IMPLEMENTASI REMOTE METHOD INVOCATION (RMI) UNTUK TES ONLINE INTERAKTIF MULTIUSER PADA LOCAL AREA NETWORK (LAN)

Adrian Nathaniel Wikana [1], Joko Purwadi[2], Restyandito[3]

Abstrak: Teknologi informasi telah berkembang dengan pesat dan membawa banyak paradigma perubahan dalam aktivitas manusia dalam betajar dan bekerja. Perkembangan teknologi informasi saat ini cenderung mengarah pada sistem terdistribusi (*client/server*) dimana fungsi sebuah komputer dapat digunakan secara terkoordinasi dengan beberapa komputer lainnya. Perkembangan tersebut dituntut untuk agar dapat dioperasionalkan dan dimanfaatkan secara optimal di berbagai bidang. Dalam bidang pendidikan perkembangan teknologi *client/server* dapat dimanfaatkan untuk membuat sistem ujian terkomputerisasi. Salah satu metode teknologi *client/server* adalah RMI. Artikel ini membahas bagaimana implementasi RMI pada tes *online* termasuk didalamnya teknologi tambahan yang dibutuhkan untuk mendukung pengimplementasian, dan keunggulan teknologi ini.

Kata Kunci: *Remote Method Invocation (RMI), Server, Client, Database Server, Java Database Connectivity (JDBC)*

1. Pendahuluan

Teknologi informasi telah berkembang dengan pesat dan membawa banyak paradigma perubahan dalam aktivitas manusia dalam belajar dan bekerja. Perkembangan teknologi informasi saat ini cenderung mengarah pada sistem terdistribusi *(client/server)* dimana fungsi sebuah komputer tidak hanya terbatas pada mesin mandiri, tetapi dapat juga digunakan secara terkoordinasi dengan beberapa komputer lainnya. Perkembangan tersebut dituntut untuk dapat dioperasionalkan dan dimanfaatkan secara optimal di berbagai bidang.

Bidang pendidikan adalah salah satu bidang yang dapat memanfaatkan perkembangan teknologi *client/server.* Salah satu bentuk pemanfaatan teknologi client/server di bidang pendidikan yang menunjang perkuliahan adalah tes *online. Remote Method Invocation* (RMI) adalah salah satu metode dalam sistem terdistribusi yang dapat diimplementasikan untuk tes *online.* Penulisan artikel ini membahas implementasi RMI untuk Tes *Online* pada *Local Area Network* (LAN).

2. Landasan Teori

Program aplikasi modem yang ada dan berkembang saat ini cenderung mengarah pada model *client/server. Client/server* ditandai dengan adanya *request* atau permintaan suatu informasi ke *server* dan *respon* atau hasil pemrosesan yang dikembalikan ke *client* dari *server.* Perkembangan program *client/server* yang cukup pesat saat ini mempengaruhi model pemrograman yang ada menjadi berbasis *client/server* pula. Java dengan *Remote Method Invocation* (RMI) merupakan salah

[1] *Adrian Nathaniel Wikana , Mahasiswa Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[2] *Joko Purwadi, S.Kom, M.Kom , Dosen Teknik Inforrnatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[3] *Restyandito, S. Kom, MSIS, Dosen Teknik Inforrnatika, Fakultas Teknik, Universitas Kristen Duta Wacana*

satu teknik pemrograman *client/server* tersebut. Untuk sebuah aplikasi yang membutuhkan koneksi dengan *database server* maka dibutuhkan metode yaitu *Java Database Connectivity* (JDBC) yang dapat mengbungkan aplikasi Java dengan *database server.*

Pada bagian ini akan dijelaskan secara teoritis tentang konsep *client/server, Java Remote Method Invocation* (RMI) danJava *Database Connectivity* (JDBC).

2.1. *Client/Server*

Client/Server mempakan item proses atau logika terpisah yang bekerja sarna pada suatu jaringan komputer untuk mengerjakan suatu tugas tertentu. Dalarn sebuah arsitektur *client/server, client* berperan dalarn mengambil data dari user dan melakukan pengiriman ke *server. Server* berperan dalarn proses penerimaan data dari *client,* melakukan pemrosesan berdasarkan permintaan tersebut, dan mengembalikan ke *client* sebagai hasil pemrosesan yang sudah dilakukan. Sebuah *server* dapat melayani beberapa *client.*

Komputer yang meminta layanan *(request)* disebut sebagai *client,* sedangkan yang menyediakan layanan *(respon)* disebut sebagai *server.*

Client-Server memiliki beberapa karakteristik sebagai berikut:

1. *Multitasking: Server* dapat melayani beberapa *client* pada saat yang sama dan mengatur pengaksesan sumberdaya.

