. Jika client ingin keluar dari sistem maka tinggal memutuskan koneksi dari server dengan memanggil perintah `close`.
8. Server mengirimkan notifikasi pemutusan dari client.
9. Client sudah tidak terkoneksi ke server.
10. Server berada dalam keadaan *listening* dan siap untuk memproses permintaan koneksi dari client yang lain.

2. Use Case Aplikasi FTP

Sedangkan use case untuk aplikasi FTP client-server diperlihatkan pada Gambar 3.2 sebagai berikut :



Gambar 3.2 Use case aplikasi FTP

Skenario normal untuk aplikasi FTP client-server ini adalah sebagai berikut :

1. FTP server dijalankan pertama kali, pada saat ini FTP server dikatakan dalam kondisi *listening* dan siap untuk menerima permintaan koneksi dari FTP client.
2. FTP client siap melakukan koneksi ke server dengan menggunakan perintah `connect`, FTP server lalu memproses permintaan dari client dengan menggunakan perintah `accept`.
3. FTP server mengirimkan balasan / notifikasi atas permintaan koneksi dari FTP client, isinya adalah informasi bahwa client sudah terkoneksi dengan server. Bila notifikasi dari server sudah diterima maka client sudah terkoneksi dengan server, pada saat ini terjadi maka client dan server sudah bisa berkomunikasi saling mentransfer file yang diinginkan.
4. Untuk mentransfer file dari client ke server, tinggal pilih file yang akan dikirim lalu kirim file tersebut, begitu juga sebaliknya.
5. Jika FTP client ingin memutuskan koneksi dari server, maka tinggal memanggil perintah `close`.
6. FTP server mengirimkan notifikasi dari pemutusan FTP client
7. FTP client sudah tidak terkoneksi ke server.
8. FTP server berada dalam keadaan listening dan siap untuk memproses permintaan koneksi dari FTP client yang lain.

3.1.2 CRC (Class-Responsibility-Collaboration)

CRC atau kelas-tanggung-jawab-kolaborator memberikan cara sederhana untuk mengidentifikasi dan mengumpulkan kelas-kelas yang relevan dengan sebuah sistem. *Tanggung jawab* adalah atribut dan operasi untuk kelas, pendeknya tanggung jawab adalah "sesuatu yang diketahui atau dilakukan oleh kelas"[3]. *Kolaborator* adalah kelas-kelas yang diperlukan untuk memberikan informasi yang dibutuhkan yang diperlukan oleh kelas untuk menyelesaikan tanggung jawab[3]. Sedangkan kelas adalah konsep *object oriented* yang mengkapsulasi data dan abstraksi prosedural yang diperlukan untuk menggambarkan isi serta tingkah laku berbagai entitas dunia nyata[3].

1. CRC Sistem Chatting

CRC untuk sistem aplikasi chatting diperlihatkan pada Gambar 3.3, Gambar 3.4 dan Gambar 3.5 :

Class
Sistem Chatting
Responsibility
1. Siapkan server 2. Siapkan client
Collaboration
1. Class Server 2. Class Client

Gambar 3.3 CRC kelas sistem chatting

Class
Server
Responsibility
1. Memproses koneksi client 2. Melakukan blocking alamat IP client 3. Berkomunikasi (chatting) dengan client 4. Menendang (kick) client keluar dari sistem 5. Memberikan notifikasi pemutusan client
Collaboration
Class Client

Gambar 3.4 CRC kelas server

Class
Client
Responsibility
1. Melakukan koneksi dengan server 2. Berkomunikasi (chatting) dengan client lain dan server 3. Memutuskan koneksi dengan server
Collaboration
Class Server

Gambar 3.5 CRC kelas client

2. CRC Aplikasi FTP

CRC aplikasi FTP client-server diperlihatkan pada Gambar 3.6, Gambar 3.7 dan Gambar 3.8 :

Class
Aplikasi FTP
Responsibility
1. Siapkan FTP server 2. Siapkan FTP client
Collaboration
1. Class FTP server 2. Class FTP client

Gambar 3.6 CRC kelas aplikasi FTP

Class
FTP Server
Responsibility 1. Memproses koneksi FTP client 2. Mengirimkan file ke client 3. Memberikan notifikasi pemutusan FTP client
Collaboration Class FTP client

Gambar 3.7 CRC kelas FTP server

Class
FTP Client
Responsibility 1. Melakukan koneksi ke FTP server 2. Mengirimkan file ke server 3. Memutuskan koneksi dari FTP server
Collaboration Class FTP server

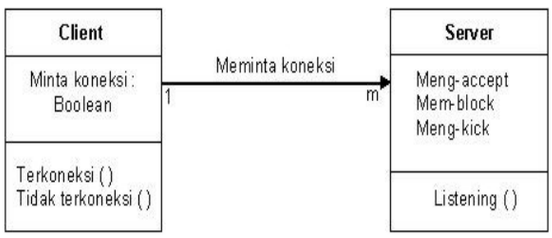
Gambar 3.8 CRC kelas FTP client

3.1.3 Hubungan Objek

Dari CRC yang telah diuraikan, maka selanjutnya ditentukan hubungan antar kelas. *Hubungan* ada di antara setiap dua kelas yang dihubungkan[3]. Begitu hubungan telah ditentukan maka masing-masing ujung dievaluasi untuk menentukan *kardinalitas*. Kardinalitas adalah hubungan antar objek yang memberikan spesifikasi dari sejumlah peristiwa dari satu objek yang dapat dihubungkan ke sejumlah peristiwa dari objek yang lain[3].

1. Hubungan Objek Sistem Chatting

Hubungan objek untuk sistem aplikasi chatting ini diperlihatkan pada Gambar 3.9 berikut :

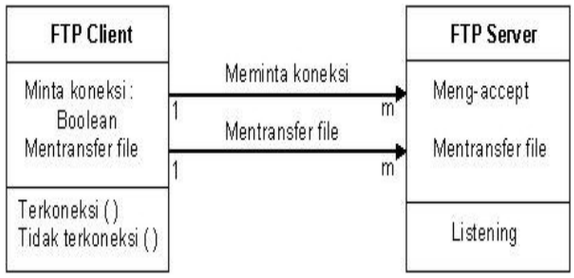


Gambar 3.9 Hubungan objek sistem chatting

Dari gambar di atas terlihat bahwa kardinalitas hubungan adalah 1:m, artinya dalam suatu jaringan sebuah client hanya dapat meminta koneksi kepada sebuah server, sedangkan server dapat memproses atau memberikan koneksi kepada beberapa client.

