6-1 即时通讯代码测试和性能测试

零声学院 https://0voice.ke.qq.com

讲师 Darren老师 QQ326873713

班主任 柚子老师 QQ2690491738

2022年06月30日

重点内容：

- 掌握0voice_im/server/src/test测试代码，有些朋友是mac系统的，也可以通过该测试代码实现聊天；
- 数据库 性能测试，测试即时通讯的性能瓶颈
- 怎么加代码

网页版本：https://www.yuque.com/docs/share/e3c3729c-4263-4d23-a485-df6aa7074c6c?#
《6-1 即时通讯代码测试和性能测试》

1 命令行测试代码

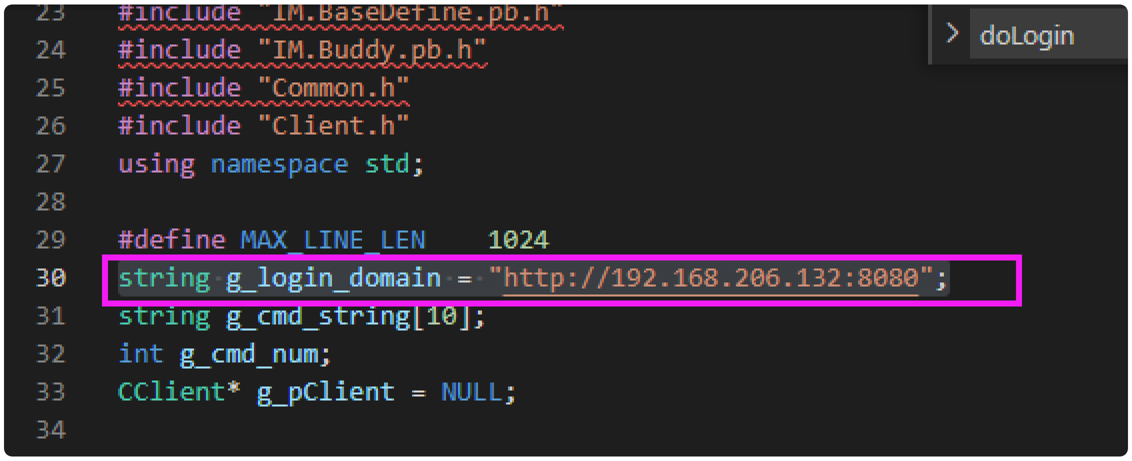
1.1 源码位置

在源码目录的位置：0voice_im/server/src/test

1.2 修改LoginServer地址

修改为自己loginserver的地址

test_client.cpp的`string g_login_domain = "http://192.168.206.132:8080";`



```
#include "IM.BaseDefine.pb.h"
#include "IM.Buddy.pb.h"
#include "Common.h"
#include "Client.h"
using namespace std;

#define MAX_LINE_LEN    1024
string g_login_domain = "http://192.168.206.132:8080";
string g_cmd_string[10];
int g_cmd_num;
CClient* g_pClient = NULL;
```