2. Adanya pemisahan *tier* dan fungsi sistem: *Client/Server* memisahkan tier dan layanan menjadi *presentation, business logic* dan *data tier.* Dengan pemisahan ini beban komputasi antara *server* dan *client* menjadi seimbang, dan *server* dapat diakses oleh beberapa *client (multiuser).*

2.2 Java *Remote Method Invocation* (RMI)

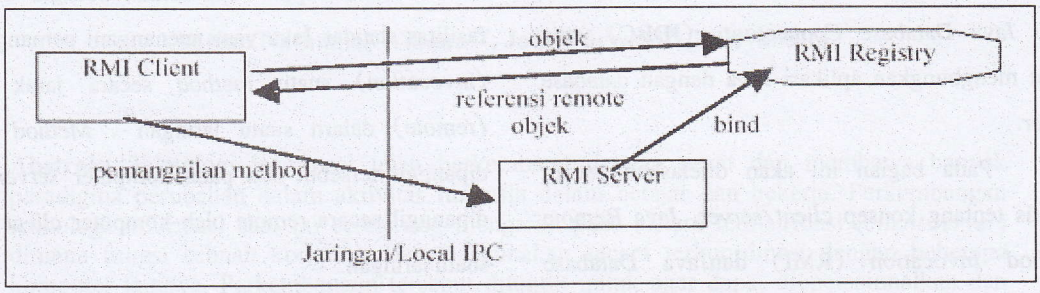
RMI dapat didefinisikan sebagai sebuah fasilitas standar Java yang menangani pemanggilan *(invocation)* suatu *method* secara jarak jauh *(remote)* dalarn suatu jaringan[4]. *Method* yang dipanggil tersebut ada pada komputer *server* dan dipanggil secara *remote* oleh komputer *client* pada suatu jaringan.

Distribusi aplikasi RMI terdiri dari dua buah program yang terpisah yaitu program *server* dan program *client.* Program *server* bertugas membuat beberapa *remote objek* dan mendaftarkan *(bind) remote* objek tersebut ke *RMI Registry* dengan nama yang unik. Program *client* bertugas membuat koneksi ke *server* dan meminta pemanggilan ke *remote* objek berdasar referensi yang diterimanya dari *RMI Registry. RMI Registry* berfungsi mengkonversi nama objek yang didaftarkan *(bind)* oleh *server* menjadi *remote objek reference. Remote Objek Reference* adalah referensi yang berisi *method-method* yang dapat dipanggil secara *remote* oleh *client.* Referensi tersebut dibutuhkan oleh *client* agar dapat mengetahui dan memanggil *remote method* pada *server. Method* yang bisa dipanggil secara *remote (remote method)* punya *remote inteiface*[5]. *Remote inteiface mendefinisikan* metode-metode yang bisa diakses secara *remote. Remote inteiface* dapat berupa prosedur atau fungsi.

Mekanisme kerja yang ada pada sebuah aplikasi RMI dimulai saat *RMI server* akan mendaftarkan *remote* objeknya ke *RMI Registry.* Proses ini disebut dengan *bind,* yaitu mendaftarkan sebuah objek dengan suatu nama yang unik. Dengan terdaftarnya *remote* objek di *RMI Registry* maka *RMI Registry* dapat memberikan referensi ke *RMI*

[4] *Budi Susanto, 2003, Pemrograman Client/Server Dengan Java2, Jakarta, Elex Media Komputindo, hlm. 137.*
[5] *Ibid.,hlm.173.*

Client, tentang *remote* objek mana yang bisa diakses saat *RMI Client* melakukan pemanggilan *method (request)*. Setelah referensi diperoleh, *RMI Client* dapat memanggil *method* yang ada di *server*, secara *remote*. *RMI Server* mengirimkan *respon* setelah dilakukan pemrosesan ke *RMI Client*.



