IMPLEMENTASI CORBA PADA PEMROGRAMAN LINTAS PLATFORM JAVA DAN DELPHI UNTUK MEMBANGUN APLIKASI SISTEM TERDISTRIBUSI

Joko Margono[*)], Maman Somantri, and Kodrat I.S

Jurusan Teknik Elektro, Universitas Diponegoro Semarang
Jl. Prof Sudharto, SH. Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: joko.margono87@gmail.com*

Abstrak

Sistem terdistribusi adalah sistem di mana pemrosesan informasi didistribusikan pada beberapa komputer dan tidak terbatas hanya pada satu mesin saja. Heterogenitas dalam sebuah sistem terdistribusi sangat mungkin terjadi, baik heterogenitas dalam hal perangkat keras maupun perangkat lunak. Heterogenitas perangkat lunak tersebut dapat dikarenakan aplikasi perangkat lunak dalam sistem terdistribusi dikembangkan menggunakan platform bahasa pemrograman yang berbeda. Pada penelitian ini penulis mengimplementasikan sebuah teknologi middleware ORB yang berada dalam framework CORBA untuk mengatasi heterogenitas perangkat lunak yang dibangun menggunakan platform bahasa pemrograman yang berbeda, khususnya platform bahasa pemrograman Java dan Delphi. Sistem yang dibangun dalam implementasi ini adalah sebuah aplikasi informasi akademik terdistribusi. Hasil implementasi dan pengujian membuktikan bahwa teknologi CORBA telah berhasil diterapkan pada sebuah aplikasi informasi akademik terdistribusi yang dibangun dengan bahasa pemrograman Java dan Borland Delphi 7. Sistem terdistribusi informasi akademik ini dapat melayani pengguna melalui layanan web, layanan pesan singkat, dan juga melalui aplikasi klien desktop. Masing-masing aplikasi tersebut ditempatkan pada komputer yang berbeda dan alamat jaringan yang berbeda pula, namun dapat saling berkomunikasi melalui pemanggilan objek di lingkungan CORBA.

Kata kunci: CORBA, middleware, objek, platform bahasa pemrograman, sistem terdistribusi

Abstract

Distributed system is a system, which information processing is distributed on several computers and not limited to just one machine only. Heterogeneity in a distributed system is very likely to occur, both heterogeneity in terms of hardware and software. Heterogeneity of software applications may be due to applications in the distributed system developed using different programming languages. In this research is implementing an ORB middleware technologies that are in the CORBA framework to overcome the heterogeneity of software developed using different platform programming languages, especially platform programming language of Java and Delphi. The system developed in this implementation is a distributed application of academic information. Implementation and testing results prove that the CORBA technology has been successfully applied to a distributed academic information application, that was developed using the programming language Java and Borland Delphi 7. Distributed system of academic information application can serve users through web services, short messaging services, and also through desktop client application. Each application is placed on different computer and different network addresses, but can communicate with each other through the invoking object in the CORBA environment.

Key word: CORBA, distributed system, middleware, object, platform programming language

1. Pendahuluan

Perkembangan teknologi pemrograman yang begitu luas sehingga menciptakan banyak bahasa pemrograman yang berbeda. Setiap *platform* bahasa pemrograman memiliki standar dan aturan tersendiri untuk membangun sebuah aplikasi program komputer. Idealnya diharapkan bahwa berbagai aplikasi yang dibangun dengan bahasa pemrograman berbeda tersebut tetap dapat saling berkomunikasi satu sama lain dalam sebuah sistem terdistribusi.

Pada saat ini hampir semua sistem berbasis komputer yang besar merupakan sistem terdistribusi. Sistem

terdistribusi adalah sistem dimana pemrosesan informasi didistribusikan pada beberapa komputer dan tidak terbatas hanya pada satu mesin saja. Salah satu karakteristik yang penting untuk sistem terdistribusi adalah keterbukaan. Keterbukaan sistem yang dimaksud disini adalah terbuka untuk banyak sistem operasi dan banyak vendor atau *platform* yang berbeda. Untuk memenuhi karakteristik tersebut maka dalam penelitian ini akan mencoba membuat sebuah aplikasi sistem terdistribusi yang menggunakan dua buah *platform* teknologi bahasa pemrograman yang berbeda.

Untuk menghubungkan beberapa aplikasi yang dibangun dengan *platform* bahasa pemrograman yang berbeda agar dapat bekerjasama membentuk sebuah sistem, salah satu cara adalah dengan memanfaatkan teknologi *middleware*. *Middleware* yang digunakan dalam penelitian ini adalah ORB (*object Request Broker*) yang ada dalam *framework* CORBA (*Common Object Request Broker Architecture*).

