ALIH FILE DATA INFORMASI PROSES PADA SISTEM INSTRUMENTASI DAN KENDALI REAKTOR BERBASIS OBJEK TERDISTRIBUSI MENGGUNAKAN JAVA J2SDK-1.4.2

Mohamad Amin HD
P2PN-BATAN

ABSTRAK

Alih file data informasi proses pada sistem instrumentasi dan kendali reaktor telah dilakukan menggunakan model sistem berbasis objek terdistribusi dengan arsitektur CORBA. Implementasi model ini dilakukan menggunakan bahasa pemrograman java, dengan langkah-langkah pembuatan dan pemetaan file interface dan meletakkannya di klien dan server, membuat program implementasi, membuat program server untuk melayani invokasi, dan membuat program klien untuk melakukan invokasi alih file data.

1. PENDAHULUAN

Perkembangan teknologi jaringan komputer, baik secara langsung maupun tidak, telah melahirkan pengetahuan baru di bidang lain, misalnya sistem topologi jaringan, sistem komunikasi data, sistem basisdata jaringan, dan lain-lain. Salah satu bentuk nyata aplikasi pengetahuan yang paling mudah dijumpai saat ini yaitu teknologi penjelajahan jaringan. Teknologi ini umumnya disebut teknologi browsing.

Pada teknologi browsing, data atau informasi yang ingin dilihat atau dibaca oleh seseorang diletakkan di sebuah komputer. Komputer ini disebut komputer pelayan atau server. Sedangkan komputer yang digunakan untuk melihat data atau informasi disebut komputer klien. Komunikasi antara komputer klien dan server dapat terjadi karena adanya teknologi sistem komunikasi data yang telah disepakati secara umum.

Sistem komunikasi data yang telah disepakati secara umum ini disebut protokol. Dengan berdasarkan pada protokol inilah berkembang model-model komunikasi jaringan. Model-model komunikasi jaringan yang banyak digunakan saat ini ada empat, yaitu Remote Procedure Call (RPC), Remote Method Invocation (RMI), Message-Oriented Middleware (MOM), dan streams.

Perkembangan model-model komunikasi jaringan telah menghasilkan beberapa metode dan perangkat lunak yang dapat digunakan untuk mempertukarkan informasi antara dua buah proses yang sedang berjalan pada dua komputer berbeda dalam suatu jaringan.

Pertukaran informasi antara dua buah proses ini disebut interproses communication. Salah satu metode pertukaran informasi yang dimaksud adalah CORBA (Common Object Request Broker Architecture) dan salah satu perangkat lunak yang menyediakan fasilitas pertukaran informasi ini adalah bahasa Java.

Informasi proses adalalah suatu data dari hasil proses yang telah diolah. Informasi Proses pada sistem instrumentasi dan kendali reaktor yang telah dikembangkan dan sedang diaplikasikan di reaktor Kartini Yogyakarta saat ini terletak di dalam ruang kontrol. Beberapa usaha telah dilakukan untuk menampilkan informasi proses tersebut pada jaringan komputer lokal di lingkungan reaktor, namun metode yang digunakan masih bersifat request-response dengan tampilan berbasis web. Permasalahan dari metode ini adalah informasi proses baru akan ditampilkan ketika dilakukan request ke server oleh seseorang, sehingga menimbulkan masalah pada saat diinginkan monitoring informasi proses secara terus menerus. Di samping itu pertukaran informasi dan alih data antara komputer di ruang kontrol dan komputer dijaringan tidak mudah untuk diimplementasikan menggunakan metode tersebut.

Dalam makalah ini, penulis termotivasi untuk mencoba mengalihkan dan menampilkan informasi proses pada sistem

instrumentasi dan kendali reaktor yang ada di ruang kontrol reaktor ke komputer jaringan lokal secara persisten dan berbasis objek terdistribusi dengan arsitektur CORBA. Dengan metode ini, data asli yang berbentuk sebuah file, diletakkan di komputer server di ruang kontrol reaktor, sedangkan method, prosedur, atau fungsinya diletakkan dikomputer klien di dalam jaringan itu. Komputer klien bertugas menampilkan informasi proses secara terus menerus. Tulisan ini membahas tata cara melakukan peralihan file data dari server ke klien dengan metode CORBA. Untuk mengimplementasikan metode ini maka digunakan bahasa permrograman Java.