Gambar 1. Mekanisme Kerja *Remote Method Invocation* (RMI)

Untuk membuat aplikasi *client/server* berbasis RMI ada empat *class* yang harus didefinisikan:

a. *Class remote interface*

```
public interface tes extends java.rmi.Remote {
public void setJawaban(String text)
Throws java.rmi.RemoteException;
public String getStatus()
throws java. rmi. RemoteException;
}
```

b. *Class* implementasi dari *interface*

```
public class tesImpl
extends java.rmi.server.UnicastRemoteObject implements tes {private String database;
public tesImpl() throws java.rmi.RemoteException
{
super ();
}
public void setkirimjawab(String s) throws
Java.rmi.RemoteException
{
this.database = s;
}
public String getjawab() throws java.rmi.RemoteException
{
return this.s;
}}}
```

c. *Class Remote Server (Server)*

```
import java.rmi.Naming;

public class SoalServer {
public SoalServer() {
try {
tes server = new tesImpl();
Naming.rebind("rmi://localhost:l099/tes",server) ;
}
catch (Exception e) {}}

public static void main (String args[]) {
new Server();
}}
```

D. *Class Remote Client (Client)*

```
import java.rmi.Naming;
import java.rmi.RemoteException;
```

```
import java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class Client {
public static void main(String[] args) {
System.setSecurityManager(new java.rmi.RMISecurityManager());
try {
tes client = (tes) Naming.lookup(nrmi://localhost/tesn);
        client.setJawab(nJawaban soal 1. . .");
System.out.println(client.getStatus ());

}
catch (MalformedURLException murle) {}
catch (RemoteException re) {}
cacth (NotBoundException nbe) {}
}}
```

2.3 *Java Database Connectivity* (JDBC)

JDBC merupakan *middleware database* yang dapat menghubungkan aplikasi Java dengan *database server.* Untuk dapat menghubungkan aplikasi Java dengan *database* melalui JDBC ada beberapa langkah yang perlu dilakukan yaitu[6]:

a. Mengakses JDBC *driver*

Contoh: *Class.* forName,(' 'com.postgresql. Driver' ') ;

b. Mempersiapkan koneksi ke *database* yang dibuat lewat *class Connection,* dan *class DriverManager. DriverManager* adalah *class* yang digunakan untuk memilih *database driver* dan membuat koneksi ke *database* tersebut

Contoh: Connection con =

DriverManager.getConnection(''jdbc:postgresql",usrnm,pswd);

c. Mempersiapkan objek untuk menerima perintah SQL lewat *class Statement. Class Statement* digunakan untuk mendefinisikan metode yang berhubungan dengan pemyataan *Structured Query Language* (SQL).

Contoh: Statement stat = con.createStatement() ;

d. Mengeksekusi perintah SQL. Untuk pengaksesan hasil pemyataan SQL digunakan *inteiface ResultSet.* Untuk mengeksekusi perintah yang dapat mengambil data digunakan perintah *executeQuery(' ').* Untuk mengeksekusi perintah yang dapat memasukkan data, atau mengubah data digunakan perintah *executeUpdate(' ').*

Contoh:

```
ResultSet hasilQuery = null;
try
{
hasilQuery = stat.executeQuery("SELECT nim FROM mahasiswa");
}
catch (Exception ex) {}
try {
while (hasilQuery.next())
{
String nimmhs = hasilQuery.getString("nim") ;
System.out.println (nimmhs) ;
}}
catch (Exception e) {}
}}
```

[6] *Rangsang Pumama, 2005, Tuntunan Pemrograman Java Jilid 3, Prestasi Pustaka, Jakarta, hlm.123.*