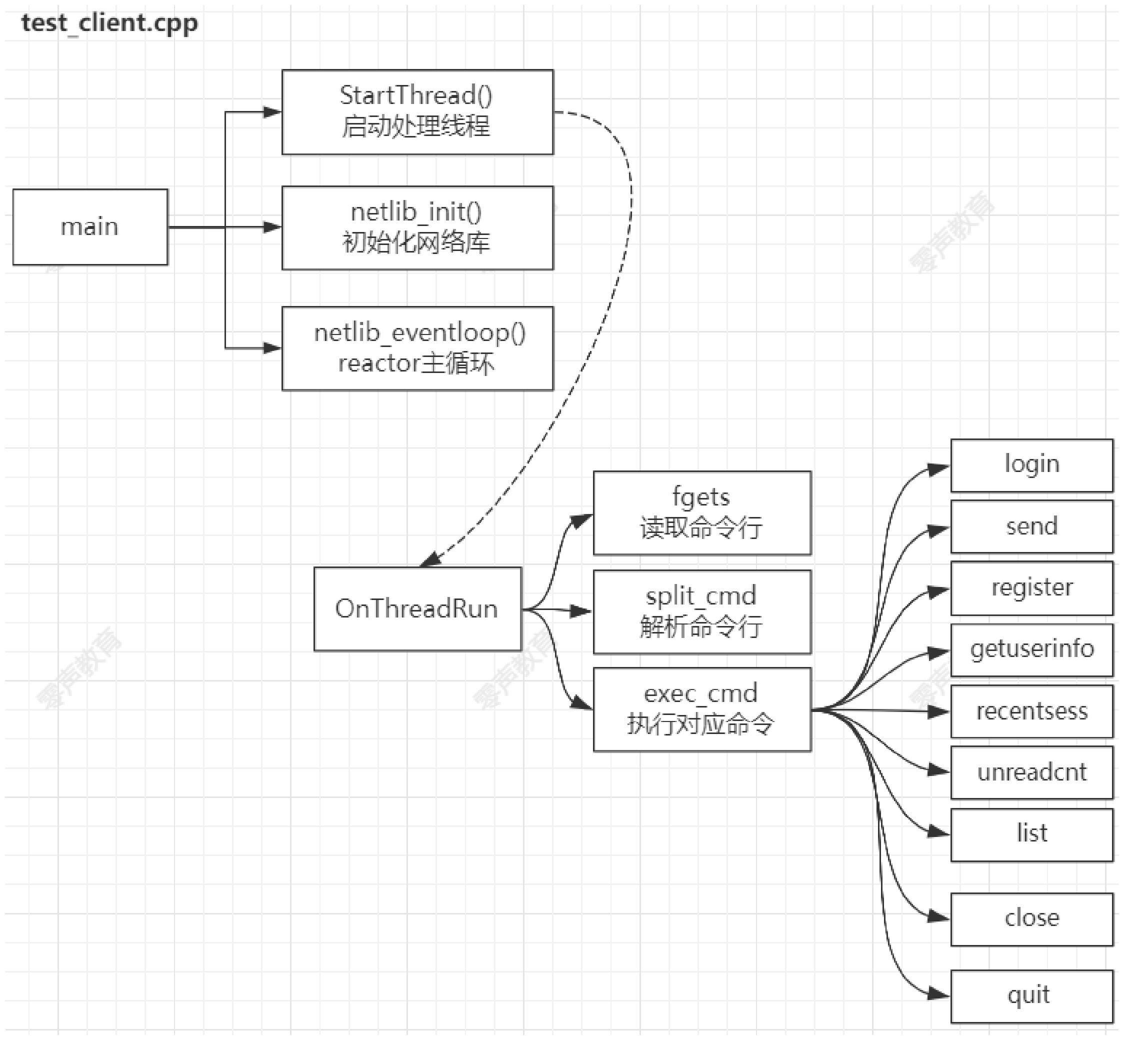
1.3 编译

使用make

```
make
```

在当前目录生成test执行文件。

1.4 测试



执行./test

> 如果报错
> ./test: error while loading shared libraries: libslog.so: cannot open shared object file: No such file or directory

添加库连接

```
#  cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
// 把自己的的0voice_im对应的test路径加入到  /etc/ld.so.conf
#  sudo echo "/home/lqf/im/0voice_im/server/src/test" >> /etc/ld.so.conf
#  sudo  ldconfig
```

1.4.1 登录

1. 运行程序

```
lqf@ubuntu:~/im/0voice_im/server/src/test$ ./test
im-client>
```

2. 登录账号，login为命令，隔壁老王为账号，123456为密码

```
im-client> login 隔壁老王 123456
```

```
CClient::connect {
   "backupIP" : "192.168.206.132",
   "code" : 0,
   "discovery" : "http://127.0.0.1/api/discovery",
   "msfsBackup" : "http://192.168.206.132:8700/",
   "msfsPrior" : "http://192.168.206.132:8700/",
   "msg" : "",
   "port" : "8000",
   "priorIP" : "192.168.206.132"
}
```