2. METODOLOGI

Metodologi yang digunakan dalam makalah ini adalah perancangan dan implementasi sistem alih data berbasi objek terdistribusi dengan langkah-langkah sebagai berikut:

Petama, membuat model rancangan fisik dari Informasi proses pada SIK reaktor menggunakan model diagram arus data Codd-Yourdon dan model rancangan sistem alih file data menggunakan sistem berbasis objek terdistribusi.

Kedua, implementasi sistem dengan membuat file interface, dilanjutkan dengan pemetaaan ke bahasa pemrograman java versi 1.4.2.

Ketiga, membuat file implementasi dalam bahasa java untuk mengaplikasikan file interface.

Keempat, membuat file server dan file client dengan ahasa java.

3. HASIL DAN PEMBAHASAN

Model rancangan fisik dari sistem informasi proses terdistribusi pada sistem instrumentasi dan kendali reaktor dilukiskan dengan menggunakan diagram aliran data seperti diperlihatkan pada LAMPIRAN gambar 1.

Pada gambar 1 terlihat bahwa data informasi proses berasal dari sensor-sensor sistem instrumentasi kendali yang diakuisisi oleh komputer server berdasarkan pada periode waktu yang telah ditentukan. Komputer server kemudian mengumpulkan data-data tersebut, memproses, dan menampilkan-nya dalam bentuk grafik-grafik, dan sekaligus menyimpanya ke dalam sebuah file. File ini kemudian dialihkan ke komputer klien, diproses, kemudian ditampilkan sebagai informasi proses dari sistem instrumentasi kendali reaktor. Pemilihan model dengan memindahkan file menjadi pilihan dalam rancangan ini dengan beberapa pertimbangan. Pertimbangan-pertimbangan tersebut antara lain, untuk menghindari hilangnya komponen data sensor pada saat pengiriman data dari server ke klien berlangsung, untuk mengantisipasi terjadinya ***shutdown*** atau hang pada server saat berlangsung alih data.

Model rancangan alih data sistem informasi proses menggunakan arsitektur CORBA dapat dilihat pada gambar 2. Gambar ini memperlihatkan bahwa pengalihan data yang berupa file dimulai ketika program aplikasi di klien melakukan permintaan atau ***request*** ke server. ***Request*** dari klien ini selanjutnya diterima oleh ORB sebagai pengelola komunikasi berbasis objek terdistribusi. ORB kemudian mengirim request ini ke mediator yang disebut POA yang bertugas untuk menghubungkan antara ORB dan bahasa aplikasi diserver yang sering disebut ***servant***. Setelah ***request*** diterima oleh **servant**, maka ***servant*** selanjutnya mengola request tersebut dan selanjutnya mengirim hasilnya kembali ke klien pemilik ***request*** melalui POA, dan ORB.

3.1. File Interface

Langkah awal yang mesti dilakukan dalam membuat program aplikasi pengalihan data menggunakan arsitektur CORBA berbasis objek terdistribusi yaitu dengan menyiapkan file interface. File interface ini bertugas sebagai bahasa untuk mendefinisi cara melakukan antarmuka pengalihan data dari server ke klien. Dalam tulisan ini file interface diberi nama FileInterface.idl. Ekstensi file ini adalah idl yang merupakan singkatan dari ***interface definition language***. Isi dari file interface ini adalah sebuah fungsi yang akan digunakan untuk memindahkan file data. Bentuk dari isi file ini sebagai berikut :

```
interface FileInterface
{
  typedef sequence<octet> Data;
  Data downloadFile(in string
  fileName);
};
```