Tujuan dari penelitian ini adalah melakukan implementasi teknologi *middleware* CORBA-ORB untuk menghubungkan komunikasi antar objek dari aplikasi yang dibangun menggunakan *platform* bahasa pemrograman yang berbeda, yaitu Java dan Delphi 7. Bentuk implementasi dari teknologi CORBA ini adalah sebuah sistem terdistribusi sederhana yang berupa aplikasi informasi akademik terdistribusi yang dapat diakses pengguna melalui layanan *web*, layanan pesan singkat, dan aplikasi klien desktop.

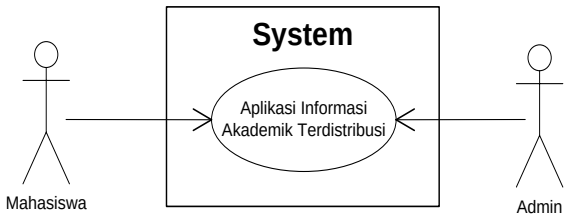
2. Metode

2.1. Perancangan Sistem Berdasarkan Pemodelan

Pada perancangan sistem terdistribusi dari aplikasi informasi akademik yang menerapkan teknologi CORBA ini, digunakan bahasa pemodelan *Unified Modeling Language* (UML).

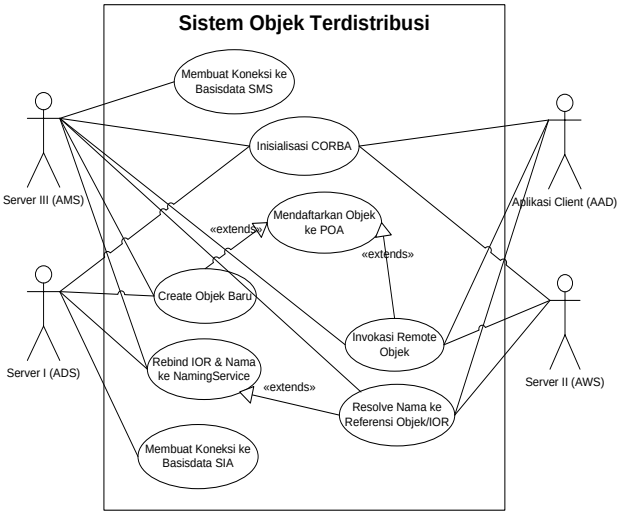
2.1.1. Perancangan Diagram *Use Case*

Sebuah *use case* merepresentasikan interaksi antara aktor dengan sistem. Seperti yang ditunjukkan pada Gambar 1, *use case* aplikasi informasi akademik menggambarkan pengguna yang melakukan akses ke sistem untuk meminta layanan informasi akademik yang ada dalam basisdata, baik melalui layanan *web* maupun layanan pesan singkat. Dan aktor lainnya sebagai admin yang berperan untuk memasukkan data informasi akademik ke dalam basisdata sistem melalui aplikasi yang dibuat khusus untuk admin.



Gambar 1 *Use Case* pada Aplikasi Informasi Akademik

Pada perancangan sistem terdistribusi ini, selain suatu sistem yang terlihat oleh pengguna sebagai sebuah aplikasi informasi akademik. Namun di dalamnya terdiri dari beberapa server yang saling berkomunikasi dan bekerja sama untuk dapat melayani permintaan informasi akademik yang dibutuhkan pengguna. Diagram *use case* yang menggambarkan interaksi antar bagian dari sistem terdistribusi ini ditunjukkan pada Gambar 2 berikut.



Gambar 2 Diagram *use case* interaksi antar bagian atau aplikasi dalam sistem terdistribusi

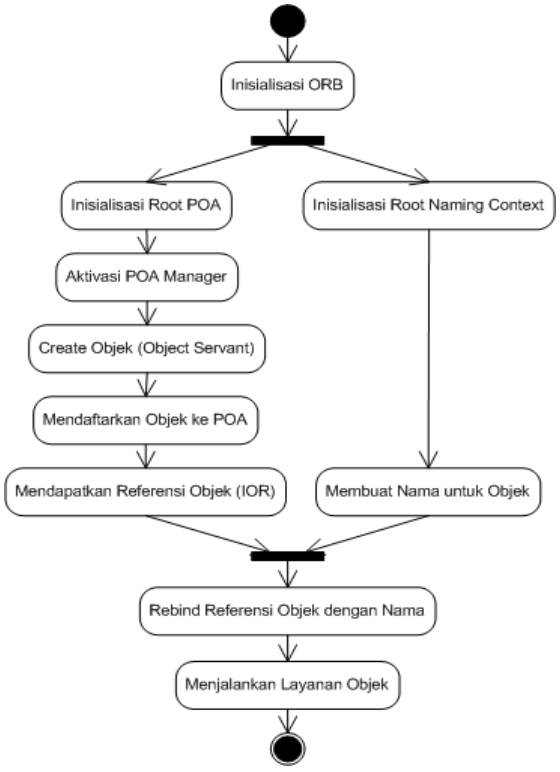
Gambar 2 memperlihatkan hal-hal apa saja yang dilakukan oleh masing-masing server atau aplikasi dalam sistem terdistribusi yang akan dibangun ini. Server I ADS (*Academic Database Server*) dan Server III AMS (*Academic Message Server*) akan melakukan pembuatan objek baru sesuai objek implementasi yang telah ditentukan pada masing-masing server tersebut, termasuk mendaftarkan objek tersebut ke *object adapter* (POA/ BOA) dan mendapatkan referensi objek. Kemudian server tersebut akan melakukan *re-binding* atau memasangkan referensi objek dengan "nama" dari masing-masing objek tersebut ke layanan penamaan (*Naming Service*). Untuk Server III ini, selain memberikan layanan objek, server ini juga menggunakan objek-objek yang disediakan oleh Server I. Untuk Server II AWS (*Academic Web Server*) dan aplikasi klien desktop admin (AAD) akan menggunakan objek-objek yang disediakan oleh dua