1.4.2 获取个人信息

```
im-client> getuserinfo 1
1
im-client> onGetUserInfo  onGetUserInfo
name:廖庆富 nick:darren

im-client> getuserinfo 2
2
im-client> onGetUserInfo  onGetUserInfo
name:谢帆 nick:mark

im-client> getuserinfo 3
3
```

im-client> onGetUserInfo onGetUserInfo
name:隔壁老王 nick:king

1.4.3 发送信息

```
enum MsgType {
  MSG_TYPE_SINGLE_TEXT = 1,
  MSG_TYPE_SINGLE_AUDIO = 2,
  MSG_TYPE_GROUP_TEXT = 17,
  MSG_TYPE_GROUP_AUDIO = 18
};
```

格式 send **toid** msgtype msg

send **3 1 今天晚上上课**

2 测试IM数据库

2.1 代码路径

1. **创建数据库**

进入 0voice_im/auto_setup/**test_mysql**
然后执行脚本**新创建**数据库
sudo ./setup.sh install

2. **测试源码目录**

进入0voice_im/server/src/test_db_proxy
编译

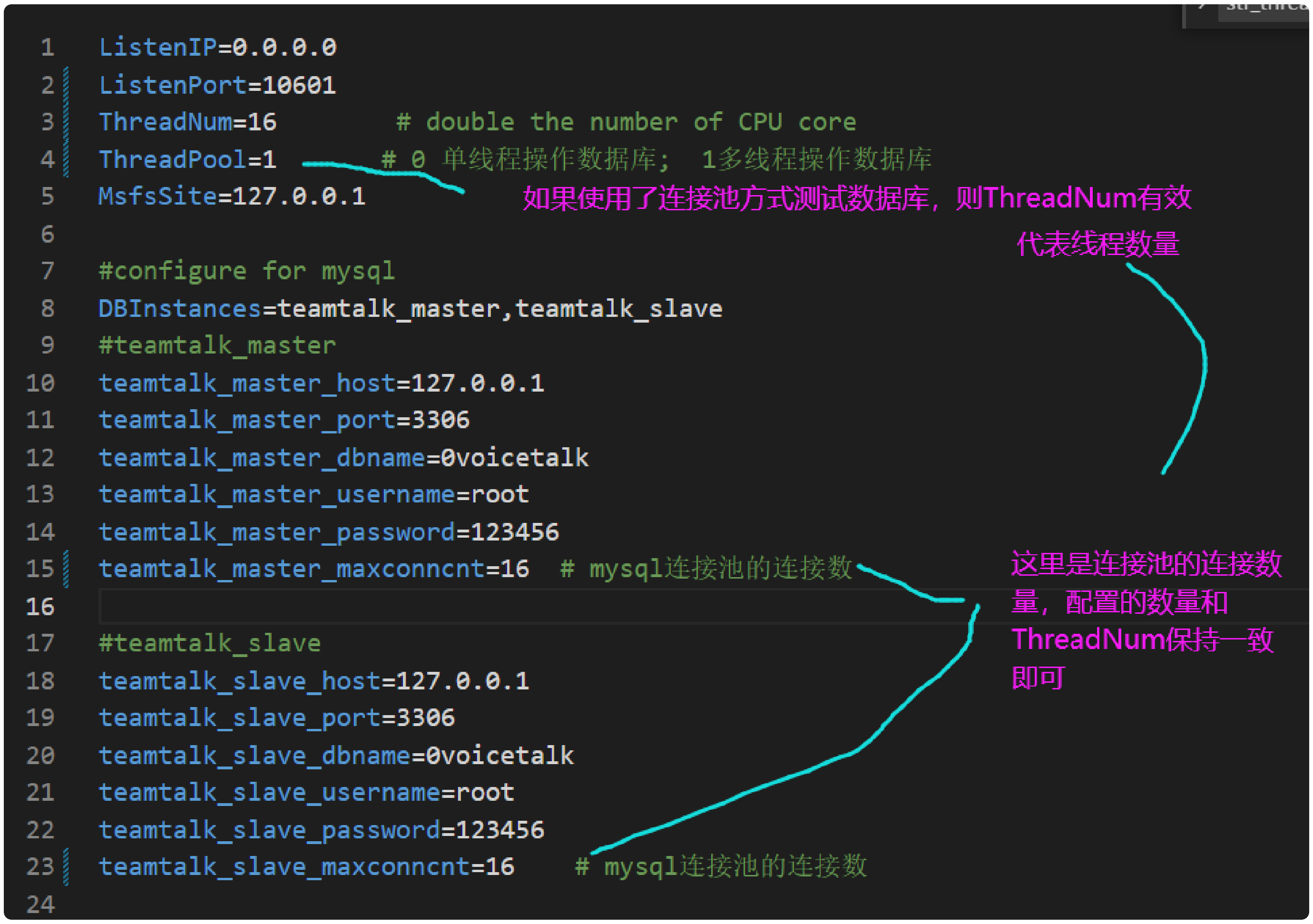
```Go
cmake .
make
```

2.2 测试

- 账号注册
- 账号登录
- 发送消息
- 获取最后一条消息

单线程测试，多线程测试。

配置信息



```
ListenIP=0.0.0.0
ListenPort=10601
ThreadNum=16         # double the number of CPU core
ThreadPool=1          # 0 单线程操作数据库;  1多线程操作数据库
MsfsSite=127.0.0.1

#configure for mysql
DBInstances=teamtalk_master,teamtalk_slave
#teamtalk_master
teamtalk_master_host=127.0.0.1
teamtalk_master_port=3306
teamtalk_master_dbname=0voicetalk
teamtalk_master_username=root
teamtalk_master_password=123456
teamtalk_master_maxconncnt=16  # mysql连接池的连接数

#teamtalk_slave
teamtalk_slave_host=127.0.0.1
teamtalk_slave_port=3306
teamtalk_slave_dbname=0voicetalk
teamtalk_slave_username=root
teamtalk_slave_password=123456
teamtalk_slave_maxconncnt=16    # mysql连接池的连接数
```

