Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, 2015, *7*, 2201-2206

Open Access

Reaserch on the Construction and Realization of Synchronization System for Wireless Spital Database Based on UUID

Song Liangong*

North China University of Water Resources and Electric Power, Zhengzhou, 450045, China

Abstract: This paper implements a database synchronization system for the SQLite. For some of the problems encountered in the design and development of systems, by looking for literature, guessing and analyzing implementation methods of these similar systems, has solved some key issues and design and implement our own solutions. Mainly UUID replaces integer data format as data representation; Design table to store synchronization information metadata; Use HTTP protocol to transmit data and resolve security authentication and encrypted transmission; Use JSON format as the serial data format on the network transmission; Use JZEE platform to process system concurrent.

Keywords: EDBMS, open source, SQLite, synchronizing system.

1. INTRODUCTION

After years of embedded systems development, it has great relationship with people learn, work, lives [1]. Embedded systems have been used in scientific research, engineering design, military technology, and various types of industrial, commercial and cultural arts, entertainment, and other aspects of people's daily lives. With the rapid development of digital information technology and network technology, computer technology, communications technology, consumer electronics integration trend is increasingly evident that beckons a huge generation of embedded applications market. Embedded system technology has become the current attention, learning research focus [2-5].

2. FEATURES OF EMBEDDED DATABASE

Implementation and application of methods and enterprise-class database embedded database has a big difference.

Embedding is the basic characteristics of an embedded database [3, 4]. Embedded database can not only be embedded into other software, but also can be embedded in a hardware device.

Real-time and embedded nature are inseparable [6]. Only a database of embedded, the first time be able to get the system resources of the system to respond to requests for the first time.

Scalability is particularly important in embedded applications [7]. First of embedded hardware and software platforms occasions are vastly different, the results are basically the customer needs according to their choice.

Consistency is necessary for the database properties [8]. Through the transaction, lock functions and data synchronization, and other technology to ensure consistency within each table in the database data, also ensure that the database and other databases in the data synchronization or mirror consistency.

Security is essential [9-12]. Ensure the physical security of the information itself, we must also ensure the safety of users of private information.

3. SPECIFIC REALIZATION OF EACH MODULE

3.1 Data Change and Monitoring Module

For synchronous client, to synchronize data, we must first know which data is modified. If there is such database operation that has executed INSERT, UPDATE or DELETE operation, it is necessary to record, to prepare for synchronization purposes. In SQLite, there is a SQLite3_update_hook () interface function. It will be executed every time when the INSERT, UPDATE, DELETE operation occurs. This system uses this interface function to achieve the monitoring of data changes. Code is as follows:

SQLite3_update_hook() (

db, update--hook, (void)db):*

Update_hook is the callback function invoked. However SQLite3 (version 3.6.10) has a bug, When the *delete* statement has no *where* clause, the *update_hook* of *delete* failures, there is no call. But when coupled with *where 1 =l*, they can trigger. It is need to pay write attention on it when writing sql statement.

After the experiment, the interface has the behavior of the document. But there is a problem is the database used to record these changes metadata is stored in the same database. When the update occurs, these records will be updated metadata table occurrence will trigger the interface function, leading to an infinite loop. So it is needed to determine whether the table is synchronized metadata table in the subroutine. Taking into account the possible expansion of the system,

1874-4443//15

the tables beginning with "sync_ ' are all counted the synchronization system table. If it is synchronizing metadata table, it do not record. This is a convention of the system, which the system uses the table beginning with "sync_ 'to storage system data. If you use this system, the user defined database table names cannot begin with "sync_". The following code to solve this problem:

```
char r[]="sync_";
char table_prefix[6]:
strncpy ( tab1e_prefix, tableName, 5 ):
table_prefix[5]='\0';
if ( stremp( r, table_prefix ) !=0 )
```

The code above detects whether the five previous chars of table name is equal "sync_". If YES, not run the rest of the code segment. Update_hook function will be called when database performs INSERT,UPDATE or DELETE operation. The follow C code is record data and change metadata:

```
memset( sql, 0, 1024 ):
if( modifyType==SQLITE_UPDAIE || modifyType==SQLITE—INSERT )
  sprintf( sql, "INSERT INTO sync_record (modify_type, db_pame, table_name, uuid, modlfy_rowid, modify_time) VA.LUES (%d, '%s', '%s', '%s', %d, strftime('%%s', 'now'))", modifyType, dbName, tableName, uuid, rowid);
Else if( modifyType == SQLITE_DELETE )
    sprintf( sql, "UPDATE sync _record SET modify_type=9, modify_time=strffime('%%s', 'now') WHERE modify_rowid=%d", rowid);
```

By SQLite3 callback function, information of data changes can be recorded in the metadata table, as synchronization server updates basis.