Bentuk dari isi file tersebut selanjutnya dipetakan ke bahasa aplikasi java di komputer server dan klien menggunakan kompilator java berbasis CORBA yang disebut ***idlj***.
Untuk pemetaan ke bahasa aplikasi java di komputer server digunakan perintah sebagai berikut :

Server> *idlj –fall* FileInterface.idl

File-file pemetaan dari hasil kompilasi FileInterface.idl diletakkan di dalam direktori Server> dengan nama-nama file masing-masing adalah sebagai berikut:
FileInterface.java, FileInterfaceHelper.java,
FileInterfaceHolder.java,
FileInterfaceOperations.java,
FileInterfacePOA.java,
_FileInterfaceStub.java,
dan sebuah direktori yang tercipta di dalam direktori Server> yakni direktori dengan nama FileInterfacePackage.
Direktori ini berisi berisi file DataHelper.java dan DataHolder.java.
Tahap selanjutnya yaitu melakukan kompilasi FileInterface.idl di komputer klien menggunakan perintah sebagai berikut:

Klien> *idlj -fclient* FileInterface.idl

Hasil dari dari kompilasi ini yaitu file-file FileInterfaceStub.java, FileInterface.java,
FileInterfaceHelper.java,
FileInterfaceHolder.java,
FileInterfaceOperations.java yang berada di dalam Klien> dan sebuah direktori yang terletak di dalam Klien> bernama FileInterfacePackage. Direktori ini berisi file DataHelper.java dan DataHolder.java.

3.2. File Implementasi

File Implementasi adalah file yang digunakan untuk menjabarkan file interface yang telah didefinisikan pertama kali. File implementasi ini merupakan turunan dari file FileInterfacePoa.java. File implementasi diletakkan di server karena bertindak sebagai servant sebagaimana yang diperlihatkan oleh Gambar 3.. File implementasi yang digunakan dalam tulisan ini dinamakan FileServant.java. Isi dari file FileServant.java ini diterangkan dengan pseudocode sebagai berikut:

1 // impor librari io dan org.omg.CORBA dari java;
2 // buat kelas bersifat publik dengan nama FileServant.
3 // Kelas ini adalah turunan dari FileInterfacePOA;
//{ Di dalam kelas ini deklarasikan:
3.1//Tipe data ORB yang bersifat privat;
3.2//Konstruktor untuk menampung tipe data ORB;
3.3//Fungsi downloadFile bersifar publik, dengan masukan berupa namafile bertipe string, dan keluaran adalah berupa isi dari namafile dengan tipe larik byte { Dalam fungsi ini dideklarasikan
3.3.1// Tipe data File yang akan menampung file dengan nama namafile;
3.3.2// larik buffer bertipe byte yang akan menampung panjang file;
3.3.3// {FileServant selanjutnya mencoba untuk melakukan:
3.3.3.1//Menyiapkan variabel penampung stream data yang masuk dari namafile
3.3.3.2//variabel penampung menampung data file dari awal file ke akhir file kemudian memberikannya ke larik buffer.
tutup variabel penampung isi file. //}
3.3.3.3//{Jika gagal maka tampilkan pesan gagal di layar monitor.}
3.3.4// kembalikan isi buffer ke fungsi downloadFile }
//}akhir dari kelas

3.3. File Aplikasi Server

File aplikasi server secara umum melakukan tugas-tugas berikut:

1. Menciptakan sebuah instans ORB dan menginisialisasinya.
2. Menciptakan sebuah instans dari kelas implementasi interface dan memberinya ORB hasil inisialisasi.
3. Mendapatkan sebuah referensi ke RootPOA dan mengaktifkan POAManager.
4. Mendapatkan sebuah rujukan objek dari file implementasi.
5. Mengambil konteks penamaan root dari layanan penamaan dan memberikan objek baru dengan nama "FileTransfer".
6. Menanti hingga terjadi permintaan dari klien.