2. Hubungan Objek Aplikasi FTP

Sedangkan hubungan objek untuk aplikasi FTP diperlihatkan pada Gambar 3.10 sebagai berikut :



Gambar 3.10 Hubungan objek aplikasi FTP

Pada aplikasi FTP di atas juga mempunyai kardinalitas hubungan 1:m, baik dalam hal permintaan koneksi maupun untuk transfer file. Sebuah FTP client hanya dapat meminta koneksi kepada sebuah FTP server, sedangkan FTP server dapat memproses ataupun memberikan koneksi kepada beberapa client. Untuk transfer file, sebuah client hanya bisa mentransfer file kepada sebuah server, sedangkan server bisa mentransfer file kepada beberapa client.

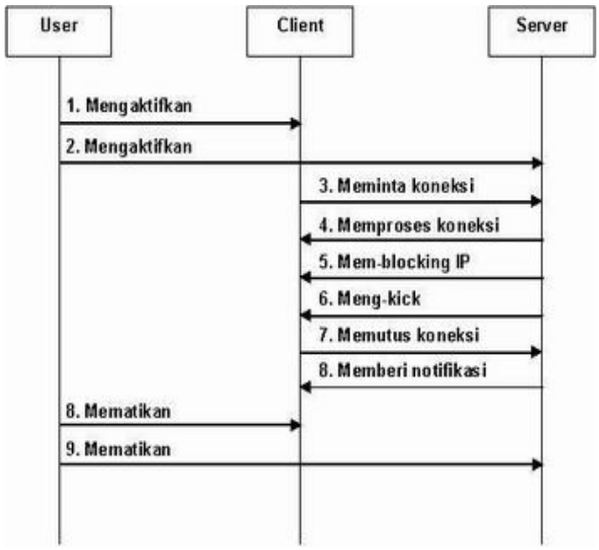
3.1.4 Tingkah Laku Objek

Model tingkah laku objek menunjukkan bagaimana sistem akan merespon kejadian atau stimulus eksternal[3]. Dalam memodelkan objek maka use case suatu sistem harus benar-benar dipahami. Use case untuk sistem aplikasi chatting dan aplikasi FTP ini telah dijelaskan pada sub bab 3.1.1 terdahulu. Tingkah laku objek dalam sub bab ini akan dilakukan dengan melakukan penelusuran event untuk masing-masing use case dan membuat diagram transisi state untuk sistem.

1. Penelusuran Event Use Case

a. Sistem Aplikasi Chatting

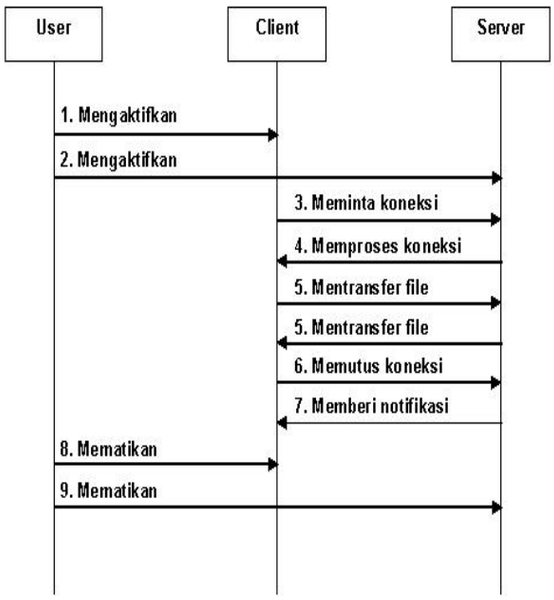
Penelusuran event use case untuk sistem chatting diperlihatkan pada Gambar 3.11 sebagai berikut :



Gambar 3.11 Event use case sistem chating

b. Aplikasi FTP

Penelusuran use case untuk aplikasi FTP diperlihatkan pada Gambar 3.12 sebagai berikut :



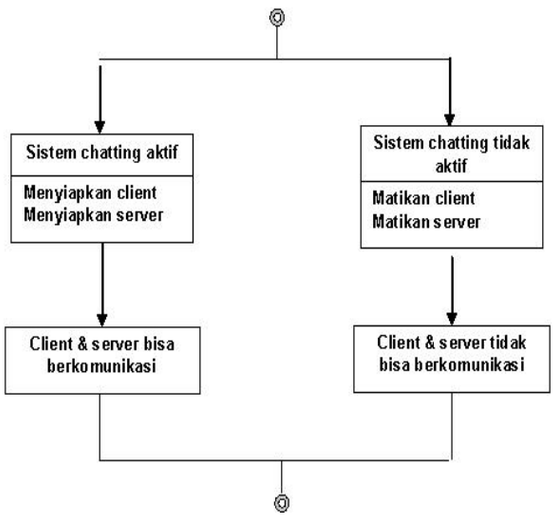
Gambar 3.12 Event use case aplikasi FTP

Dalam kedua buah gambar penelusuran event use case tersebut, masing-masing anak panah merepresentasikan suatu event (ditarik dari sebuah use case) dan menunjukkan bagaimana event menyalurkan tingkah laku di antara objek.

2. Diagram Transisi State

a. Sistem Aplikasi Chatting

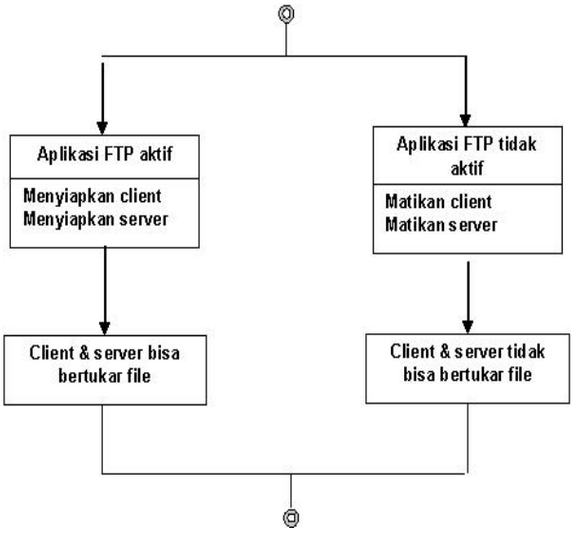
Diagram transisi state untuk sistem aplikasi chatting diperlihatkan pada Gambar 3.13 sebagai berikut :



Gambar 3.13 Diagram transisi state sistem chatting

b. Aplikasi FTP

Diagram transisi state untuk aplikasi FTP diperlihatkan pada Gambar 3.14 sebagai berikut :



Gambar 3.14 Diagram transisi state aplikasi FTP

Dua buah gambar diagram transisi state tersebut mempresentasikan tingkah laku dari suatu sistem dengan menggambarkan keadaannya dan kejadian yang menyebabkan sistem mengubah keadaan.