如果使用了连接池方式测试数据库，则ThreadNum有效

代表线程数量

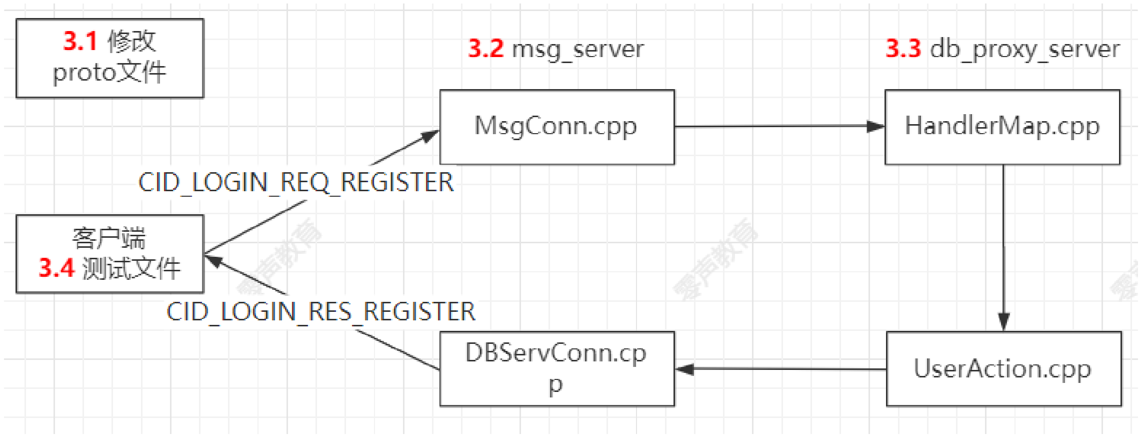
这里是连接池的连接数量，配置的数量和ThreadNum保持一致即可

测试项目	1线程 10000次	16线程 10000次	4线程	8线程
注册账号	83186ms/8.3ms	8628ms/0.83ms	28255ms/2.8ms	
发送消息	355260ms/35.5ms	28531ms/2.8ms		
获取最后一条消息	4742ms/0.47ms	5065ms/0.5ms		
账号登录	2356ms/0.23ms	2283ms/0.22ms		

不同操作涉及的数据库表单

1. 注册账号：写入IMUser
2. 账号登录：读取IMUser
3. 发送消息：读取2个IMRecentSession，写入IMMessage、写入2个IMRecentSession
4. 获取最后一条消息：IMMessage

3 怎么加代码



3.1 修改proto文件

记得编译

IM.BaseDefine.proto

```
enum LoginCmdID{
```

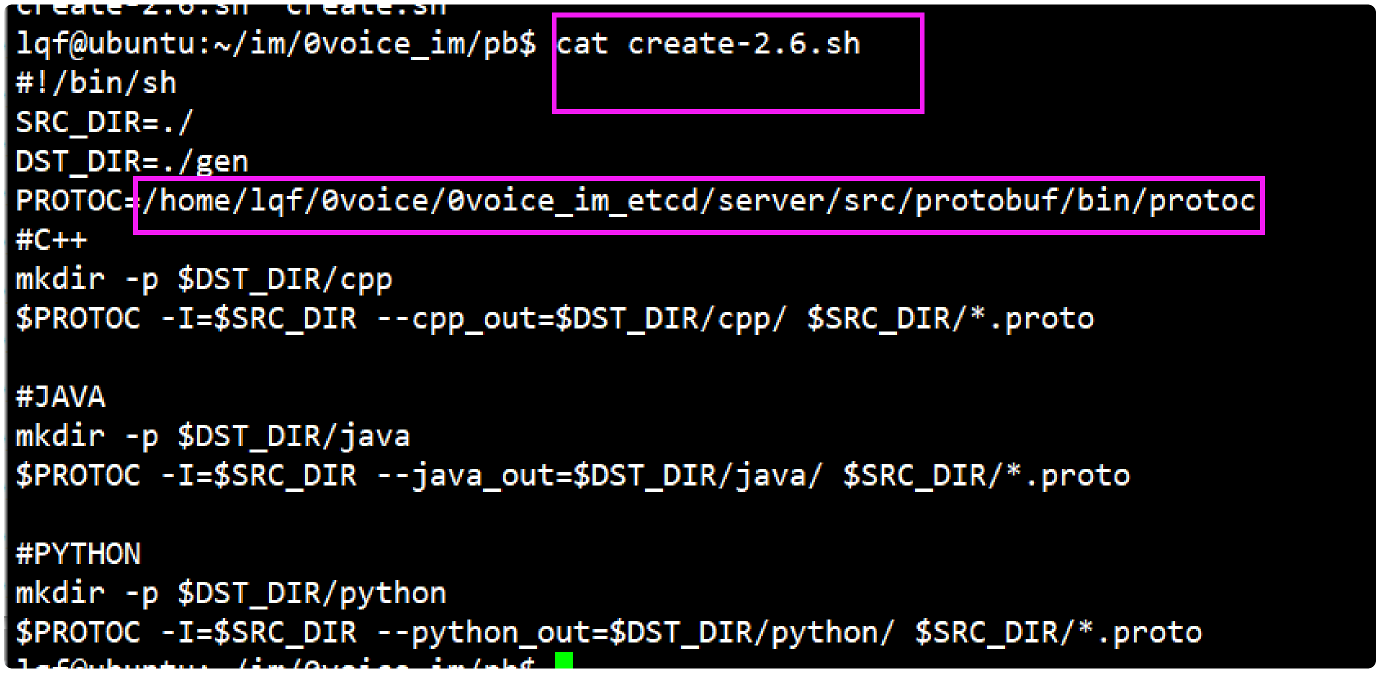
```
CID_LOGIN_REQ_MSGSERVER        = 0x0101;       //
CID_LOGIN_RES_MSGSERVER        = 0x0102;  //
CID_LOGIN_REQ_USERLOGIN        = 0x0103;  //
CID_LOGIN_RES_USERLOGIN        = 0x0104;  //
CID_LOGIN_REQ_LOGINOUT         = 0x0105;//
CID_LOGIN_RES_LOGINOUT         = 0x0106; //
CID_LOGIN_KICK_USER                 = 0x0107;     //
CID_LOGIN_REQ_DEVICETOKEN      = 0x0108;  //
CID_LOGIN_RES_DEVICETOKEN      = 0x0109;  //
CID_LOGIN_REQ_KICKPCCLIENT     = 0x010a;
CID_LOGIN_RES_KICKPCCLIENT     = 0x010b;
CID_LOGIN_REQ_PUSH_SHIELD      = 0x010c;  //勿扰
CID_LOGIN_RES_PUSH_SHIELD      = 0x010d;   //
CID_LOGIN_REQ_QUERY_PUSH_SHIELD = 0x010e; //
CID_LOGIN_RES_QUERY_PUSH_SHIELD = 0x010f;
 CID_LOGIN_REQ_REGISTER = 0x0110;     // 注册新用户
CID_LOGIN_RES_REGISTER = 0x0111;
}
```