server yang lainnya itu untuk mendapatkan informasi sesuai apa yang dibutuhkan oleh pengguna.

2.1.2. Perancangan Diagram Aktivitas

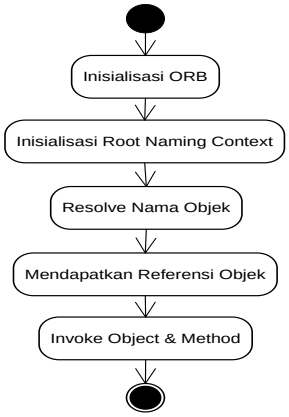
Diagram aktivitas menggambarkan logika prosedural dan aliran kerja dalam sistem yang sedang dirancang. Berikut ini diagram aktivitas yang menunjukkan proses aktivitas pada server penyedia layanan objek dan aktivitas pada klien yang mengakses objek *remote*.

Pada Gambar 3 berikut, menunjukkan diagram aktivitas pada server penyedia layanan objek. Aktivitasnya diawali dari inisialisasi ORB sebagai servis utama untuk layanan CORBA, kemudian inisialisasi *object adapter* (POA) dan aktivasi manager POA. Selanjutnya membuat atau instansiasi objek-objek implementasi (*object servant*) yang nantinya objek tersebut dapat dipanggil dari aplikasi klien atau sistem yang lain. Objek tersebut kemudian didaftarkan ke *object adapter* dan mendapatkan referensi objek yang dikenal dengan IOR (*Interoperable Object reference*). Pada sisi lain, perlu menyiapkan untuk layanan penamaan atas objek-objek yang telah dibuat tersebut. Pada layanan penamaan diawali dengan inisialisasi *Root Naming Context* untuk mendapatkan *root* konteks penamaan. Kemudian membuat nama untuk objek dan tentunya nama tersebut dimasukkan dalam daftar konteks penamaan. Dan akhirnya referensi objek tersebut diikat/ *rebind* dengan nama ke layanan penamaan.



Gambar 3 Aktivitas pada server penyedia layanan objek

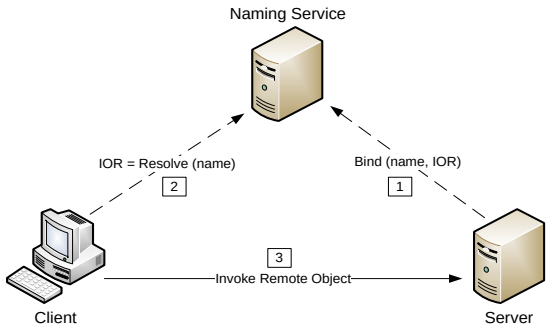
Sedangkan pada Gambar 4 berikut menunjukkan diagram aktivitas yang menggambarkan urutan aktivitas pada klien yang mengakses objek *remote*. Klien merupakan aplikasi atau sistem yang menggunakan objek *remote* yang disediakan oleh server penyedia layanan objek. Aktivitas pada klien diawali juga dengan inisialisasi ORB untuk mengetahui lokasi layanan ORB-CORBA berjalan. Kemudian inisialisasi *Root Naming Context* untuk mendapatkan konteks penamaan awal. Dan selanjutnya melakukan *resolve* atas nama objek yang akan dipanggil ke layanan penamaan untuk mendapatkan referensi objek yang dimaksud. Akhirnya dengan referensi objek tersebut klien dapat melakukan invokasi ke objek *remote* dan dapat menggunakan metode dari objek tersebut.