3.2 Desain Sistem Berorientasi Objek

Desain berorientasi objek *(object oriented design / OOD)* mentransformasi model analisis yang dibuat dengan menggunakan OOA ke dalam suatu model desain yang berfungsi sebagai cetak biru perangkat lunak yang akan dibuat. Desain sebuah aplikasi yang berorientasi objek harus mengacu kepada prinsip desain subsistem, desain kelas dan objek, desain pesan dan desain tanggung jawab. Desain subsistem diperoleh dengan mempertimbangkan keseluruhan persyaratan user (direpresentasikan dengan use case) dan event serta keadaan yang dapat diamati secara eksternal (model tingkah laku objek). Desain kelas dan objek diperoleh dari atribut, operasi dan kolaborasi yang diisikan pada model CRC. Sedangkan desain pesan dikendalikan oleh model hubungan objek. Pada desain aplikasi ini hanya akan dibahas tiga desain saja yaitu desain subsistem, desain kelas dan objek dan desain pesan.

3.2.1 Desain Subsistem

Dalam membuat desain sistem berorientasi objek harus dilakukan partisi terhadap model analisis untuk menentukan kumpulan kelas, hubungan dan tingkah laku. Semua elemen ini dikemas sebagai *subsistem*[3]. Subsistem harus memiliki interface yang terdefinisi secara baik sehingga semua bagian-bagiannya dapat berkomunikasi dengan sistem.

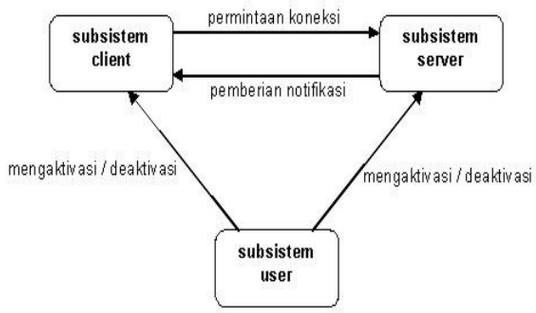
a. Sistem Chatting

Tabel kolaborasi subsistem untuk sistem chatting adalah sebagai berikut :

Tabel 3-1 Kolaborasi subsistem untuk sistem chatting

Kontrak	Tipe	Kolaborasi	Kelas	Operasi
1.Subsistem server	Client-server	Subsistem client	Server	Memberi koneksi
2.Subsistem client	Client-server	Subsistem server	Client	Meminta koneksi
3.Subsistem user	-	- Subsistem client - Subsistem server	Sistem Chatting	Mengaktifkan / Me-nonaktifkan

Dari tabel 3-1 tersebut dapat dibuat grafik kolaborasi subsistem sebagai berikut :



Gambar 3.15 Grafik kolaborasi subsistem untuk sistem chatting

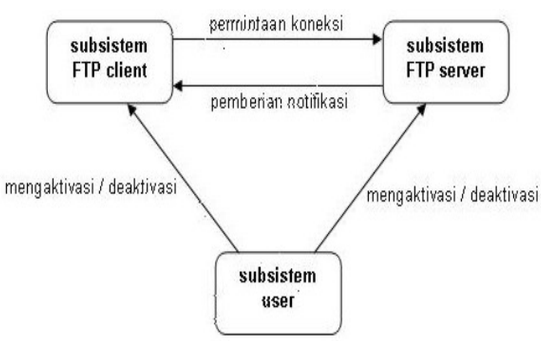
b. Aplikasi FTP

Karena aplikasi ini juga menganut sistem client-server maka tabel dan grafik kolaborasinya pun hampir sama dengan sistem chatting. Tabel kolaborasi subsistem untuk aplikasi FTP adalah sebagai berikut :

Tabel 3-2 Kolaborasi subsistem untuk aplikasi FTP

Kontrak	Tipe	Kolaborasi	Kelas	Operasi
1. Subsistem FTP server	Client-server	Subsistem FTP client	FTP server	Memberi koneksi
2. Subsistem FTP client	Client-server	Subsistem FTP server	FTP client	Meminta koneksi
3. Subsistem FTP user		- Subsistem FTP client - Subsistem FTP server	Sistem Chatting	Mengakftikan / Me-nonaktifkan

Dari tabel 3-2 di atas dapat dibuat grafik kolaborasi subsistem sebagai berikut :



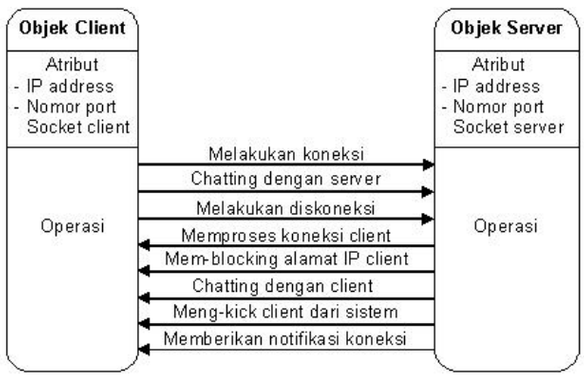
Gambar 3.16 Grafik kolaborasi subsistem untuk sistem chatting

3.2.2 Desain Kelas dan Objek

Dalam konteks desain berorientasi objek, desain objek harus mengembangkan desain detail mengenai atribut dan operasi yang membangun masing-masing kelas. Desain objek sudah dapat mewakili sebuah kelas karena setiap objek mewarisi sifat dari kelasnya. Dari CRC proses OOA maka sudah bisa dibuat desain objek untuk sistem chatting dan aplikasi FTP.

