Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB III

METODOLOGI PENELITIAN

Untuk melakukan penelitian, diperlukan langkah-langkah kerja yang jelas. Oleh karena itu, akan diuraikan tentang pembuatan sistem, pengujian fungsi sistem, pengujian performa sistem serta peralatan dan bahan yang diperlukan dalam penelitian. Sistem terbagi menjadi tiga bagian yaitu *client*, *agent* dan *database server*. Selain itu, dibuat juga aplikasi *tester* untuk mempermudah dalam melaksanakan pengujian. Penelitian dilakukan secara eksperimen dengan metode kuantitatif.

III.1 Perancangan Sistem

Sistem yang dibuat terdiri dari program *agent*, program *client* dan *database server*. Masing-masing program memberikan *input* dan memberikan respon sesuai layanan yang dibutuhkan. Pembuatan sistem dilakukan dengan langkah-langkah sebagai berikut.

III.1.1 Merancang *use case* diagram

Perancangan *use case* diagram sistem dijelaskan sebagai berikut.

1. Dibuat tiga komponen/pelaku sistem yaitu program *agent*, program *client* dan *database server* sesuai dengan konsep *multi-tier* dengan mengganti *server* dengan *agent*. Tidak seperti *server*, *agent* tidak bertindak secara otonom sehingga *agent* juga melakukan aktivitas tanpa keterlibatan *user*.
2. Fungsi-fungsi yang dilakukan oleh program *agent* dan program *client* yaitu data *forwarding*, *request* data, mengirim pesan ke *client* lain di *agent* lain, *broadcast agent* id, *broadcast client* id dan *chat*. Data *forwarding* bertujuan

Pembuatan Sistem ..., Nikolaus Very Permana, FTI UMN, 2012

agar *agent* yang menerima data dari *client* mampu mengirimkan data kepada semua *client* di jaringannya. *Request* data adalah aktivitas permintaan data dari *database server* oleh *client*. *Broadcast agent id* dan *broadcast client id* bertujuan agar antar *client* dan *agent* saling mengenal. Fungsi *chat* bertujuan agar antar *client* dalam satu jaringan dapat bertukar pesan tanpa melalui *agent*.

Sesuai dengan kebutuhan sistem, pengaksesan *database* oleh program *agent* dilakukan dengan dua fungsi standar yaitu *read database* dan *insert* ke *database*. *Use case* diagram dibuat berdasarkan interaksi antara *agent* dan *client* dalam sistem. Oleh karena itu, interaksi program *agent* dan program *client* yang dirancang merupakan kriteria keberhasilan fungsi-fungsi *agent*. Dari keberhasilan sistem, dapat ditentukan kebenaran/validitas sistem.

III.1.2 Membuat *class* diagram

Class diagram dibuat berdasarkan *use case* diagram yang telah dibuat. Ada dua *class* diagram yang dibuat yaitu *class* diagram program *agent* dan *class* diagram program *client*. Perancangan *class* diagram dijelaskan sebagai berikut.

1. Pada program *agent*, dibuat 5 *class* yaitu Agent, ChatHandler, DBConnector, FileDataServer dan FileByteServer. *Class* Agent merupakan *class* utama yang bertugas menjalankan fungsi-fungsi inisialisasi dan manajemen *thread*. *Class* ChatHandler bertugas menangani data pesan dari *client* dan kemudian meneruskan ke *client* lain yang terhubung dengannya. *Class* DBConnector bertugas meneruskan *request* data dari *client* ke *database server*. *Class* FileDataServer dasn FileByteServer adalah dua *class* yang meneruskan data

file yang dikirim oleh *client*. FileDataServer menangani informasi *file* (nama dan besar *file*), sedangkan FileByteServer menangani *byte file* yang dikirim.