Gambar 4 Diagram aktivitas pada klien yang mengakses remote objek

2.2. Gambaran Umum Sistem dengan Teknologi CORBA

Teknologi CORBA merupakan salah satu solusi untuk pengembangan aplikasi sistem terdistribusi, karena memungkinkan pemanggilan atau komunikasi antar objek dari berbagai aplikasi yang ada dalam sebuah sistem yang mampu menangani heterogenitas aplikasi dalam sistem terdistribusi tersebut. Gambaran umum untuk aplikasi dengan teknologi CORBA ini dapat dideskripsikan bahwa sebuah program server penyedia layanan objek dapat membuat objek-objek implementasi yang didaftarkan dan dikelola oleh *obejct adaper*. Informasi mengenai objek tersebut yang telah terdaftar pada *object adapter* memiliki identitas yang disebut dengan *object reference* atau referensi objek. Referensi objek tersebut dapat diikat atau di-*rebind* dengan sebuah ‘nama’ ke layanan penamaan yang merupakan bagian dari *CORBA Services.*

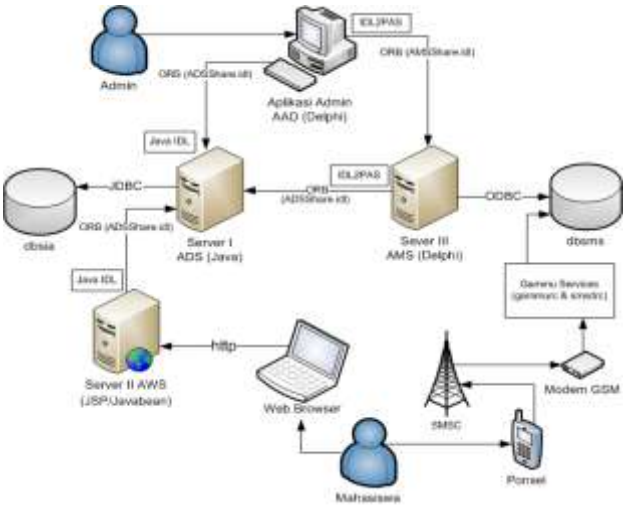


Gambar 5 Gambaran umum arsitektur pemanggilan objek pada implementasi CORBA

Pada Gambar 5 di atas, menunjukkan mekanisme klien untuk dapat memanggil objek *remote* dapat dijelaskan sebagai berikut. Program server mendaftarkan referensi objek dari objek-objek *servant* yang dimilikinya ke layanan penamaan. Program klien harus mengetahui nama-nama objek pada server yang didaftarkan pada layanan penamaan. Dengan nama objek yang telah diketahui oleh klien tersebut, kemudian dapat di-*resolve* atau didapatkan referensi objek yang memiliki informasi identitas dan keberadaan dari objek yang akan dipanggil tersebut. Setelah referensi objek didapatkan, klien dapat langsung melakukan invokasi ke objek pada server tersebut.

2.3. Perancangan Topologi sistem

Rancangan topologi sistem ini menggambarkan arsitektur dari sistem yang akan dibangun yang menjelaskan hubungan geometris antara unsur-unsur dasar penyusun sistem tersebut, yaitu server aplikasi, basisdata, PC, dll. Gambar rancangan topologi sistem dari implementasi CORBA pada aplikasi informasi akademik terdistribusi ini dapat diperlihatkan pada Gambar 6 berikut.



Gambar 6 Rancangan topologi sistem aplikasi informasi akademik dengan teknologi CORBA

Pada Gambar 6 menunjukkan topologi sistem dari rancangan arsitektur sistem yang diusulkan. Server I ADS menyediakan layanan objek-objek untuk dapat mengakses informasi akademik yang tersimpan dalam basis data ‘dbsia’. Server II AWS sebagai penyedia layanan *web* yang dapat diakses oleh mahasiswa melalui aplikasi perambah *web* dan AWS ini sekaligus sebagai klien yang menggunakan objek-objek pada server ADS. Server III AMS sebagai penyedia layanan *SMS Autoresponder* untuk akses informasi akademik melalui pesan singkat. Dan server AMS ini juga menyediakan objek yang dapat digunakan untuk akses informasi layanan SMS yang tersimpan dalam basis data ‘dbsms’. Aplikasi desktop untuk admin (AAD) digunakan admin untuk melakukan entri data ke basis data ‘dbsia’ dan juga untuk monitoring layanan SMS yang tersimpan dalam basis data ‘dbsms’. Sehingga aplikasi AAD ini akan menggunakan objek yang disediakan oleh server ADS maupun objek yang disediakan oleh server AMS.