a. Sistem Chatting

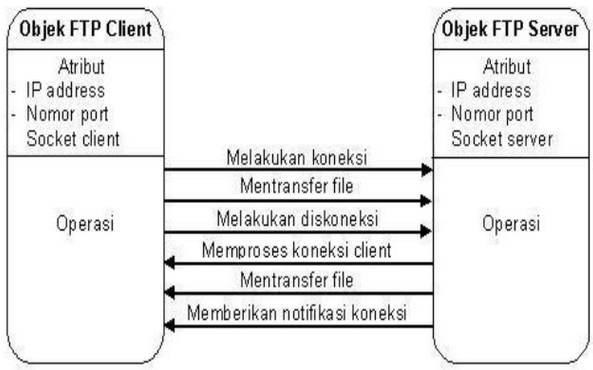
Desain objek sistem chatting bisa direpresentasikan dengan gambar 3.17 berikut :



Gambar 3.17 Desain objek sistem chatting

b. Aplikasi FTP

Desain objek aplikasi FTP bisa direpresentasikan dengan gambar 3.18 berikut :



Gambar 3.18 Desain objek aplikasi FTP

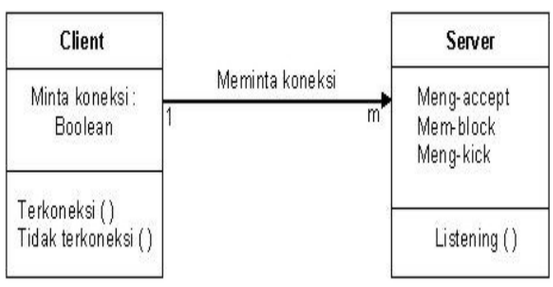
3.2.3 Desain Pesan

Pesan adalah alat di mana objek-objek berinteraksi[3]. Pesan menstimulasi banyak tingkah laku yang ditemui di dalam objek penerimaan. Operasi di dalam suatu objek pengirim memunculkan pesan dengan bentuk :

pesan : [tujuan, operasi, parameter]

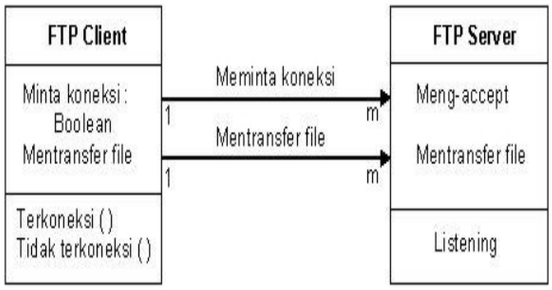
di mana tujuan menentukan objek penerima yang distimulasi oleh pesan tersebut, operasi mengacu pada metode yang akan menerima pesan dan parameter memberikan informasi yang dibutuhkan agar operasi berhasil. Sebuah pesan akan muncul apabila ada sebuah operasi / permintaan dari suatu objek yang ditujukan untuk objek lain di dalam sebuah sistem.

a. Sistem Chatting



Gambar tersebut adalah Gambar 3.9 hubungan objek sistem chatting, dari gambar tersebut terlihat bahwa objek client meminta koneksi kepada objek server. Pesan yang muncul dari adanya operasi tersebut adalah sebagai berikut :
pesan : [server, meng-accept, permintaan koneksi]
Sebagai balasan dari operasi dari client maka server pun bisa mengirim pesan sebagai berikut :
pesan : [client, permintaan koneksi di-accept]

b. Aplikasi FTP



Dari Gambar 3.10 hubungan objek aplikasi FTP di atas maka akan muncul desain pesan sebagai berikut :
pesan : [FTP server, meng-accept, permintaan koneksi]
pesan : [FTP server, mentransfer file, permintaan transfer file]
Sebagai balasan dari operasi dari FTP client maka FTP server pun bisa mengirim pesan sebagai berikut :
pesan : [FTP client, permintaan koneksi di-accept]
pesan : [FTP client, permintaan transfer file diterima]

IV. IMPLEMENTASI DAN PENGUJIAN

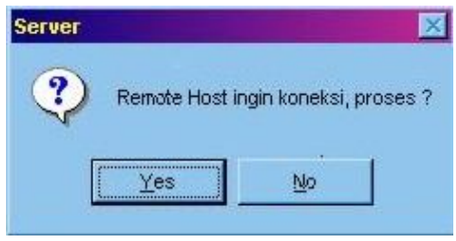
Bab ini akan menjelaskan implementasi dan pengujian software aplikasi sistem chatting dengan fasilitas FTP dan IP address blocking, project aplikasi ini terdiri atas dua bagian besar yaitu client dan server. Dalam aplikasi client terdapat tiga form yaitu frmClient, ftpClient dan frmHelp. Pada aplikasi server terdapat enam form yaitu frmBlock, frmClientInfo, frmHelp, frmServer, frmSetting dan ftpServer. Modul standart yang digunakan secara bersama oleh client dan server adalah clientstatus, modSetting dan ragam. Modul lain yang digunakan adalah modftpclient untuk client dan modftpserver untuk server.

Pada pengujian ini penulis akan menggunakan strategi pengujian berorientasi objek dengan metode desain pengujian berdasarkan skenario *(scenario-based).* Pengujian berdasarkan skenario berkosentrasi pada apa yang dilakukan pemakai dan juga pada apa yang dilakukan oleh sistem[3]. Ini berarti pemberian tugas-tugas (melalui use case) kepada pemakai sebagai penguji untuk mengaplikasikan sebuah sistem yang telah dibuat. Pengujian berdasarkan skenario cenderung menggunakan subsistem bertingkat di dalam suatu pengujian tunggal, artinya pemakai tidak membatasi diri dengan menggunakan satu sistem pada suatu waktu. Berikut ini adalah pengujian dan analisa sistem chatting dengan fasilitas FTP dan IP address blocking dengan metode *scenario-based.*

4.1 Pengujian dan Analisa Sistem Chatting

4.1.1 Pengujian Use Case Client Meminta Koneksi

Pengujian ini menunjukkan use case permintaan koneksi client ke server. Penulis menggunakan dua buah aplikasi client dengan nickname **Client_1** dan **Client_2** untuk melakukan koneksi ke server.
Sebelum melakukan koneksi sebaiknya kotak nickname diisi dahulu supaya ID client tidak tertulis sebagai **"Guest"** pada *chatroom.* Pada saat client melakukan koneksi maka akan muncul sebuah pesan pada server seperti Gambar 4.1 berikut :