3.2. Metadata Storage Modules

Synchronize information metadata needed for data synchronization is stored in the same database in the form of table too [13]. At present, there are mainly three tables (when upgraded version number and format of the table may change in the future).

Table sync_record: The table mainly records the data modified information in the user table, used to determine the user table that data synchronization is required at the time of transfer to the server.

This part of the realization of C language code as follows:

```
Int SQLite3_sync_config(SQLite3* conn){
    SQLite3_exee(conn, "CREATE TABLE IF NOT EXISTS sync_record(modify_rowid INTEGER, uuid TEXT, modify_type INT, modify_time INTEGER, table_name TEXT, db_name Text);", NULL, NULL, NULL);
```

If there is no record tables, establish information table of record changed:

As the same way of another two tables:

```
char** result;
int r, c;
SQLite3_get_table(conn, "SELECT * FROM sync_configure", &result, &r, &c, NULL);
if(r<l)
    SQLite3_exec(conn,"INSERT INTO sync_configure (login_narne, login_pass, sync_url, fite_condition ) VALUES ('green','testPwd', 'http://localhost:8080/resources/helloworld', 'l=1' );", NIJLL, NULL, NIJLL);
    SQLite3_free_table(result):
}
```

SQLite3_sync_configure function initials the synchronization information metadata, and creates table storing synchronization information.

At center database, synchronization server creates corresponding metadata table. JAVA code as follows: *Conn = ds.getConnection();*

```
Statement stmt = conn.createStatement();
Sql = "CREATE TABLE IF NOT EXISTS sync_record(uuid TEXT, table_name TEXT, modify_type INTEGER, sync_time BIGINT)";
stmt.execute(sql):
```

Create this table, synchronization metadata of each synchronization client is stored in the sync_record table.

3.3. Connection Center Database Module of Synchronization Server

Synchronization server needs to connect with the central server to the client and synchronized to send and receive data center servers [14]. The system uses the Tomcat server. Configure Tomcat connection pool is in context.xml file.

```
<Context>
<Resource name= "jdbc/mysql"
    Auth="Container"
    Type="javax.sql.Datasouree"
    driverClassName="com.mysql.jdbe.Driver"
    url="jdbe:mysql://localhost/ttt"
    username="root"
    password=""
    maxActive="100"
    maxldle="10"
    maWait="10000"/>
</Context>
```

After configuring the data source by the following code can obtain the connecting to the database, and execute SQL statements.

```
InitialContext ctx=new InitialContext();
ds=(Datasoure) cts.lookup("java:comp/env/jdbc/msql");
conn=ds.getConnection();
```

3.4. Definition Transfer Format of Synchronous Data

After comparing XML, JSON, binary format, JSON format is a more appropriate format, because its less consump-

tion of both broadband and analytical resources, but very easy to read, useful for debugging. For synchronous data transfer of this system, do a certain format design. JAVA code used to represent a packet is as follows:

```
Public class PacketClass{
  // Name where the data needs to be synchronized table
  public String table_name;
  // The type of data synchronization, INSERT, UPDATE, SELECT
  public int modify_type;
  // The packet data synchronization field number
  public int column_num;
  // The == number of data rows of packet contains
     public int row_num;
     // The name set of the fields
     public String[] columns;
     // The Set of data sets
     public List<string[]> rows:
}
```

Converted into JSON format, for example, *{"table_name" : "table1", "modify_type" : 18, "column_num":4, "row_num":1, "columns": ["UUID", "LASTMODIFY", "COLI", "COL2"], "rows": [["2d36f4cf-ldb2-4f67-9029-affa6470376b", "1240037551", "hello everyone!", "123"]]}*

From this data, we can see JSON meaning it represents. The table to be synchronized of this data packet is table 1. The type of data synchronized is UPDATE (18 is SQLITE_UPDATE constants value SQLITE defined). The data has four fields, including a line of data to be synchronized, and the latter part is the value of the data. The sending and receiving ends following this format can pack and unpack the data, and finally stored in the database.