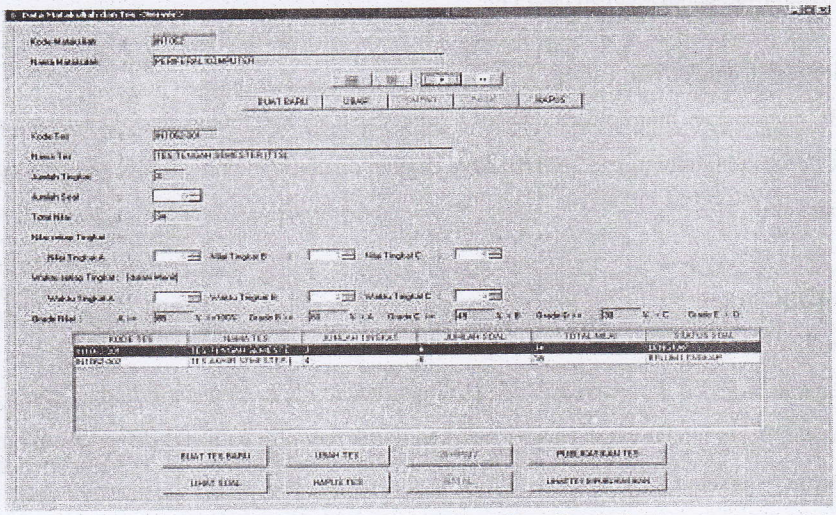
e. Menutup perintah dan koneksi yan gtelah dibuka

Contoh: stat. close () ;

Con.close() ;

3. Basil Implementasi RMI Untuk Tes *Online* Pada LAN

Pada bagian ini disajikan hasil pengimplementasian RMI untuk Tes *Online*dimulai dari program *server* sampai program *client.* Tampilan utama dari program *server* untuk memasukkan data soal tes dapat dilihat pada Gambar 2.

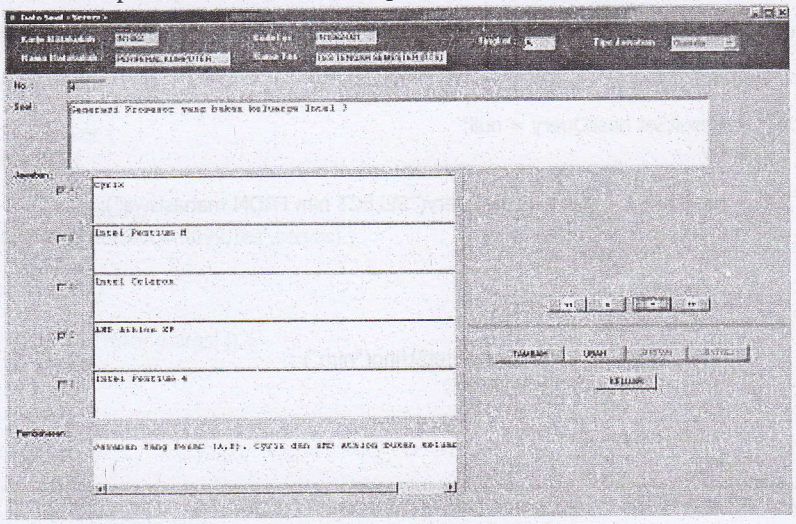


Gambar 2. Program *Server- Form* Data Tes *Online*

Pada *form* diatas program *RMI Server* akan di *bind* ke *RMI Registry.* Proses bind dijalankan dengan mengeksekusi perintah berikut:

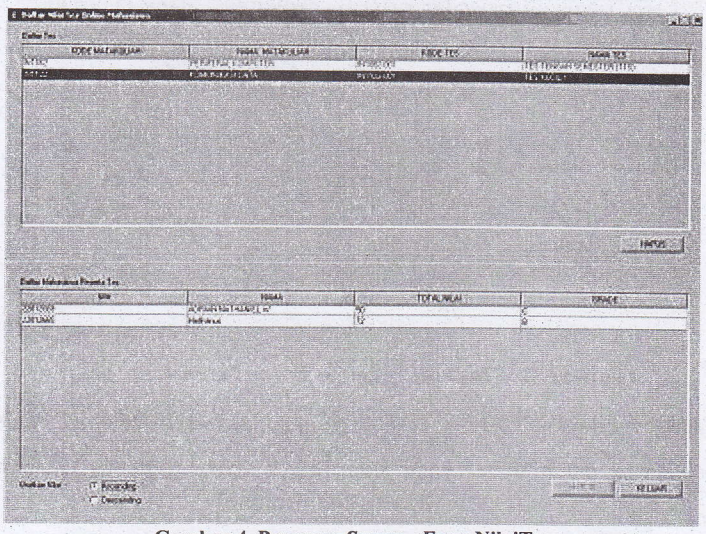
```
try {
tes server = new tesimpl();
Naming.rebind("rmi://127.0.0.1:1099/tes",server);
}
catch (Exception e) {}
```

Berikutnya administrator dapat memasukkan soal tes pada *form* yang ditunjukkan pada gambar 3.