3. Hasil dan Analisa

3.1. Implementasi Sistem

Langkah-langkah membuat aplikasi sistem terdistribusi dengan menerapkan teknologi CORBA adalah sebagai berikut:

3.1.1. Definisi *interface* objek dengan bahasa IDL

Langkah pertama untuk membuat aplikasi dengan teknologi CORBA adalah menentukan objek dan *interface*-nya menggunakan bahasa IDL. Pendefinisian objek dengan bahasa IDL ini berfungsi untuk menggambarkan sebuah antarmuka dalam bahasa yang independen sehingga memungkinkan komunikasi antara objek yang berbeda bahasa pemrogramannya. IDL ini dapat digunakan untuk mendefinisikan modul, *interface*, struktur data, dll. Definisi *interface* ini juga menunjukkan nilai masukkan dan keluaran dari suatu metode pada objek tertentu.

3.1.2. Generate berkas IDL ke kode bahasa pemrograman yang digunakan

Berkas IDL yang telah dibuat tersebut perlu di-*generate* atau dikompilasi ke bahasa pemrograman yang digunakan (contoh: Java, Delphi/objek Pascal, dll) agar dapat digunakan pada kelas implementasi di sisi server dan juga sebagai *interface* di sisi server maupun klien.

Pada penelitian ini *generate* berkas IDL ke Java menggunakan kompilator ‘idlj’ yang menggunakan pemetaan IDL ke Java untuk mengkonversi definisi *interface* IDL ke *interface*, kelas, dan metode yang sesuai dengan Java. Dan untuk *generate* IDL ke kode Pascal menggunakan kompilator ‘Idl2pas’.

3.1.3. Membuat kelas implementasi objek

Kelas implementasi objek atau disebut *servant* adalah kelas-kelas yang mengimplementasikan *interface* suatu objek yang telah didefinisikan dalam IDL. Kelas implementasi objek ini dibangun di sisi server.

3.1.4. Membuat kelas utama untuk menjalankan layanan objek di sisi server

Pada kelas utama ini berisi kode program yang mengimplementasikan tahap-tahap dalam menjalankan server penyedia layanan objek dan dapat dijelaskan sebagai berikut.

a. Melakukan inisialisasi layanan CORBA – ORB;
b. Mendapatkan referensi *Root POA* dan aktivasi *POA Manager*;
c. Instansiasi objek *servant*, yaitu melakukan instansiasi kelas implementasi yang telah dibuat ke masing-masing objek *servant*;
d. Mendapatkan referensi objek yang diasosiasikan dengan *servant*;
e. Mendapatkan inisial *Naming Context*, untuk mendapatkan referensi objek ke server penamaan;
f. Mendaftarkan objek *servant* ke server penamaan (*Name Server*);
g. Menunggu invokasi objek dari klien.

3.1.5. Membuat program klien

Pada program klien ini berisi kode program yang mengimplementasikan tahap-tahap dalam melakukan pemanggilan atau invokasi objek *remote* ke server.

a. Melakukan inisialisasi layanan CORBA – ORB;
b. Mendapatkan inisial *Naming Context*, untuk mendapatkan referensi objek ke server penamaan;
c. Melakukan *resolve* suatu 'nama' dari objek yang akan dipanggil untuk mendapatkan referensi objek;
d. Melakukan invokasi atau pemanggilan objek dan metode ke objek *remote* yang dimaksud tersebut.

3.2. Pengujian Sistem

3.2.1. Spesifikasi komputer

Pada pengujian kali ini menggunakan sebuah komputer desktop dan sebuah laptop yang mewakili beberapa server yang digunakan, serta aplikasi pengguna. Komputer desktop digunakan sebagai server I ADS termasuk di dalamnya basis data 'dbsia'. Dan komputer desktop ini ditambahkan pula mesin virtual (VirtualBox) yang digunakan sebagai server II AWS dan Router 1 (Mikrotik). Spesifikasi dari komputer Desktop yang digunakan dalam pengujian adalah sebagai berikut.

a. Prosesor Intel Core 2 Duo CPU E7500 @ 2,93 GHz
b. RAM 2,00 GB
c. Harddisk 80 GB

Komputer laptop digunakan sebagai server III AMS termasuk di dalamnya basis data 'dbsms'. Dan komputer laptop ini ditambahkan pula mesin virtual (VirtualBox) yang digunakan sebagai Router 2 (Mikrotik) dan juga sebagai komputer pengguna untuk menjalankan aplikasi perambah *web* bagi pengguna mahasiswa dan aplikasi desktop AAD bagi admin. Spesifikasi dari komputer laptop yang digunakan dalam pengujian adalah sebagai berikut.

a. Prosesor Intel Core 2 Duo CPU T5870 @ 2,00 GHz
b. RAM 2,00 GB
c. Hardisk 250 GB

3.2.2. Spesifikasi Modem

Pada pengujian ini menggunakan sebuah modem GSM untuk menerima dan mengirim pesan SMS informasi akademik kepada pengguna dengan spesifikasi sebagai berikut.