3.5. Data Derails and Synchronous

Data from the terminal database to the central database, experienced three steps: First, remove the data from the database and packaged as a data packet; Second, send data packets to the synchronization server; Third, the synchronization server unpacked and converted to SQL statements executed in order to write data to a central database. Similar process of data from the central database to the terminal database. The first two steps of the code below:

Read data from the database:

```
if( modify_type==SQLITE_INSERT || modify_type==SQLITE_UPDATE )
  sprintf(sql, "SELECT * FROM table1 where rowed in(
     select modify_rowid from sync_record where modify_type=%d)", modify_type );
else
     sprintf( sq1,"SELECT uuid, modify_time from sync_record where modify_type=%d", modlfy_type);
```

Packaged as JSON format:

```
for( i=0 ;i<columu_num; ++i ){
    json_object_array_add(column_obj,
    json_object_new_string(result[i])):
}
for( i=0: i<row_num; ++i ){
      row=json_object_new_array();
      for( j=0; j<column_num: ++j ){
           prinif("%s\t", result[i*column_num+column_num+j] );
           json_object_array_add( row,
      json_object_new_string( result[i*column_num+column_num
    +j]) );
    }
      Json_object_array_add( row_pbj, row );
      printf("\n");
}
json_object_object_add( full_obj, "table_name", table_name_obj ):
json_object_object_add( full_obj, "modify_type", json_object_new_int( modify_type) );
json_object_object_add( full_obj, "column_num", json_object_new_int(column_num) );
json_object_object_add( full_obj, "row_num", json_object_new_int(row_num) );
json_object_object_add( full_obj, "columns", column_obj );
json_object_object_add( full_obj, "rows", row_obj ):
```

After the data is packed as a sequence of characters sent to the synchronization server *via* HTTP

```
curl_easy_setopt( curl, CURLOPT_UPLOAD,1 );
res= curl_easy_perform(curl):
```

After synchronization server receives the data, unpack and convert SQL execution.

```
@PUT
@Consumes("applicatiom/json")
Public Response Put(String content) throws SQLException{
     Gson gson = new Gson():
     Gson outjson = newGson();
     String sql:
     PacketClass packet = gson.from.Json(content, Paeket-Class. class);
}
```

Tags @PUT and @Consumes("application/json") of the method mean that the method is the function responding PUT method of HTTP, the format of the data received in JSON format. Parameters *String content* is the content of the incoming JSON data.

JSON contents of the packet can be converted into JAVA class instance. Packet class is the content of the data to be synchronized. By the data for this instance, the combination

generate a complete SQL statement execution to the central database.

3.6. Return News Data of Center Database

When the synchronization client sends data to upload, begin to download data. Synchronization client transmits the time of complete synchronization of data last time. The server to determine which data should be sent to the synchronization client based on this time. Here a little confusing that the data last updated time and last synchronization time. Here determination is based on the synchronization time rather than the updating time. Suppose A, B are synchronous terminal, C is a synchronization server, there is a data D, last updated date is on the 10^{th}, the last synchronization date is 10^{th}. That is, the data on the terminal A is modified on the 10^{th}, but in the 12^{th} A to C was a process of synchronizing the changes applied to the central database. if the last update time of B and C is 9^{th}, then two times are greater than the 9^{th}. This data is in the A and C synchronized set, but if the last update date to the 11^{th}, the 10^{th} is less than 11. If you compare the synchronization time and update time, which is not to be counted as data synchronization data collection. And in fact, in which data should be synchronized list because data B is also no change in the data. Comparison should be updated synchronization time instead of the data.

3.7. Security Module

In synchronous systems, only the part database of the system node can access and upload their data to a central database. To ensure the security of the system, it is necessary to authenticate nodes. HTTP protocol defines the part of the certification. Tomcat implements the HTTP protocol as a WEB server also has part of the function with the agreement on certification. In synchronous system, the node having synchronization function needs through the authentication, in order to exchange data with a central server. Tomcat and CURL both flexible support authentications. Secure transmission to achieve the purpose, the client and server are set to the same authentication configuration, you can achieve security management based on the HTTP protocol.