IM.Login.proto

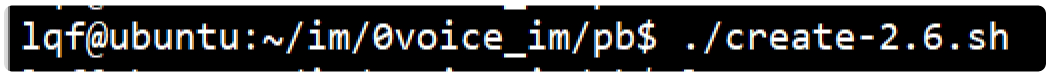
添加请求和响应的对象

Protobuf 复制代码

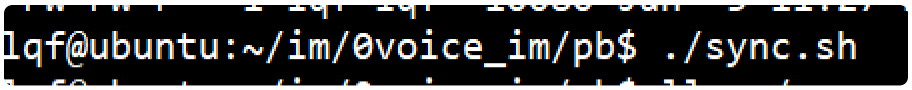
```protobuf
message IMRegisterReq{
//cmd id:        0x0110
required string user_name = 1; // 用户名须唯一，可以是邮箱，电话等
required string password = 2;  // 独立模式、插件模式填app授权码+user_name的md5; 托管模式填用户密码的md5。app授权码由后台分配
optional uint32 sex = 3;   // 1:男; 2:女; 0:未知
optional string nick = 4;   // 昵称
optional string avatar = 5;    // 头像
optional string phone = 6;   // 手机
optional string email = 7;      // 邮箱
}

message IMRegisterRes{
//cmd id:        0x0111
required string user_name = 1;
required IM.BaseDefine.ResultType result_code = 2;
optional string result_string = 3;
optional uint32 user_id = 4;
optional bytes attach_data = 20;
}
```



```
create-2.6.sh  create.sh
lqf@ubuntu:~/im/0voice_im/pb$ cat create-2.6.sh
#!/bin/sh
SRC_DIR=./
DST_DIR=./gen
PROTOC=/home/lqf/0voice/0voice_im_etcd/server/src/protobuf/bin/protoc
#C++
mkdir -p $DST_DIR/cpp
$PROTOC -I=$SRC_DIR --cpp_out=$DST_DIR/cpp/ $SRC_DIR/*.proto

#JAVA
mkdir -p $DST_DIR/java
$PROTOC -I=$SRC_DIR --java_out=$DST_DIR/java/ $SRC_DIR/*.proto

#PYTHON
mkdir -p $DST_DIR/python
$PROTOC -I=$SRC_DIR --python_out=$DST_DIR/python/ $SRC_DIR/*.proto
lqf@ubuntu:~/im/0voice_im/pb$
```



```
lqf@ubuntu:~/im/0voice_im/pb$ ./create-2.6.sh
```



```
lqf@ubuntu:~/im/0voice_im/pb$ ./sync.sh
```