Gambar 3. Program *Server - Form* Soal

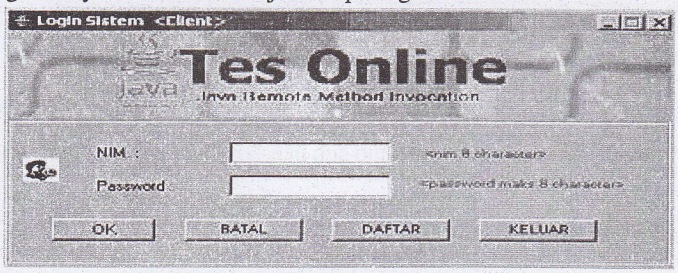
Berikutnya jika tes *online* sudah selesai, administrator dapat melihat hasil nilai tes *online* yang sudah masuk ke program *server* pada form nilai tes. *Form* untuk melihat nilai tes *online* ditunjukkan pada gambar 4.



Gambar 4. Program *Server* - *Form* NilaiTes

Pada *form server* gambar 3 dan 4 *method* yang dieksekusi adalah *Java Database Connectivity* (JDBC) untuk menyimpan atau membaca data dari *database.*

Berikutnya adalah hasil pengimplementasian program *RMI Client untuk Tes Online.* Tampilan utama dari program *client* untuk *login ke system server ditunjukkan* pada gambar 5.



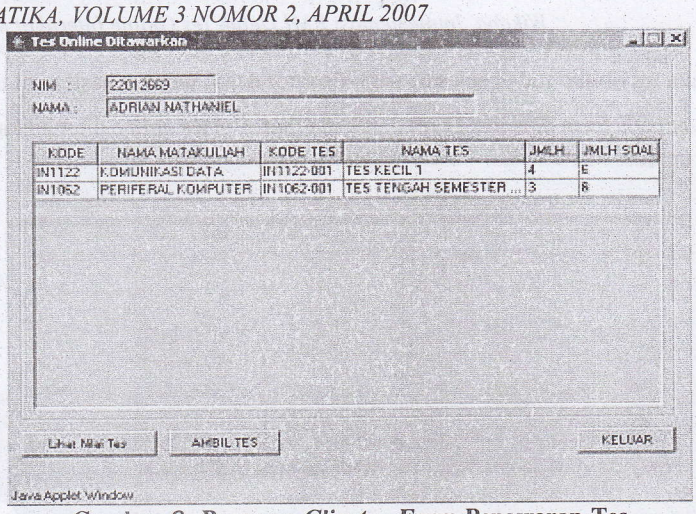
Gambar 5. Program *Client* - *Form Login*

Dengan dijalankannya program *client* maIm pemanggilan *method* pada program *Server* mulai dikerjakan. *Method* pada *RMI Server* yang dipanggil oleh *RMI Client* pada *form login* adalah *method login, method* daftar. Program *server* yang diimplementasikan pada *class* implementasi *interface* (teslmpI) mengeksekusi *method Java Database Connectivity* (JDBC) Ulltuk melakukan operasi dengan *database.*

Pada saat *form login* dijalankan, program *client* akan mencari *remote objek reference (lookup)* pada *RMI Registry* di *server.* Proses dikerjakan dengan menjalankan perintah berikut:

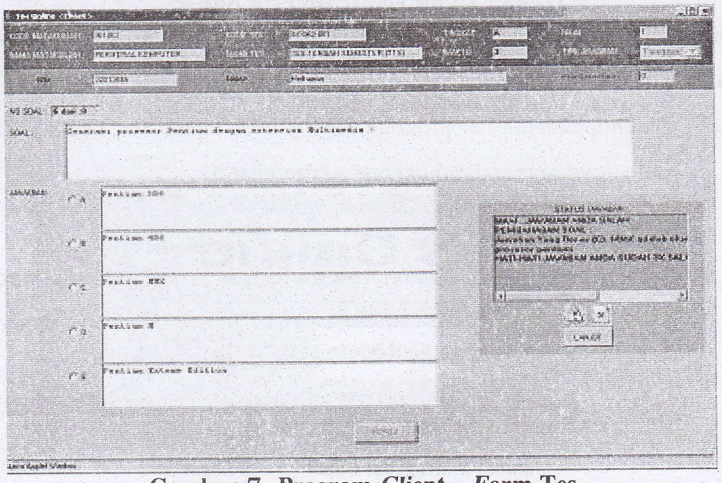
```
try {
tes tesclient = (tes) Naming.lookup("rmi://"+host+"/tes");
}
catch (MalformedURLException murle) {}
catch (RemoteException re) {}
catch (NotBoundException nbe) {}
```

Berikutnya peserta tes dapat memilih jenis tes *online* yang ditawarkan oleh *server* yang ditunjukkan pada gambar 6.