On the server side, according to the manual Tomcat, Tomcat security configuration is as follows:

In conf/server.xml file,

<Realm className="...class name for this implementation"
...other attributes for this implementation.../>

This label indicates the function of Realm module. Realm module corresponds to the "data collection" which store user name and password which can access web content. This "data collection" may be a relational database; it could be an xml file or any other format, because you can custom authentication module. Achieve certification logic modules are all implemented. org.apache.catalina.Realm interface. Tomcat has 5 Realm modules:

JDBCRealm,

DataSoureeRealm,

JNDIRealm,

MemoryRealm,

JAASRealm

If a user name and password are stored in a database, you can use either of the first two. If stored in the Tomcat configuration xml file, using MemoryRealm. If a user name and password storage areas do not meet the native module, the module can be customized to achieve.

The system uses JDBCRealin. When there are a large number of users, the general systems use a database to store the user name and password. Modify cof / server.xml, add Realm labels:

<Realm className="org.apache.eatalina.realm.JBCRealm"
debug="99"
driverName="org.git.mm.mysql.Driver"
conneetion-URL="jdbe:mysql://loealhost/authority?user=dbuser&password=dbpass"
userTable="users" userNameCol="user_name" user-CredCol="user_pass"
userRoleTable="user_roles" roleNameCol="rolename"/>

On the client, according to a document Curl parameter sends the user name and password to be set are:

CURLOP_HTTPAUTH,CURLOPT_USERPWD.

The user name and password of Clients used to authenticate are stored in this synchronization system metadata, stored in sync_confg table. First read the data in the table, and then sent through verification by curl. Code is as follows:

```
char* sql = malloe(512);
memset(sql, 0, 1024):
sprintf(sql, "SELECT login_name, login_pass, sync_url, fiter_condition FROM sync_configure"):
SQLite3_get_table(db, sql, &result, &row, &col, NULL);
// Obtain user name and password from the configuration table to synchronize access to the server's required
strcpy( usemame, result[4] ):
strcpy( password, result[5] ):
strcpy( sync_url, result[6] );
sprintf( userpwd, "%s:%s", usemame, password ):
// Format of the user name and password as defined format of the http protocol
curl_easy_setopt (curl, CURLOPT_HTTRAUTH, CURLAUTH_BASIC)
curl_easy_setopt ( curl, CURI OPT_USERPWD, userpwd):
// Setting options CURL
```

Sync client and synchronization server according to the HTTP protocol defined after a good setup and programming, you can use the HTTP protocol defines the security section to achieve a synchronization system security definitions. There are several safe ways HTTP protocol, implemented here is BASIC way. If necessary, set up at both ends of the other agreement on it.

Table sync_record: The table mainly records the data modified information in the user table, used to determine the

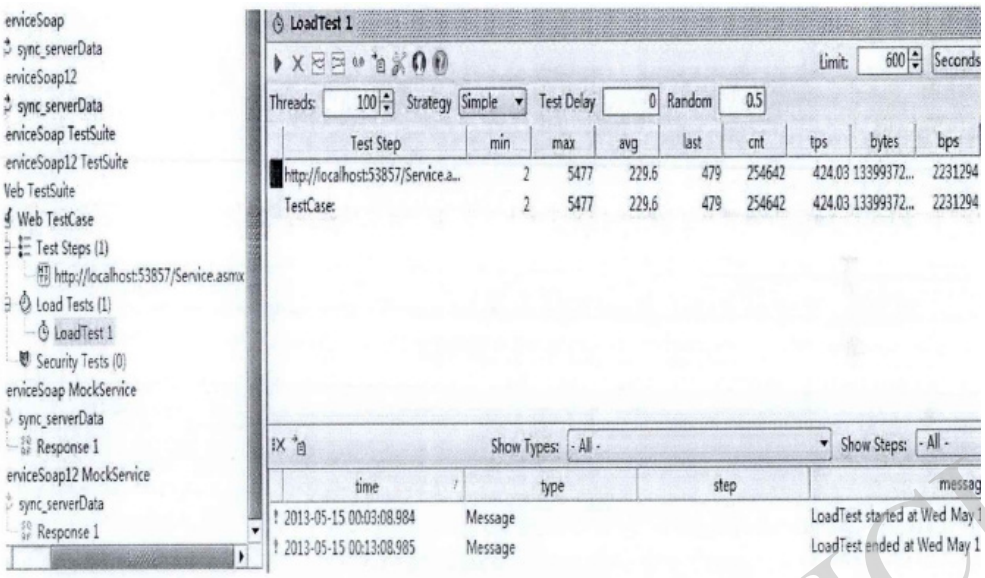

Fig. (1). The load test results.