a. USB untuk jaringan GSM
b. Mendukung layanan SMS
c. Mendukung kartu SIM GSM
d. *Interface* ke PC dengan USB 2.0
e. Kompatibel dengan OS Windows 2000/XP/Vista/7

3.2.3. Kebutuhan perangkat lunak

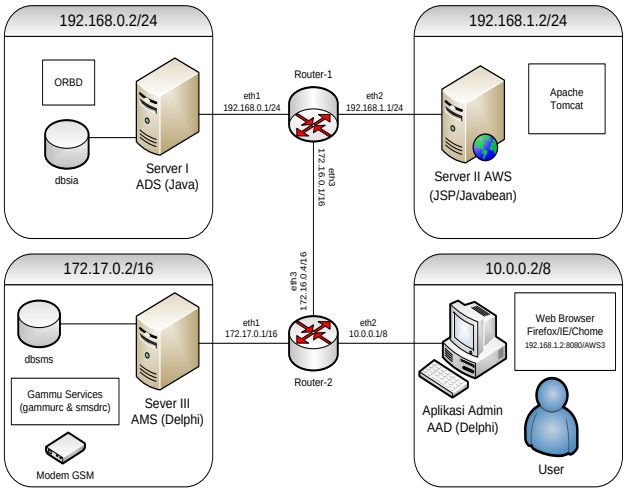
Kebutuhan perangkat lunak untuk pengujian sistem pada implementasi teknologi CORBA dalam penelitian ini adalah sebagai berikut.

a. Windows 7 Ultimate (OS)
b. XAMPP Windows 1.8.1 (termasuk MySQL versi 5.5.27 dan Apache Tomcat versi 7.0.30)
c. Paket Java SE Development Kit 7 (termasuk JRE 7, MySQL Connector Java, dan aplikasi ORBD)
d. Borland Delphi 7 (termasuk VisiBroker 4.5)
e. MySQL Connector ODBC versi 5.1.8-win32
f. Aplikasi *SMS Gateway* Gammu-1.32.0-Windows
g. Oracle VirtualBox-4.2.10-Win
h. Mikrotik OS versi 5.20 sebagai Router

3.2.4. Pengujian Teknologi CORBA pada Sistem Terdistribusi

Pengujian ini dilakukan menggunakan *host*, alamat, dan jaringan yang berbeda untuk setiap server maupun komputer pengguna. Pada pengujian ini aplikasi server I ADS (*Academic Database Server*) ditempatkan pada komputer desktop, termasuk basis data 'dbsia' dan aplikasi server penamaan ORBD dengan alamat IP (192.168.0.2/24). Untuk server II AWS (*Academic Web Server*) ditempatkan pada mesin virtual yang ada pada komputer desktop dengan alamat IP (192.168.1.2/24). Untuk server III AMS (*Academic Message Server*) ditempatkan pada komputer laptop, termasuk basis data 'dbsms' dan aplikasi Gammu sebagai *SMS Gateway*, serta modem GSM juga ada di komputer ini dengan alamat IP (172.17.0.2/16). Dan untuk aplikasi-aplikasi klien yaitu

perambah *web* bagi pengguna mahasiswa dan aplikasi desktop AAD bagi pengguna admin ditempatkan pada mesin virtual yang ada pada komputer laptop dengan alamat IP (10.0.0.2/8). Pada pengujian ini menggunakan jaringan yang berbeda untuk setiap server dan komputer pengguna, sehingga memerlukan *router* pada jaringan. *Router* yang digunakan merupakan Mikrotik OS yang dipasang pada mesin virtual (VirtualBox). Supaya lebih jelas dapat dilihat skema jaringan untuk pengujian ini pada Gambar 7 berikut.



Gambar 7 Skema jaringan untuk pengujian

Langkah-langkah untuk pengujian adalah dengan menjalankan aplikasi-aplikasi yang ada pada setiap server yang ada pada sistem terdistribusi dari aplikasi informasi akademik ini. Langkah pertama menjalankan aplikasi ORBD sebagai pusat server penamaan atas objek-objek yang didistribusikan dalam sistem. Untuk menjalankan ORBD dapat melalui perintah / *command windows* atau membuat berkas *batch windows* (.bat) dengan kode "`start orbd -ORBInitialHost 192.168.0.2 -ORBInitialPort 1050`". Kemudian menjalankan aplikasi pada Server I (ADS3) dengan perintah / *command windows* " `start java -cp . ADS3 -ORBInitialHost 192.168.0.2 -ORBInitialPort 1050` ". Selanjutnya menjalankan Server II AWS sebagai server *web* dengan cara menjalankan mesin Tomcate 7 di komputer dengan alamat IP (192.168.1.2). Dilanjulkan dengan menjalankan Server III AMS sebagai penyedia layanan SMS dengan perintah " `start AMS -ORBInitRef NameService= iioploc://192.168.0.2:1050/NameService` ". Untuk menjalankan aplikasi klien untuk admin (AAD) dengan perintah "`start AAD -ORBInitRef NameService =iioploc://192.168.0.2:1050/NameService`". Untuk pengguna mahasiswa melalui aplikasi perambah *web* dengan menuju ke url "`http://192.168.1.2/AWS3/`" dan dapat melalui layanan SMS ke nomor yang ada pada modem server AMS.