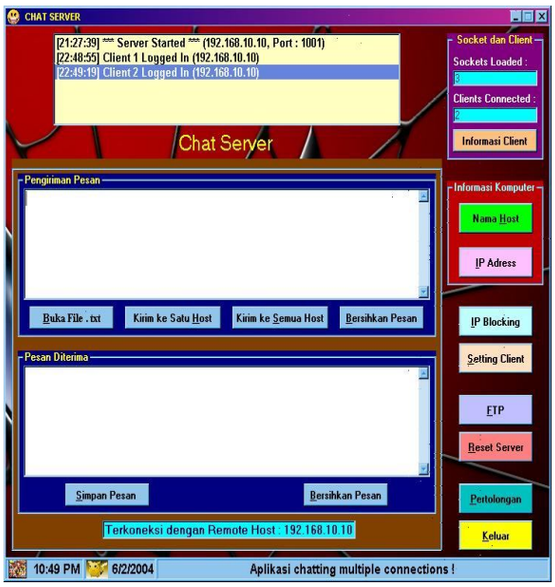


Gambar 4.1 Pesan pada server saat client meminta koneksi

Tampilan form Server setelah memproses koneksi yang diminta oleh **Client_1** ditunjukkan oleh Gambar 4.2.

Gambar 4.2 Tampilan form Server setelah Client_1 terkoneksi

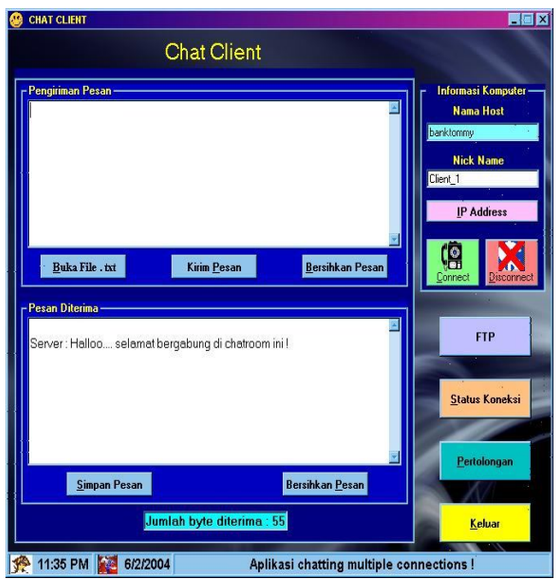
Pada saat **Client_2** melakukan koneksi juga akan muncul konfirmasi pesan koneksi seperti pada Gambar 4.1, sedangkan pada kotak status form Server akan muncul informasi tentang **Client_2**. Hasil dari koneksi Client_2 ditunjukkan oleh Gambar 4.3.



Gambar 4.3 Tampilan form Server setelah Client_2 terkoneksi

4.1.2 Pengujian Use Case Server Mengirimkan Notifikasi

Pengujian ini menunjukkan use case pemberian notifikasi server kepada client yang menyatakan bahwa client sudah terkoneksi dan siap untuk melakukan *chat* di dalam sistem chatting. Implementasi use case ini adalah dikirimkannya notifikasi pesan untuk client. Hasil pengujian ditunjukkan pada Gambar 4.4, notifikasi ini akan sama dan berlaku untuk setiap client yang permintaan koneksinya telah disetujui oleh server.

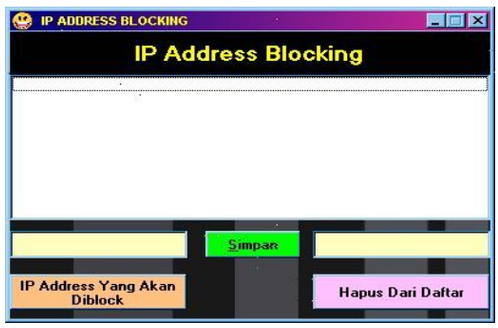


Gambar 4.4 Notifikasi server atas permintaan koneksi Client_1

Informasi jumlah byte yang diterima menunjukkan jumlah karakter data yang dikirimkan oleh server, setiap satu karakter ASCII dihitung 1 byte. Jumlah byte data yang diterima ini ditampilkan pada label **informasi** di bagian bawah aplikasi client.

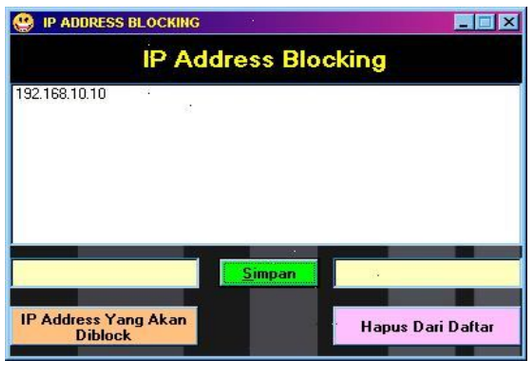
4.1.4 Pengujian Use Case Server Mem-blocking IP Address Client

Pengujian ini merupakan implementasi dari use case pem-*blocking*-an IP address sebuah client oleh server. Untuk melakukan hal ini terlebih dahulu aktifkanlah form **IP Address Blocking** pada form Server dengan menekan tombol **IP Blocking**. Tampilan form IP Address Blocking ditunjukkan oleh Gambar 4.5 sebagai berikut :



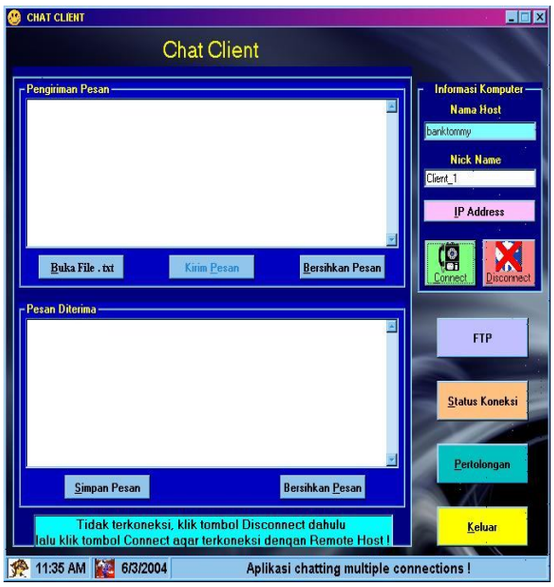
Gambar 4.5 Tampilan awal form IP Address Blocking