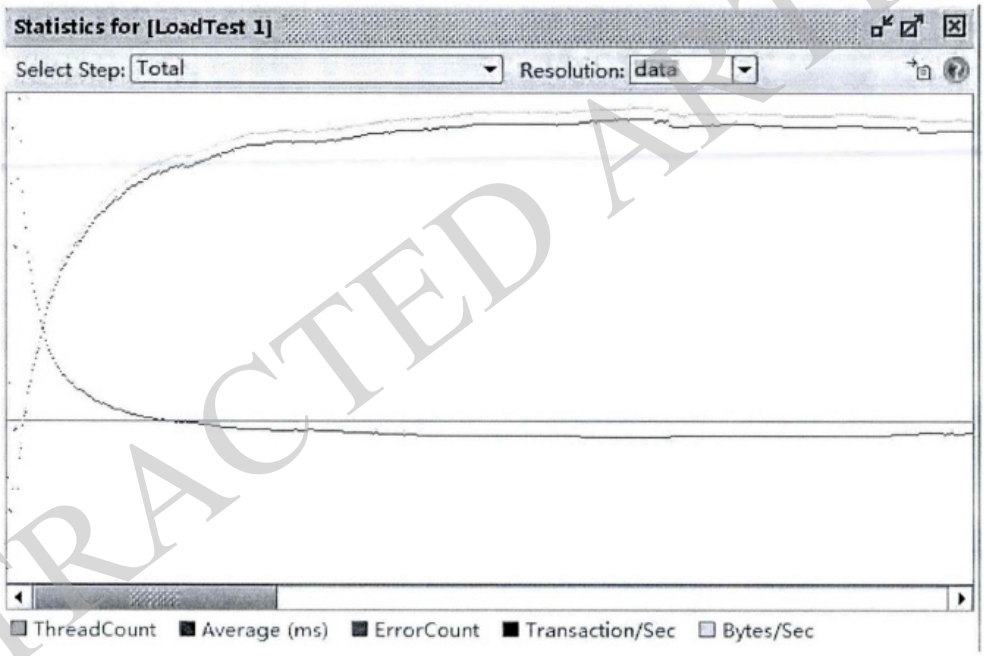

Fig. (2). The load test curve.

user table that data synchronization is required at the time of transfer to the server.

4. SYSTEM PERFORMANCE TESTING

To test the performance of the system, we use SOAPUI to test interfaces' performance. Fig. (**1**) is a diagram of the load test results WSDL service provided by the server. Fig. (**2**) is test data graph.

See throughput (tps), the corresponding time, the data transfer rate information from the graph. Response time is the response total time to spend on n requests. Through testing, found that with the increase in the number of concurrent threads, the average response time of the system will be more balanced, but also greatly enhance the speed of bytes transferred, and tend to be stable. Therefore, the server-side data synchronization interface WebService developed to meet the system requirements, data synchronization.

While the system handles concurrent requests using a multi-threaded distributed and dynamic multi-granularity locking mechanism for handling, made system performance testing. Fig. (**3**) is a multi-granularity locking and performance testing multi-threaded post. It can be found that the use of this strategy for dealing with the distribution request, can make data transmission efficiency in equilibrium, can quickly handle multiple concurrent requests terminal, transmission efficiency has been greatly improved. But the error rate will be increased as the number of concurrent requests thread, also increased.