Berdasar hasil pengujian sistem terdistribusi yang dibangun yaitu untuk aplikasi informasi akademik dengan skema jaringan seperti ditunjukkan Gambar 7 di atas, bahwa Server I ADS dapat dijalankan dengan baik di komputer dengan IP (192.168.0.2/24) tersebut bersama dengan server penamaan ORBD. Server II AWS dapat dijalankan dengan baik menggunakan Tomcat 7 di komputer dengan IP (192.168.1.2/24). Server III AMS dapat dijalankan dengan baik di komputer dengan IP (172.17.0.2/16) bersamaan dengan Gammu servis sebagai *SMS Gateway* yang digunakannya. Dan server AMS ini telah berhasil melayani permintaan informasi akademik melalui layanan pesan singkat. Aplikasi admin AAD telah dapat dijalankan di komputer dengan IP (10.0.0.2/8) dan dapat melakukan entri dan penyimpanan data akademik ke basis data 'dbsia' yang ada pada server I melalui proses invokasi objek yang disediakan server ADS tersebut. Dan aplikasi AAD ini juga dapat mengakases informasi layanan SMS dalam basis data 'dbsms' melalui invokasi objek yang disediakan server AMS. Aplikasi perambah *web* juga dapat mengakses informasi akademik melalui layanan *web* ke server AWS dengan url "http://192.168.1.2:8080/AWS3/" yang menunjukkan lokasi server *web* dijalankan.

3.3. Analisa Hasil Penelitian

Pada bagian akhir dari bab ini akan dijelaskan mengenai analisis dari implementasi teknologi CORBA pada pemrograman lintas *platform* bahasa pemrograman yaitu *platform* Java dan Borland Delphi 7 untuk membangun sebuah aplikasi sistem terdistribusi yang dalam penelitian ini berupa aplikasi informasi akademik terdistribusi. Sistem terdistribusi yang dimaksud dalam penelitian ini yaitu adanya pendistribusian objek-objek yang menyusun sistem dan pendistribusian fungsi pada masing-masing server di dalamnya. Teknologi CORBA sangat memungkinkan untuk objek-objek dalam sebuah sistem dapat didistribusian pada aplikasi, server, atau klien pada mesin yang sama maupun mesin yang berbeda. Pada implementasi ini mencerminkan juga adanya sebuah distribusi fungsi pada setiap bagian dari sistem terdistribusi tersebut, yaitu server ADS memiliki fungsi sebagai pengelola dan akses kontrol basisdata informasi akademik, server *web* AWS memiliki fungsi sebagai penyedia layanan *web* informasi akademik, dan server AMS memiliki fungsi sebagai penyedia layanan pesan singkat informasi akademik dan sebagai pengontrol akses ke basis data SMS.

Berdasar implementasi dan pengujian yang telah dilakukan, bahwa teknologi CORBA dapat diterapkan pada sebuah aplikasi informasi akademik terdistribusi. Pada implementasi tersebut telah berhasil membangun komunikasi antar aplikasi yang dibangun menggunakan bahasa pemrograman yang berbeda. Masing-masing aplikasi tersebut ditempatkan pada server atau mesin komputer yang berbeda, namun terhubung melalui

jaringan komputer. Penggunaan teknologi CORBA dapat menjembatani komunikasi dari aplikasi-aplikasi tersebut yang berada pada server yang berbeda, dan tetap dapat bekerjasama serta berkomunikasi melalui pengiriman pesan / *string* melalui jaringan komputer yang menghubungkannya. Sistem ini dapat berjalan karena setiap bagian dapat berfungsi dan berperan sebagaimana mestinya. Server I - ADS dapat menyediakan akses basis data informasi akademik yang dibutuhkan oleh aplikasi pada server lain. Server II – AWS dapat menyediakan layanan *web* untuk dapat memberikan halama-halaman *web* yang diperlukan pengguna. Server III – SMS dapat menyediakan layanan *SMS Autoresponder* yang dapat diakses pengguna melalui layanan pesan singkat. Dan aplikasi desktop AAD yang dapat diakses oleh pengguna admin.