Gambar 6. Program *Client - Form* Penawaran Tes

Method yang dipanggil pada *RMI Server* pada *form* penawaran adalah *method* setTes, *method* mulai. Berikutnya setelah peserta tes memilih tes yang akan diikuti maka *form* tes untuk menjawab soal akan dijalankan, seperti tampak pada gambar 7.



Gambar 7. Program *Client _ Form* Tes

Pada *Form* Tes gambar 7, *RMI Client* akan memanggil *method* pada *RMI Server* secara *remote*. *Method* yang dipanggil adalah *method* setSoal, getSoal, setJawaban, getPembahasan.

Proses *RMI Client* memanggil *method* setSoal untuk mengambil soal dari *server* dikerjakan dengan perintah berikut:

```
tesclient.setSoal(this.kodetes,no);
String soal= tesclient.getSoal() ;
```

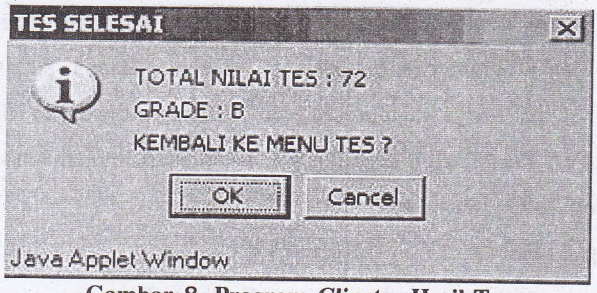
Proses pada *RMI Server* yaitu dilakukan pengambilan soal dari *database server* dengan menjalankan *method* setSoal dan mengembalikan data soal ke *RMI Client* saat *method* getSoal dipanggil. Proses dikerjakan dengan perintah berikut:

```
public synchronized void setSoal(String kode,int no) {
this.kodetes=kode.trim();
this.tingkat=tgkt.trim() ;
this.nosoal=no;
try {
Class.forName("org.postgresql.Driver");
```

```
con
DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/d btes", "postgres", "skripsi");
stm= con.createStatement();
result = stm.executeQuery("SELECT detailsoal.soal FROM
detailsoal WHERE detailsoal.no soal='"+this.nosoal+"' and detailsoal.kode_tes='"+this.kodetes+"'");
}
catch (Exception ex) {}
try {
if (result.next()) {
this.soal=result.getStrin_("soal");
}
result.close();
stm.close();
con.close();
}
catch (Exception ej {}}

public String getSoal() throws java.rmi.RemoteException {return this.soal;
}
```

Berikutnya setelah *tesonline* berakhir maka akan ditampilkan hasil nilai tes pada program *RMI Client* yang ditunjukkan pada gambar 8.



Gambar 8. Program *Client* - Hasil Tes

Program *client/server* Implementasi RMI Untuk Tes *Online* tersebut diujikan dengan sampling tiga buah komputer yaitu sebuah komputer sebagai *server*, dan dua buah komputer sebagai *client*.

4. Analisis Hasil Penelitian

Pada sub bab ini akan diuraikan analisis dari implementasi *Remote Method Invocation* (RMI) pada kasus Tes *Online*. Analisis diuraikan dengan konsep dan teori *Client/Server* sebagai berikut:

a. RMI kaitannya dengan karakteristik *Client/Server Multitasking*

Proses pada *RMI Server* memungkinkan untuk diakses oleh sejumlah *client* secara bersama-sama atau dikenal dengan istilah *Multitasking*. *RMI Server* hams mampu melayani pennintaan simultan ini dan mengembalikan hasil proses dengan benar. Agar *method* yang dipanggil tersebut dapat mengembalikan hasil yang benar meskipun dipanggil secara bersama-sama oleh sejumlah *client*, maim *method* tersebut hams mendapat teknologi pendukung yaitu sinkronisasi *(synchronized)*.