File aplikasi server yang digunakan di dalam tulisan ini dinamakan FileServer.java. Isi dari file FileServer.java

ini ditulis menggunakan pseudocode sebagai berikut:

```
//impor java.io.*;org.omg.CORBA.*;
org.omg.CosNaming.*;
//impor org.omg.PortableServer.*;
org.omg.PortableServer.POA;
//impor
org.omg.CosNaming.NamingContextPacka
ge.*;
//;
// publik kelas FileServer {
// publik statik void main (String args[]) {
// coba{
// 1. ciptakan dan inisialisasi ORB
//       ORB orb = ORB.init(args, null);
// 2. ciptakan satu instans dari file
     servant dan berikan instans orb.
//    FileServant srvRef = new
     FileServant(orb);
//3. Dapatkan referensi ke rootpoa dan
     aktifkan POAManager
//    POA poa = POAHelper.narrow
    (orb.resolve_initial_references ( __
     "RootPOA"));
//    poa.the_POAManager().activate();
//4. Dapatkan referensi objek yang
     berasal dari file servant
//    org.omg.CORBA.Object ref =
poa.servant_to_reference(srvRef);
//    FileInterface href =
FileInterfaceHelper.narrow(ref);
 //5. Ambil konteks penamaan root.
NameService memanggil name service.
// org.omg.CORBA.Object objRef =
orb.resolve_initial_references("
NameService");
// Gunakan NamingContextExt sebagai
bagian interoperable name service
// NamingContextExt ncRef =
   NamingContextExtHelper.narrow
(objRef);
// Satukan referensi objek berdasarkan
    nama
// String name = "FileTransfer";
// NameComponent path[] = ncRef.to_
    name( name );
// ncRef.rebind(path, href);
// System.out.println("Server sedang
    aktif....");
// Wait for invocations from clients
// orb.run();
//  } catch(Exception e) {
     System.err.println("ERROR: " +
    e.getMessage());
    e.printStackTrace(System.out);}
// System.out.println("Tugas Transfer File
   Selesai ....."); }}
```

3.4. File Aplikasi Client

File aplikasi client dijalankan di komputer klien. File aplikasi klien secara umum melakukan tugas-tugas berikut:

1. Menciptakan dan menginisialisasi ORB.
2. Mendapatkan sebuah referensi ke konteks penamaan root.
3. Mengamati objek"FileTransfer" di dalam konteks penamaan dan mendapatkan referensi ke konteks penamaan tersebut.
4. Memanggil fungsi downloadFile.
5. Menyimpan hasil downloadFile dengan nama yang tidak berubah ke direktori FileClient.

File aplikasi client yang digunakan di dalam tulisan ini dinamakan FileClient.java. Isi dari file FileClient.java ini ditulis menggunakan pseudocode sebagai berikut :

```
//impor java.io.*;impor java.util.*;impor
org.omg.CORBA.*;
//impor
org.omg.CosNaming.NamingContextPacka
ge.*;
//impor org.omg.CosNaming.*;
//;
//publik kelas FileClient {
//publik statik void main(String argv[]) {
//coba{
//1. ciptakan dan inisialisasi ORB
//    ORB orb = ORB.init(argv, null);
//2. ambil konteks penamaan root
//    org.omg.CORBA.Object objRef =
orb.resolve_initial_references("__
          NameService");
//3. NamingContextExt sebagai pengganti
Naming Context
//NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);
//    String name = "FileTansfer";
//    NameComponent path[] =
ncRef.to_name(name);
        //    FileInterfaceOperations fileRef
= FileInterfaceHelper.narrow( __
//    ncRef.resolve(path));
//  if(argv.length < 1) {
// System.out.println ("Gunakan/
  ketikkan: java  FileClient  namafile");}
//  mendownload dan menyimpan file
//  File file = new File(argv[0]);
//  byte data[] = fileRef.
    downloadFile(argv[0]);
//  BufferedOutputStream output = new
//  BufferedOutputStream(new
FileOutputStream(argv[0]));
```

```
// output.write(data, 0, data.length);
// output.flush();
// output.close();
// } catch(Exception e) {
// System.out.println("FileClient Error: " +
e.getMessage());
// e.printStackTrace(); }}}
```