Teknologi CORBA menyangkut sebuah paradigma komunikasi antar objek, baik objek yang dimiliki oleh aplikasi yang berbeda maupun objek yang berada pada mesin yang berbeda yang berada dalam sebuah lingkungan CORBA. Komunikasi antar objek tersebut tidak dibatasi oleh adanya *platform* bahasa pemrograman dan perbedaan atau heterogenitas lain yang mungkin ada dalam sebuah jaringan komputer. CORBA ini merupakan sebuah jembatan penghubung komunikasi objek antara aplikasi satu dengan yang lainnya dalam sebuah sistem terdistribusi. Jika digambarkan ke dalam arsitektur klien-server, maka aplikasi yang berperan sebagai penyedia layanan objek dapat dikatakan sebagai sisi server dan aplikasi pengguna atau yang memanggil/ invokasi atas objek yang disediakan sisi server tersebut dapat dikatakan sebagai sisi klien.

Berdasar implementasi dan pembahasan yang telah dilakukan dalam tugas akhir ini dapat menunjukkan bahwa CORBA merupakan sebuah *framework* yang dapat dijadikan pilihan untuk membangun aplikasi sistem terdistribusi. Pengembangan teknologi CORBA yang berdasar pada konsep pemrograman berorientasi objek sangat menguntungkan sekali dalam pengembangan sistem terdistribusi, hal ini karena akan memudahkan pengembang dalam memperluas dan mengembangkan sistem yang dibangun dengan konsep pemrograman ini. Konsep pemanggilan objek antar aplikasi atau komponen dalam sebuah sistem terdistribusi sangat menguntungkan, karena sebuah objek yang telah diciptakan pada sebuah aplikasi akan dapat digunakan oleh aplikasi lain yang memerlukan objek yang sama tersebut, sehingga akan menjadi lebih efisien.

4. Kesimpulan

Berdasarkan dari penelitian tugas akhir yang berjudul "Implementasi CORBA pada Pemrograman Lintas Platform Java dan Delphi untuk Membangun Aplikasi Sistem Terdistribusi" yang diimplementasikan pada aplikasi informasi akademik, maka dapat diambil beberapa kesimpulan sebagai berikut. Teknologi CORBA telah berhasil diimplementasikan pada aplikasi informasi akademik terdistribusi yang dapat diakses pengguna melalui layanan *web*, layanan pesan singkat, dan aplikasi desktop. Aplikasi-aplikasi yang dibangun dengan *platform* bahasa pemrograman yang berbeda (Java dan Borland Delphi 7) dan berada pada server atau mesin komputer yang berbeda dapat saling berkomunikasi dan bekerja sama melalui komunikasi jaringan komputer dengan memanfaatkan teknologi CORBA.

Referensi

[1]. Li M, Puder A, Schieferdecker I. *A Test Framework for CORBA Interoperability*. Proceedings of the Fifth International Enterprise Distributed Object Computing Conference (EDOC'01), IEEE. 2001.

[2]. Blair G, Coulouris G, Dollimore J, Kindberg T. Distributed Systems: Concepts and Design. 5th Edition. Boston: Addison-Wesley. 2012.

[3]. Steen MV, Tanenbaum AS. Distributed Systems: Principles and Paradigms. 2nd Edition. Upper Saddle River: Prentice Hall. 2007.

[4]. Pilhofer F, Puder A, Romer K. Distributed Systems Architecture: A Middleware Approach. San Francisco: Morgan Kaufmann. 2006.

[5]. Bukhres O, Tari Z. Fundamentals of Distributed Object Systems: The CORBA Perspective. Hoboken: Wiley-Interscience. 2001.

[6]. Koller J, Tobermann JC. Global CAPE-OPEN: D822 Migration Cookbook. http://www.colan.org/Download/D822_migration_cookbook.pdf. 2002.

[7]. Henning M, Vinoski S. Advanced CORBA Programming with C++. Boston: Addison-Wesley. 1999.

[8]. Sekler M, Zambon G. Beginning JSP, JSF and Tomcat Web Development: From Novice to Professional. New York City: Apress. 2007.

[9]. Armstrong E, dkk. The J2EE 1.4 Tutorial For Sun Java System Application Server Platform. Edition 8.2. California: Sun Microsystems. 2005.

[10]. Pacheco X, Teixeira S. Delphi 5 Developer's Guide. Indianapolis: Sams Publishing. 2000.

[11]. Object Management Group. *The Common Object Request Broker: Architecture and Specification*. 1999.

[12]. ----. Java Platform Standard Edition 7 Documentation. http://docs.oracle.com/javase/7/docs/. Oracle Corporation. 2012.

[13]. ----. Developer's Guide Borland Delphi 7 for Windows. Borland Software Corporation. Scotts Valley. 2002.

[14]. ----. Delphi Language Guide, Delphi for Microsoft Win32, Delphi for the Microsoft .NET Framework. Borland Software Corporation. Scotts Valley. 2004.