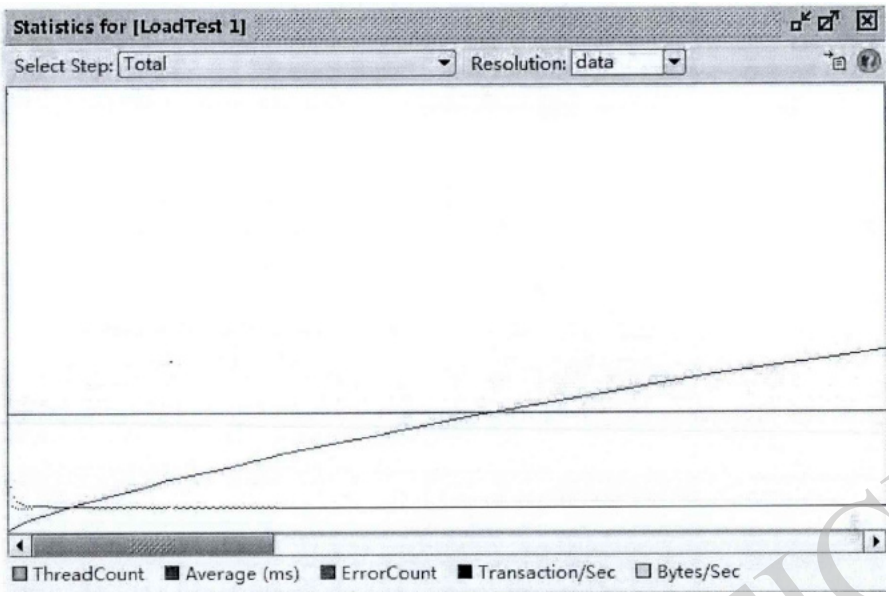

Fig. (3). Multi-granularity locking and multi-threaded performance test.

CONCLUSION

This paper implements a database synchronization system for the SQLite. For some of the problems encountered in the design and development of systems, by looking for literature, guessing and analyzing implementation methods of these similar systems, has solved some key issues and design and implement our own solutions. Mainly UUID replaces integer data format as data representation; Design table to store synchronization information metadata; Use HTTP protocol to transmit data and resolve security authentication and encrypted transmission; Use JSON format as the serial data format on the network transmission; Use JZEE platform to process system concurrent.

This system has the basic functions for synchronizing SQLite, could be regarded as a database synchronization system prototype. Applicable synchronization topology is the scenarios of a center for multiple endpoints. This system is just a prototype. To become a commercial-grade system, it needs a visual management system and a good caching system and caching algorithms developed based on system characteristics.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] D. Chen, X.D. Han, and W. Wang, "Use of SQLite on embedded system", In: *Proceedings of 2010 International Conference on Intelligent Computing and Cognitive Informatics (ICICCI2010)*, 2010, pp. 210-213.

[2] H. Zheng, and B. Zeng, "Design and Implementation of ARM-based mobile hand held terminal for community medical", *Applied Mechanics and Materials*, vol. 263-266, pp. 1623-1628, 2012.

[3] S. Agarwal, D. Starobinski, and A. Trachtenberg, "*Fast PDA Synchronization Using Characteristic Polynomial Interpolation*", Boston University, IEEE, 2002.

[4] S.Y. Lee, "Synchronizing techniques for mobile DBMSs", *Database Research*, vol. 17, no. 3, pp. 29-41, 2001.

[5] S.W. Kim, "Synchronicity nation in an Ivmbcddcd DBMS Environment", In: *IJCSNS International Journal of Computer Science and Network Security*, 2006.

[6] Oracle Corporation, *Oracle Database Lite: Synchronizing Data Between Device and Oracle Database.2007*APaehe Activemq, http://activemq.apache.org.2011.

[7] A. KuPsys, and R. Ekwall, "Aiehiteetural issues of JMS compliant group communication", *4th IEEE International Symposium on Network Computing and Application*, 2005, pp. 139-148.

[8] P. Wang, "Research on the embedded system teaching", In: *International Workshop on Education Technology and Training*, 2008.

[9] W. Xia, "A distributed database management system for college teaching", *Computer Engineering and Design*, vol. 30, no. 23, pp. 5325-5328, 2009..

[10] G. Cheng, and M. Z. Li, "The application of database technology in network ma-nagement system", *Applied Mechanics and Materials*, vols. 644-650, pp. 2850-2853, 2014.

[11] P. Marwedel, "Embedded System Design," Springer, New York, 2011, pp. 20-30.

[12] D. Luo, "Optimization in Cognitive Radio Network", *Wireless Communication Technology*, vol. 22, no. 2, pp. 17-21, 2013.

[13] P. Zhang, Q. He, Z.Y. Feng, and Q.X. Zhang, "Reconfiguration decision making in cognitive wireless network", *Chinese Science Bulletin*, no. 28, pp. 29-31, 2012.

[14] N. Zhao, S. Li, and Z. Wu, "Cognitive radio engine design based on ant colon optimization", *Wireless Personal Communications*, no. 1, pp. 44-48, 2012.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015