3.5. Pengujian Alih Data

Untuk mengetahui bahwa metode sistem berbasis objek terdistribusi ini bekerja atau tidak maka dilakukan pengujian. Langkah-langkah yang diperlukan dalam melakukan pengujian adalah sebagai berikut :

1. Menjalankan layanan penamaan CORBA dengan menggunakan perintah ***orbd*** baik di server maupun di klien. Misalkan Server> adalah prompt di server, dan Client> adalah prompt di klien, maka perintahnya adalah sebagai berikut :

Server>orbd –ORBInitialPort 2500 –ORBInitialHost 192.162.4.4

Client>orbd –ORBInitialPort 2500 –ORBInitialHost 192.162.4.4

Hasil eksekusi dari perintah orbd ini diperlihatkan oleh Gambar 4 pada lampiran.

2. Menjalankan program server dengan menggunakan perintah ***java***. Contoh :

Server>java FileServer –ORBInitialPort 2500 –ORBInitialHost 192.162.4.4

Hasil eksekusi dari perintah java FileServer ini diperlihatkan oleh Gambar 5 pada lampiran.

3. Menjalankan program klien juga dengan menggunakan perintah ***java***. Contoh berikut adalah menjalnkan program FileClient untuk mengalihkan data sistem informasi proses dengan ekstensi txt yang berada di komputer server direktori c, kemudian memindahkannya ke klien.

Client>java FileClient c:sisfoProses.txt –ORBInitialPort 2500 –ORBInitialHost 192.162.4.4

Hasil eksekusi dari perintah java fileclient ini diperlihatkan oleh Gambar 6 pada lampiran.

4. KESIMPULAN

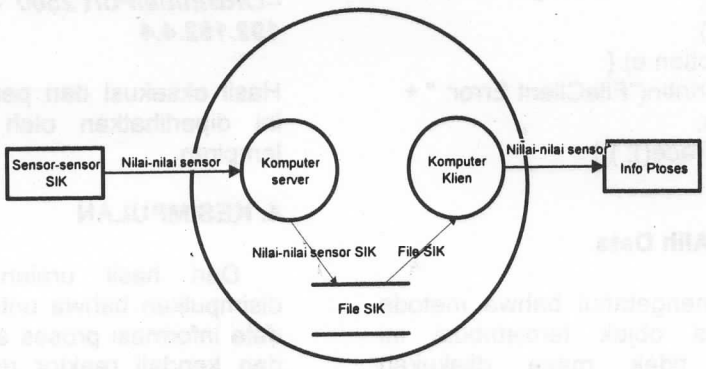
Dari hasil uraian di atas, dapat disimpulkan bahwa untuk mengalihkan file data informasi proses sistem instrumentasi dan kendali reaktor dari server ke klien menggunakan metode yang berbasis pada objek terdistribusi dengan arsitektur CORBA dan dengan bahasa aplikasi java, maka dilakukan langkah-langkah sebagai berikut :

1. Membuat file interface yang disebut file idl.
2. Melakukan kompilasi file idl untuk dipetakan server dan klien.
3. Membuat program implementasi diserver.
4. Membuat program aplikasi di server untuk melayani permintaan klien.
5. Membuat program aplikasi di klien untuk melakukan permintaan pemindahan file data.