2. Pada program *client*, dibuat 7 *class* yaitu Client, Form1, Aroom, RDBForm, MsgRcvr, Chat dan Chat_init. Selain *class* Client, *class-class* lain merupakan *windows form*/tampilan windows. *Class* Client bertugas melakukan fungsi-fungsi inisialisasi dan manajemen *thread*. Form1 memiliki tiga fungsi yaitu *login*, menampilkan daftar *agent* dan menampilkan daftar *client* dalam satu jaringan. *Form* Aroom melakukan fungsi-fungsi yang berhubungan dengan program *agent* yaitu data *forwarding* dan *request* data. Fungsi *request* data yang diinisialisasi form Aroom dilanjutkan RDBForm. RDBForm bertugas untuk menampilkan data seluruh *agent*, data *client* dan menyampaikan pesan ke *client* di *agent* yang berbeda. Pesan akan diterima oleh *client* dan ditampilkan dalam form MsgRcvr. Data *forwarding* terbagi menjadi *message forwarding* dan *file forwarding*. *Form* Chat dan Chat_init adalah dua *form* yang memungkinkan *client* melakukan komunikasi pesan. *Form* Chat_init dijalankan oleh *client* yang memulai *chat*, sedangkan *form* Chat dijalankan oleh *client* tujuan *chat*.

III.1.3 Membuat *flowchart* proses

Flowchart proses merupakan penggambaran cara berpikir program secara detail. Ada dua *flowchart* proses yang dibuat yaitu *flowchart* proses *agent* dan *flowchart* proses *client*. *Flowchart* proses *agent* dibuat berdasarkan *class* diagram *agent*. *Flowchart* proses *client* dibuat berdasarkan *class* diagram *client*. Dari *class* diagram yang dibuat, dapat disusun urutan fungsi berdasarkan proses yang dijalankan. Misalnya, fungsi broadcast dijalankan setelah memiliki IP komputer. *Flowchart* proses dipilah

Pembuatan Sistem ..., Nikolaus Very Permana, FTI UMN, 2012

berdasarkan aktivitas yang diperlukan masing-masing program untuk memenuhi kebutuhan yang tercantum pada *use case* diagram.

III.2 Implementasi *Database*

Sebelum *database* dibuat, data yang perlu disimpan perlu diketahui terlebih dahulu. Setelah mengetahui data yang perlu disimpan, dibuat *Entity Relationship Diagram* (ERD) untuk membagi dan menyederhanakan data. Diagram ERD membagi *database* menjadi tiga tabel yaitu tabel agent, client dan message. Tiga tabel dan hubungannya dijelaskan sebagai berikut.

1. Tabel agent berisi nama *agent* dan IP *agent* yang berjalan. Program *agent* akan mengirimkan nama dan IP-nya ketika dinyalakan dan akan menghapus ketika dimatikan.

2. Tabel client berisi nama *client*, IP *client* dan nama *agent* yang terhubung dengan *client*. Setiap ada *client* terhubung dengan *agent*, *agent* akan mengirimkan data *client* tersebut ke *database*. Data akan dihapus ketika *client* keluar dari aplikasi *agent*.

3. Tabel message berisi nama *agent* tujuan, nama *client* tujuan, IP *client* tujuan, nama *agent* pengirim, nama *client* pengirim, IP *client* pengirim, isi *pesan* dan status pembacaan.

4. Tabel agent terhubung satu arah dengan tabel client dan message karena tabel client diakses berdasarkan nama *agent* dari *client* tersebut dan tabel message diakses berdasarkan nama *agent* tujuan.

5. Tabel client dan tabel message tidak berhubungan karena tabel message diakses berdasarkan nama *agent*, bukan nama *client*.

Setelah isi tabel dan hubungan tabel jelas, dilanjutkan dengan implementasi *database server*. Dilakukan instalasi DBMS MySQL pada *device server*. Setelah *database server* dibuat, dapat dilakukan instalasi MySQL *connector* pada komputer. MySQL *connector* diperlukan karena memiliki *library-library* yang digunakan untuk mengakses DBMS MySQL. Secara praktis, MySQL connector ini hanya diperlukan saat pengembangan aplikasi karena setelah aplikasi selesai dibangun, *library-library* yang digunakan dapat di-*copy* pada *folder* lokal.