Untuk melakukan *blocking* terhadap IP address client masukkanlah terlebih dahulu IP address client yang akan di-*block*, lalu tekan tombol **IP Address Yang Akan Diblock**, lalu tekan tombol simpan. Sebagai contoh pada pengujian kali ini penulis menggunakan IP address 192.168.10.10, tampilan form setelah penginputan IP address ini ditunjukkan oleh Gambar 4.6 berikut :



Gambar 4.6 Tampilan form setelah penginputan IP address client

Untuk menguji hasil dari pem-blocking-an IP address client ini maka **Client_1** mencoba melakukan koneksi ke server (**Client_2** pun memiliki IP address yang sama karena pengujian ini dilakukan pada sebuah komputer menggunakan terminal program). Hasil pengujian ini ditunjukkan oleh Gambar 4.7 berikut :

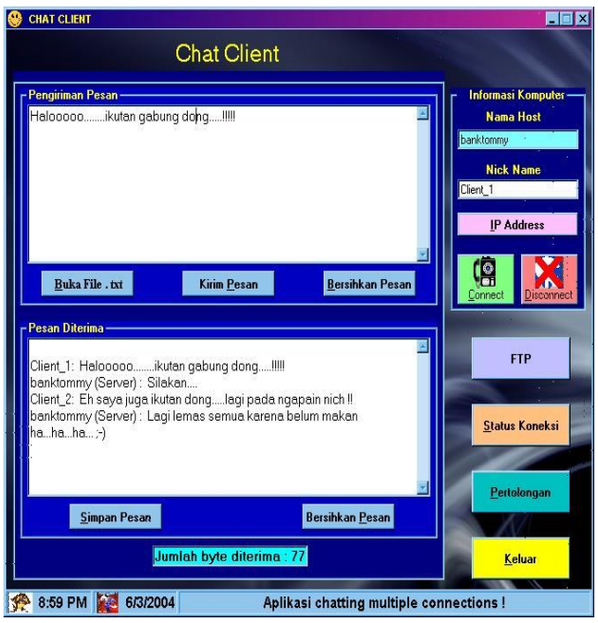


Gambar 4.7 Client_1 tidak bisa terkoneksi dengan server karena IP address-nya di-blocking

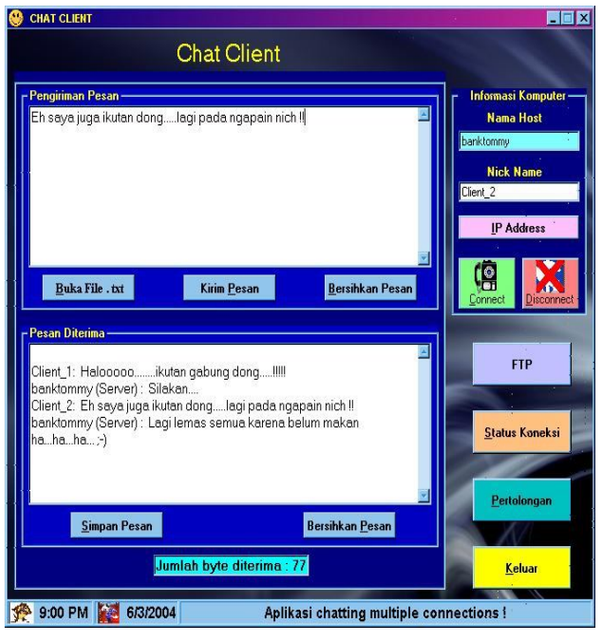
Dari Gambar 4.7 terlihat bahwa **Client_1** tidak bisa terkoneksi dengan server karena IP address-nya ter-*blocking*.

Pengujian Use Case Client Melakukan Chatting dengan Server

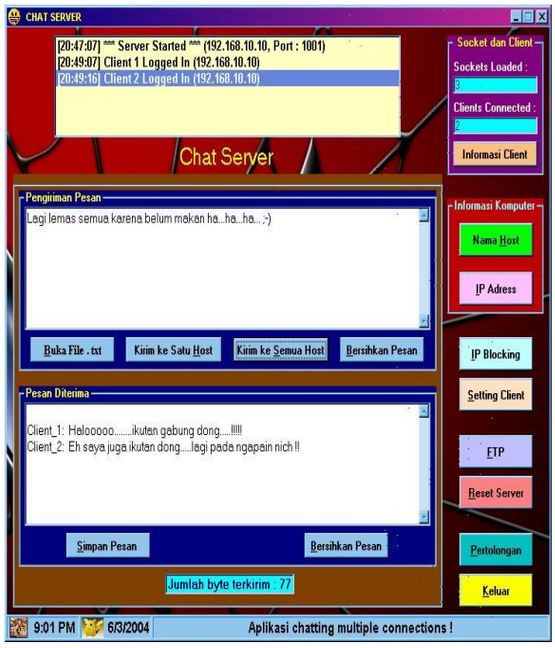
Pengujian untuk use case ini adalah inti dari sistem aplikasi yang dibuat, dimana client bisa semua objek yang ada pada sistem chatting ini bisa saling "bercakap-cakap" satu sama lain. Seperti pada pengujian-pengujian terdahulu, kali ini pun penulis menggunakan dua buah client yaitu **Client_1** dan **Client_2** untuk implementasi. Gambar hasil pengujian ini ditunjukkan oleh Gambar 4.8, Gambar 4.9 dan Gambar 4.10.



Gambar 4.8 Tampilan hasil chatting pada Client_1



Gambar 4.9 Tampilan hasil chatting pada Client_2



Gambar 4.10 Tampilan hasil chatting pada server

Proses yang terjadi pada chatting tersebut yaitu pertama-tama **Client_1** menulis pesan, server membalas pesan **Client_1**, **Client_2** menulis pesan dan server membalas pesan **Client_2**.

Pada pengujian use case client melakukan chatting dengan server yang telah dilakukan, server mengirimkan pesannya kepada semua client yang berada pada sistem chatting dengan menggunakan tombol **Kirim ke Semua Host**. Server juga bisa mengirimkan pesan hanya kepada satu client dengan menggunakan tombol **Kirim ke Satu Host**.