b. Sinkronisasi *Method*

Proses yang dikerjakan dengan sinkronisasi yaitu pemanggilan *method* oleh suatu *client* pada suatu objek *server* akan diproses dan diselesaikan dahulu sebelum *method* tersebut memproses pemanggilan dari *client*

lain. Ketika sebuah *method* sinkronisasi dijalankan, dan *client* kedua juga memanggil *method* tersebut, maka *client* kedua akan dideaktivasi dan menunggu *client* pertama selesai mengeksekusi *methodnya.* Dengan model demikian hasil pemrosesan *server* menjadi benar. Untuk membuat suatu *method* menjadi *method* sinkronisasi dengan menambahkan *keyword "synchronized"* pada *method* tersebut.

c. Keunggulan teknologi RMI

Beban komputasi *server* dan *client* seimbang. Program RMI adalah hasil kolaborasi *RMI Server* dan *RMI Client.* Pada RMI tidak semuanya harus dikerjakan oleh program *server,* sebagian proses dapat dikeIjakan oleh program *client.* Pada kasus Tes *Online, RMI Server* hanya memproses pengaksesan *database server.* Proses tes dikerjakan *RMI Client,* seperti *random,* menyimpan *array* data tes beserta nilainya. Dengan pembagian fungsi RMI ini maka beban *server* dan *client* menjadi seimbang.

d. Perbandingan RM1 dengan Metode *Client/Server* lainnya

RM1 mempunyai perbedaan dengan metode *client/server* lainnya seperti PHP. Perbedaan tersebut dapat ditinjau dari pemrosesan yang dilakukan. Pemrosesan *business logic* RMI dapat dikerjakan pada *server* maupun *client.* Pemrosesan *business logic* pada PHP hanya dapat dikerjakan pada *server (server side scripting).* Untuk pengembangan aplikasi skala *Enterprise* RMI mempunyai unjuk kerja yang lebih baik. Namun sistem yang dibutuhkan untuk pengembangan aplikasi dengan RMI juga. jauh lebih kompleks dan rumit dibanding teknologi PHP yang sudah banyak digunakan.

5. Kesimpulan

Berdasarkan hasil pembahasan kajian dan_analisis dari penelitian, maka dapat diambil kesimpulan sebagai berikut:

a. RMI dapat diimplementasikan untuk Tes *Online Multiuser* pada *Local Area Network*

b. Sinkronisasi method diperlukan agar proses *multitasking* pada Tes Online dapat mengembalikan hasil yang benar ketika dipanggil bersama-sama oleh *client* secara simultan

c. Beban keIja sistem *client / server* menjadi seimbang dan unjuk kerja sistem menjadi lebih handal karena adanya pembagian fungsi antara *RMI Server* dan *RMI Client.*

d. Pengembangan sistem menjadi lebih mudah, karena adanya pemisahan *tier.* Program *RMI Client* tidak perlu diubah jika ada perubahan terhadap implementasi *business logic* pada *RMI Server.*

6. Daftar Pustaka

Coulouris, George., Jean Dollimore, Tim Kindberg, 2001, *Distributed Systems Concepts and Design Third Edition,* Addison Wesley, Amerika

Deitel, H.M. , Deitell, P.l., 1997, *Java How To Program, Prentice Hall,* New Jersey

Horstmann, Cay S., Gary Cornell, 2000, *Core Java 2 Volume II-Advanced Features,* Sun Microsystems Press, California, USA

Indrajani, Martin, 2004, *Pemrograman Berorientasi Objek dengan JA VA,* Elex Media Komputindo, Jakarta

Kadir, Abdul., 2004, *Dasar Pemrograman Java* 2, Penerbit ANDI, Y ogyakarta

Purnama, Rangsang. , 2005, *Tuntunan Pemrograman Java Jilid* 3, Prestasi Pustaka, Jakarta

Susanto, Budi, 2003, *Pemrograman Client/Server Dengan Java 2*, Elex MediaKomputindo, Jakarta

Utdirartatmo, Firar, 2002, *Mengelola Database Server PostgreSQL di Linux dan Windows*, Elex Media Komputindo, Jakarta