III.3 Pembuatan Program

Pembuatan program dilakukan dengan menerjemahkan *flowchart* proses yang telah dibuat ke dalam kode sumber. Program *agent* dibuat sebagai program *console* sedangkan program *client* dibuat sebagai *windows application*.

Program *agent* dan program *client* memiliki beberapa layanan yang harus dijalankan secara *multi-thread*. *Thread* harus dimatikan secara tuntas karena jika ada *thread* tersisa, dapat terjadi kesalahan/*error* ketika mengeksekusi program di lain waktu. Untuk melakukannya, digunakan metode *interrupt* dan *abort*. Meskipun demikian, jika dalam *thread* tersebut terdapat fungsi komunikasi, maka fungsi tersebut perlu dimatikan terlebih dahulu. Fungsi komunikasi tersebut dapat berupa *networkstream*, *tcplistener* atau pun *tcpclient*.

Dalam program *agent*, karakteristik otonomi *agent* dilihat pada proses *broadcast agent id*. Dalam proses tersebut, program *agent* menyebarkan informasi yang diperlukan untuk membangun koneksi. *Agent* id dikirimkan secara *broadcast* setiap 500 ms sehingga *client* mengenali keberadaan *agent* di jaringannya. Proses tersebut dijalankan oleh program *agent* tanpa keterlibatan *user* dan memiliki sistem kontrol terhadap aktivitas pengiriman tersebut.

Dalam proses pengiriman *file*, *file* diubah menjadi *byte* sebelum dikirim. Selain *byte* dari *file* tersebut, diperlukan nama dan panjang *file*. Nama *file* diperlukan dalam proses penyimpanan dan penentuan alamat *file*. Panjang *file* diperlukan untuk melakukan rekonstruksi *file* dari *byte*.

III.4 Pengujian Sistem

Pengujian fungsi sistem yang dilakukan meliputi uji fungsional dan uji performa sistem. Uji fungsional dilakukan dengan membandingkan antara rancangan aplikasi dengan hasil aplikasi. Setelah sesuai dengan rancangan, dapat dilakukan uji performa. Uji performa harus dilakukan karena selain sesuai dengan rancangan, aplikasi harus mempunyai performa yang terukur. Dari pengukuran tersebut, kehandalan sistem dapat ditentukan.

III.4.1 Uji fungsional sistem

Untuk melakukan uji fungsional ini, telah dirangkum proses-proses yang harus dapat dijalankan oleh aplikasi. Proses-proses tersebut terdaftar sebagai berikut.

1. Proses inisialisasi koneksi *client* dengan *agent*.
2. Proses pengenalan *client* ke *client* lain.
3. Proses pengiriman data dari *client* ke *agent* untuk diteruskan.
4. Proses permintaan daftar *agent* dan daftar *client* dari *client* ke *server*.
5. Proses pengiriman data dari *client* ke *client* di *agent* lain.

Dari hasil/*output* aplikasi, dilakukan pencocokan terhadap keberhasilan proses-proses di atas. Jika proses-proses di atas berhasil dilakukan, maka program *agent* dan *client* dapat dinyatakan sesuai dengan sistem yang dirancang.