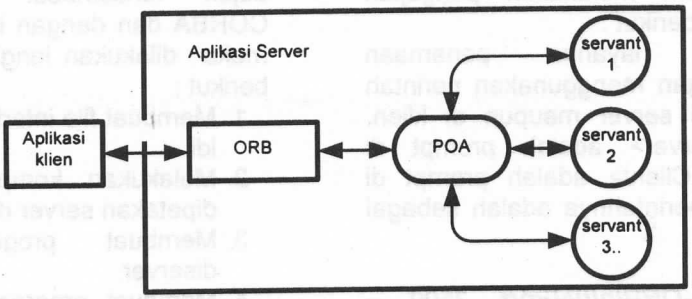
ACUAN

1. TANEMBAUM ANDREW S.,VAN STEEN MARTEEN. *Distributed Systems, principles and paradigms*, Prentice Hall, International Editions, 2002
2. BENNY HERMAWAN. Menguasai Java 2 dan Object Oriented Programming. ANDI yogyakarta, 2004.
3. QUSAY MOHAMAD. *Distributed Java Programming with RMI and CORBA*, article, January 2002
4. QUSAY MOHAMAD. *CORBA Programming with J2SE 1.4*, article, Mei 2002.
5. QUSAY MOHAMAD. *CORBA Communication*, article, April 2001.

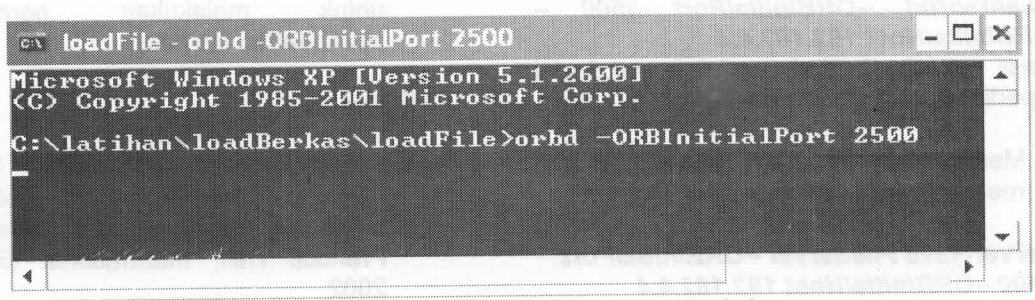
LAMPIRAN



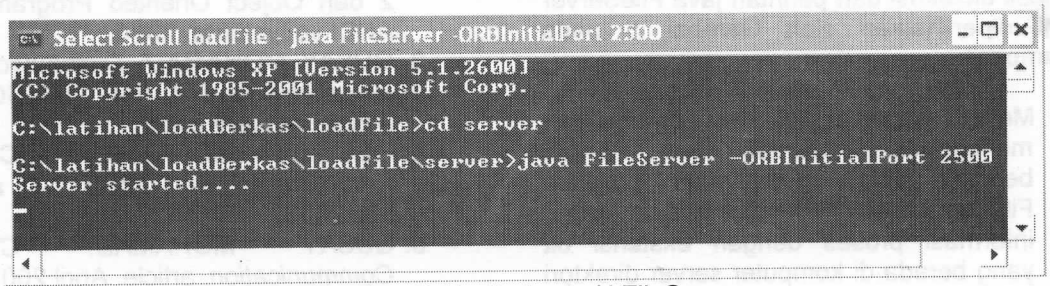
Gambar 1. Model Rancangan Fisik Informasi Proses pada SIK Reaktor



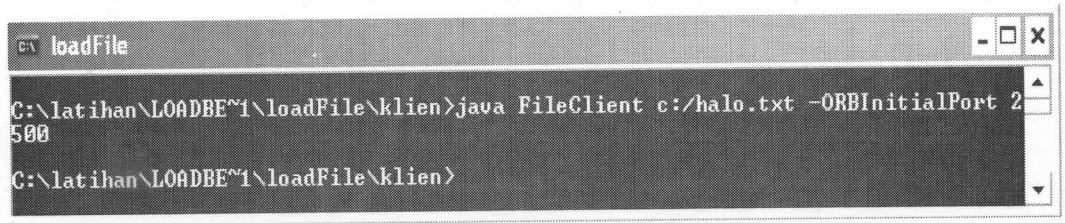
Gambar 2. Model rancangan alih data



Gambar 4. Hasil eksekusi perintah **orbd**



Gambar 5. Hasil eksekusi FileServer



Gambar 6. Hasil eksekusi FileClient