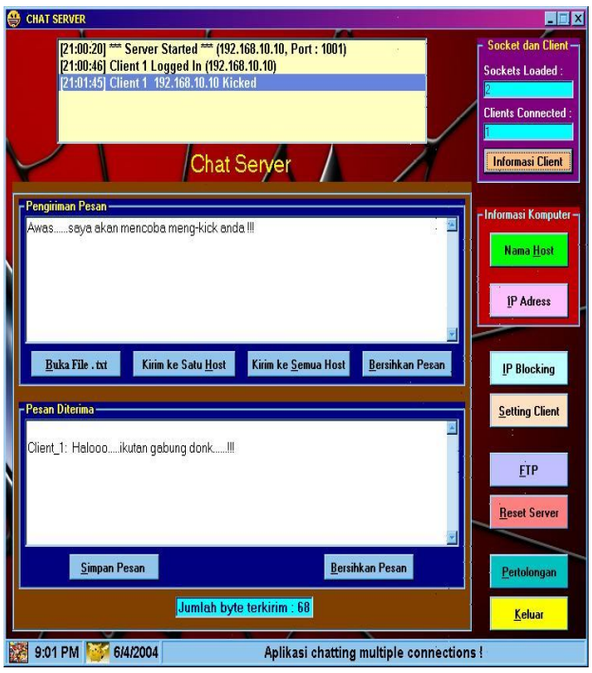
Untuk mengirimkan pesan kepada salah satu client, aktifkanlah form Infomasi Client dengan menekan tombol **Informasi Client**. Tampilan form Infomasi Client ditunjukkan oleh Gambar 4.11 berikut :

Gambar 4.11 Tampilan form Informasi Client

Pilihlah **Client ID** yang akan dikirimi pesan pada daftar tersebut, lalu tekan tombol **Pakai**. Tulis pesan yang akan dikirim lalu tekan tombol **Kirim ke Satu Host**, maka pesan tersebut akan dikirimkan ke salah client yang dituju oleh server.

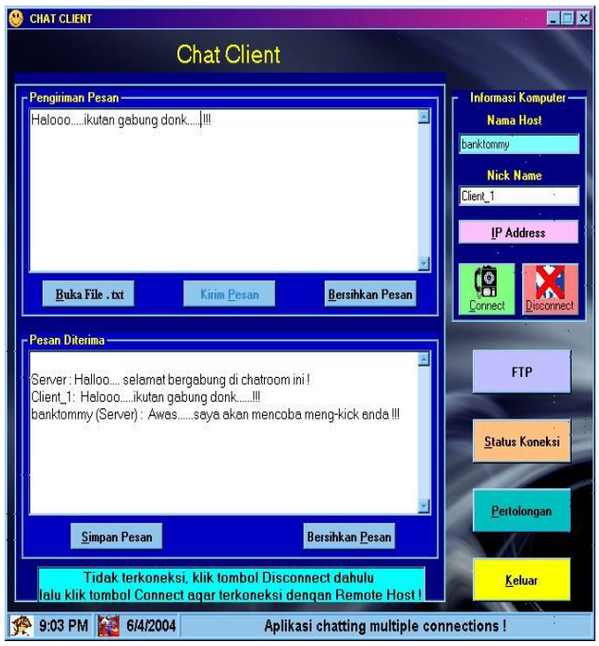
4.1.6 Pengujian Use Case Server Meng-kick Client

Pengujian use case ini merupakan implementasi dari kewenangan yang dimiliki oleh sistem chatting pada umumnya. Server berhak "menendang" siapa saja yang tergabung dalam sistem ini. Untuk meng-*kick* client, harus diaktifkan terlebih dahulu form Informasi Client, pilih client pada daftar lalu tekan tombol **Kick**. Hasil pengujian use case ini ditunjukkan oleh Gambar 4.12.



Gambar 4.12 Tampilan form server setelah meng-kick client

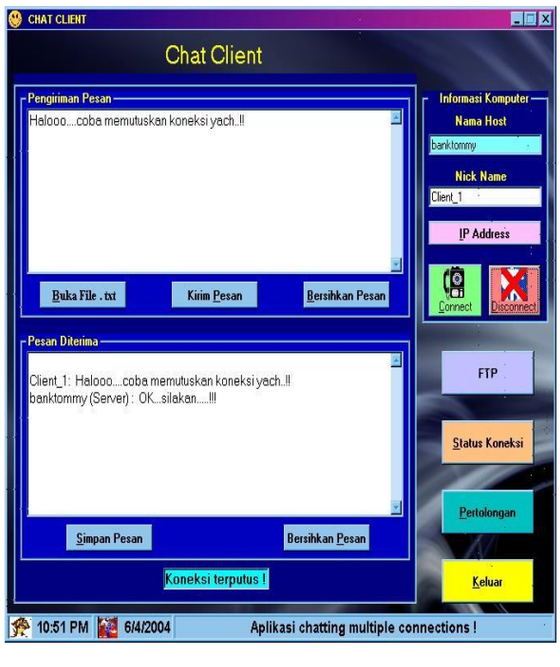
Sedangkan pada form Client informasi tentang diputuskannya koneksi client oleh server ditampilkan oleh **label info** di bagian bawah form aplikasi. Hasil tampilan pengujian pada client ditunjukkan oleh Gambar 4.13.



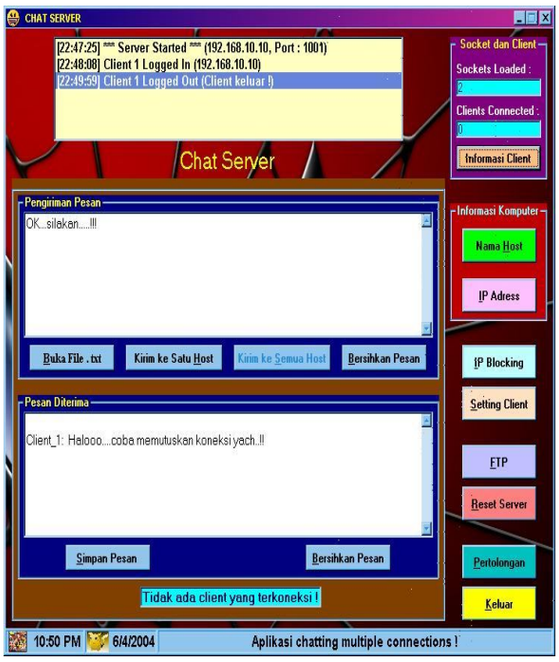
Gambar 4.13 Tampilan client setelah di-kick oleh server

4.1.7 Pengujian Use Case Client Memutuskan Koneksi dari Server

Pengujian ini merupakan implementasi dari use case client memutuskan koneksi dari server dengan menekan tombol **Disconnect**. Hasil pengujian ditunjukkan oleh Gambar 4.14 dan Gambar 4.15.