Pembuatan Sistem ..., Nikolaus Very Permana, FTI UMN, 2012

III.4.2 Uji performa sistem

Untuk memastikan bahwa program handal dalam mentransmisikan data, dilakukan pengujian terhadap tiga variabel yang berhubungan dengan pengiriman data yaitu *response time*, konsistensi data dan skalabilitas. Pengujian dilakukan dengan menggunakan program *tester*. Program *tester* dibangun dari fungsi-fungsi yang dimiliki program *client*. Program *tester* mampu menerima *broadcast* id dari program *agent*, membangun koneksi dan melakukan komunikasi data selayaknya program *client* berhubungan dengan program *agent*. Program *tester* memiliki fitur-fitur tambahan untuk melakukan dan memudahkan analisa yaitu mengambil data yang akan dikirim dari *file*, *stopwatch*, penyimpanan *log* data dan pengecekan data. Melalui program *tester*, pengujian *response time*, konsistensi data dan skalabilitas dilakukan secara bersamaan namun tercatat dalam *log* data yang berbeda. Pengujian dilakukan sebanyak 100 kali. Pengujian tiga variabel tersebut terhadap sistem dijelaskan sebagai berikut.

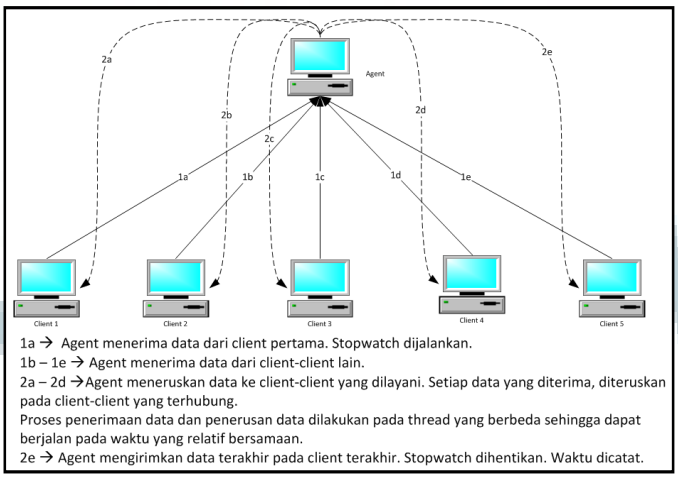
III.4.2.1 *Response time*

Uji *response time* dilakukan melalui pengiriman data secara bersamaan (*concurrent*). Pengiriman secara bersamaan dimaksudkan untuk mengamati performa sistem jika diberi beban yang tinggi. Beban yang tinggi terjadi jika jumlah *request* yang diberikan sama atau lebih tinggi daripada kemampuan sistem. Pengukuran dilakukan terhadap kemampuan program *agent* untuk melakukan pengiriman data kepada sejumlah *client*. Pengujian *response time* secara bersamaan dilakukan sebagai berikut.

1. Menggunakan program *tester* untuk menampung isi data teks dari *file* berukuran 1 MB yang diperoleh melalui FileDialog. Data ini digunakan juga pada pengujian konsistensi data. Program *tester* membentuk

koneksi dengan *agent*. Jumlah koneksi yang diuji bervariasi yaitu 5, 10, 15 dan 20 komputer. Setiap koneksi yang terbentuk akan menambah nilai *counter* pada *agent* dengan nilai satu. Nilai *counter* ini dikirimkan secara terus-menerus sampai nilainya sama dengan jumlah koneksi yang diinginkan. Setelah itu, data dikirim dari masing-masing koneksi. Dengan metode tersebut pengiriman data dilakukan pada waktu yang relatif bersamaan.

2. Program *agent* akan mencatat waktu yang dibutuhkan oleh program *agent*. Untuk memudahkan proses pengukuran tersebut digunakan fitur *stopwatch* dari .NET yang ditambahkan pada program *agent*. *Stopwatch* tersebut akan dimulai ketika *agent* menerima data pertama kali dan diakhiri setelah selesai mengirim data ke *client* terakhir yang terkoneksi. Pengiriman data dilakukan dengan metode write dari *class* NetworkStream. Metode write mencakup proses pengiriman *buffer* melewati jaringan sampai diperoleh *acknowledgement*/ACK [31]. Pencatatan waktu akan disimpan oleh *agent* dalam *log* data. Mekanisme secara jelas dapat diamati pada gambar 3.1 berikut.



Gambar 3. 1 Mekanisme pengujian *response time*

Hasil yang diperoleh dari uji *response time* ini adalah data waktu yang diperlukan oleh program *agent* untuk menerima sampai dengan selesai mengirim data dalam berbagai ukuran data dan jumlah koneksi. Dari 100 kali pengujian, akan didapatkan rata-rata *response time* pengiriman data.

Untuk menguji bahwa pengiriman data *file* juga berjalan, dilakukan pengujian *response time* dengan metode yang sama dengan data berupa *file* berekstensi mp3, jpg dan flv. Pengujian tersebut dilakukan 5 kali untuk masing-masing jenis *file*. *File* yang digunakan berukuran kurang lebih 1 MB.

III.4.2.2 Konsistensi data

Uji konsistensi data berfokus pada data yang diterima oleh *agent* dari program *tester* dan dikirimkan kepada *client-client* yang ditanganinya. Data dianggap benar jika data yang dikirim dan diterima adalah sama. Langkah-langkahnya adalah sebagai berikut.

1. Menggunakan program *tester* yang sama untuk pengukuran *response time*. Data yang dikirimkan disimpan dalam variabel sementara. Program *tester* juga menerima data dari program *agent*.
2. Data yang dikirimkan dikirimkan dibandingkan dengan data yang diterima. Data yang dikirimkan dicatat dalam *file* teks “LogData” oleh *agent*. Hasil perbandingan disimpan dalam *file* teks “LogCorrectness” oleh program *tester*.

Hasil yang diperoleh dari uji konsistensi data ini adalah benar jika data yang diterima sama dengan data yang dikirim dan sebaliknya. Dari 100 kali pengujian, setiap data benar akan meningkatkan persentase konsistensi data sebesar 1%. Dengan benar sebanyak 100 kali, akan didapat tingkat konsistensi data sebesar 100%.

III.4.2.3 Skalabilitas

Skalabilitas adalah kemampuan sistem, jaringan atau proses untuk menangani jumlah pekerjaan yang meningkat dalam suatu batas kemampuan atau kemampuan untuk diperluas sehingga mampu menangani perkembangan tersebut [17]. Untuk penggunaan di 100 pengguna, diperlukan sistem tambahan atau performa akan menurun.

Uji skalabilitas untuk suatu sistem terdistribusi berfokus pada skalabilitas beban. Skalabilitas beban adalah kemampuan sistem terdistribusi untuk menampung sejumlah *input* yang diberikan baik lebih besar atau lebih kecil dari yang disediakan oleh sistem [23].

Uji skalabilitas dilakukan melalui langkah-langkah sebagai berikut.

1. Menggunakan hasil yang diperoleh dari uji konsistensi data untuk dihitung jumlah data cocok dan tidak cocok.
2. Jika hasil uji konsistensi data seluruhnya benar maka sistem dinyatakan mampu menangani pengiriman data secara bersamaan dengan jumlah tersebut.

Hasil yang diperoleh dari uji skalabilitas ini adalah kesimpulan apakah sistem dapat menangani beban dalam berbagai ukuran data dan jumlah koneksi pada saat kondisi *heavy load buffer*.

III.5 Peralatan dan Bahan

1. *Personal Computer*

   a. Prosesor : Intel® Core 2 Duo

   b. Memori : 1GB DDR2 SDRAM

   c. Sistem operasi : Microsoft® Windows® XP® Professional

2. Jaringan komputer berkabel lokal dengan topologi fisik star.
3. *Database* MySQL untuk membangun *database server.*
4. Aplikasi Microsoft® Visual Studio® untuk membangun program *agent* dan *client*.
5. Aplikasi MySQL *connector* untuk melengkapi *library-library* pengaksesan DBMS MySQL yang belum tersedia dalam Microsoft® Visual Studio®.
6. Aplikasi XAMPP sebagai *bundle package* untuk instalasi DBMS MySQL.