Gambar 4.14 Tampilan hasil pemutusan koneksi pada form Client



Gambar 4.15 Tampilan hasil pemutusan koneksi pada form Server

Use case server memberikan notifikasi pemutusan koneksi client secara otomatis juga sudah terwakili oleh use case ini karena kode program ini sudah dilengkapi dengan perintah untuk menampilkan pesan status koneksi.

4.2 Pengujian Dan Analisa Aplikasi FTP

4.2.1 Pengujian Use Case FTP Client Melakukan Koneksi ke FTP Server

Pengujian ini merupakan implementasi dari use case FTP Client melakukan koneksi ke FTP Server dengan menekan tombol **Connect** pada form FTP Client. Hasil pengujian ini diimplementasikan menjadi satu bagian dengan hasil pengujian use case pemrosesan koneksi oleh FTP Server.

4.2.2 Pengujian Use Case FTP Server Memproses Koneksi

Sebagai tindak lanjut dari use case permintaan koneksi dari FTP Client maka FTP Server akan memproses permintaan koneksi tersebut. Hasil kejadian ini ditunjukkan oleh Gambar 4.16 dan Gambar 4.17.

Gambar 4.16 Tampilan FTP Client setelah melakukan koneksi ke FTP Server

Gambar 4.17 Tampilan FTP Server saat terkoneksi dengan FTP Client

4.2.3 Pengujian Use Case Mentransfer File

Pengujian terhadap use case mentransfer file adalah inti dari aplikasi FTP yang disisipkan pada sistem chatting. Implementasi use case ini ditunjukkan oleh Gambar 4.18 dan Gambar 4.19 sebagai hasil yang diperoleh pada pengujian. Pada pengujian ini yang melakukan proses pengiriman file adalah FTP Client, hasil pengujian yang sama juga akan didapatkan apabila yang melakukan pengiriman file adalah FTP Server.Untuk mentransfer file, pertama tekan tombol **Buka File**, pilih file yang akan dikirim lalu tekan tombol **Kirim**. Gambar hasil pengujian pada form FTP Client ditunjukkan oleh Gambar 4.18.

Gambar 4.18 Tampilan FTP Client saat mentransfer data

Pada saat FTP Client sedang mentransfer data, pada FTP Server juga akan menampilkan informasi tentang status jumlah byte pengiriman data dari FTP Client. Hasil pengujian ditunjukkan oleh Gambar 4.19 berikut :

Gambar 4.19 Tampilan FTP Server saat menerima data

4.2.4 Pengujian Use Case FTP Client Memutuskan Koneksi

Pengujian ini merupakan implementasi dari use case pemutusan koneksi yang dilakukan oleh FTP Client. Setelah dilakukan pengujian use case ini maka hasil gambar tampilan yang diperoleh pada client menunjukkan status tidak terkoneksi dan pada server menunjukkan status tidak terkoneksi tetapi dalam kondisi *listening*. Secara otomatis, use case pemberian notifikasi FTP Server terhadap pemutusan koneksi FTP Client juga sudah diwakili oleh use case pemutusan koneksi ini karena label status memberikan informasi tentang status koneksi yang sama.

V. PENUTUP

5.1 Kesimpulan

Dari pengujian yang telah dilakukan pada tugas akhir ini dapat ditarik kesimpulan sebagai berikut :

1. Sistem chatting ini menggunakan model *client-server* dengan sistem manajemen terpusat, artinya setiap client yang ingin bergabung harus mendapat persetujuan terlebih dahulu dari server, sistem berjalan dengan baik.
2. Dalam analisis dan desain diperoleh tiga kelas, yaitu kelas sistem chatting, kelas server dan kelas client. Interaksi antar objek dilakukan dengan pengiriman pesan-pesan dari satu objek ke objek yang lain (*message passing*) seperti use case permintaan koneksi yang dilakukan client.

5.2 SARAN

Setelah melakukan pemodelan dan pengujian aplikasi chatting dengan fasilitas FTP dan IP address *blocking* ini, saran yang dianjurkan penulis untuk pengembangan sistem lebih lanjut yaitu pemberian fasilitas tambahan seperti multimedia, *videostreaming* dan penggunaan ikon-ikon ekspresi perasaan sehingga lebih menarik bagi para pengguna / user.

DAFTAR PUSTAKA :

1. Halvorson, M., ***"Step by Step Microsoft Visual Basic 6.0 Professional"*** (terjemahan), Jakarta : PT. Elex Media Komputindo, 2000.
2. Heywood, D., ***"Konsep & Penerapan Microsoft TCP/IP"*** (terjemahan), Yogyakarta : Andi, 1997.
3. Pressman, Roger S.,Ph.D., ***"Rekayasa Perangkat Lunak, Pendekatan Praktisi, Buku Dua"*** (terjemahan), Yogyakarta : Andi, 2002.
4. Stevens, Richard W., ***"UNIX Network Programming Volume I, Networking APIs: Sockets and XTI"***, Prentice-Hall,Inc.,1998.
5. Tanenbaum, Andrew S., ***"Jaringan Komputer Jilid I"*** (terjemahan), Jakarta : PT. Prenhallindo, 1996.
6. Wahana Komputer Semarang, ***"Panduan Praktis Pemrograman Visual Basic 6.0 Tingkat Lanjut"***, Yogyakarta : Andi, 2002.

Tommy Budianto, Mahasiswa Ekstensi angkatan 2002, lulusan Program Studi Diploma III Elektro UNDIP tahun 2002. Saat ini sedang menyelesaikan studi Srata -1 (S1) di Jurusan Teknik Elektro UNDIP dengan konsentrasi Teknik Informatika & Komputer.
E-mail: this-man@telkom.net

Mengetahui/Mengesahkan:

Pembimbing I	Pembimbing II
Agung BP ST, MIT	Aghus Sofwan ST, MT
NIP.132 137 932	NIP. 132 163 757