3.2 msg_server

MsgConn.cpp

C++ 复制代码

```cpp
void CMsgConn::HandlePdu(CImPdu *pPdu)
{
    switch (pPdu->GetCommandId())
    {
    case CID_LOGIN_REQ_REGISTER:
        _HandleRegistRequest(pPdu);
        break;
    }
    ....
}
void CMsgConn::_HandleRegisterRequest(CImPdu *pPdu)
{
    // refuse second regist request
    uint64_t cur_time = get_tick_count();
    if (m_regist_time > cur_time - 5000)
    { // 5秒内不能重复注册
        log_warn("duplicate RegistRequest in the same conn in 5 seconds");
        return;
    }

    uint32_t result = 0;
    string result_string = "";
    IM::Login::IMRegistReq msg;
    CHECK_PB_PARSE_MSG(msg.ParseFromArray(pPdu->GetBodyData(), pPdu->GetBodyLength()));
    log("app_id=%u, user_id=%u, user_name=%s, password=%s, sex=%u, nick=%s, avatar=%s, phone=%s, email=%s, company=%s, address=%s", msg.app_id(), msg.user_id(), msg.user_name().c_str(), msg.password().c_str(), msg.sex(), msg.nick().c_str(), msg.avatar().c_str(), msg.phone().c_str(), msg.email().c_str(), msg.company().c_str(), msg.address().c_str());
    if (msg.app_id() < APP_TYPE_ZERO_VOICE || msg.user_name().size() < 2 || msg.password().size() < 6)
    {
        result = IM::BaseDefine::RESULT_PARAM_ERROR;
        result_string = "parameter error";
    }

    CDBServConn *pDbConn = get_db_serv_conn_for_login();
    if (0 == result)
    {
        // check if db server connection is OK
        if (!pDbConn)
        {
            result = IM::BaseDefine::RESULT_PARAM_ERROR;
            result_string = "server exception";
        }
    }

    if (result)
    {
        log_error("app_id=%u, user_id=%u, user_name=%s, result=%d, result_string=%s", msg.app_id(), msg.user_id(), msg.user_name().c_str(), result, result_string.c_str());
        IM::Login::IMRegistRes msg;
        msg.set_result_code((IM::BaseDefine::ResultType)result);
        msg.set_result_string(result_string);
        CImPdu pdu;
        pdu.SetPBMsg(&msg);
        pdu.SetFlag(pPdu->GetFlag());
        pdu.SetServiceId(SID_LOGIN);
        pdu.SetCommandId(CID_LOGIN_RES_REGISTER);
        pdu.SetSeqNum(pPdu->GetSeqNum());
        SendPdu(&pdu);
        Close();
        return;
    }

    CImUser *pImUser = CImUserManager::GetInstance()->GetImUserByLoginName(msg.app_id(), msg.user_name());
    if (!pImUser)
    {
        pImUser = new CImUser(msg.user_name());
        CImUserManager::GetInstance()->AddImUserByLoginName(msg.app_id(), msg.user_name(), pImUser);
    }
    pImUser->AddUnValidateMsgConn(this);

    CDbAttachData attach_data(ATTACH_TYPE_HANDLE, m_handle, 0);
    msg.set_attach_data(attach_data.GetBuffer(), attach_data.GetLength());
    CImPdu pdu;
    pdu.SetPBMsg(&msg);
    pdu.SetFlag(pPdu->GetFlag());
    pdu.SetServiceId(SID_LOGIN);
    pdu.SetCommandId(CID_LOGIN_REQ_REGIST);
    pdu.SetSeqNum(pPdu->GetSeqNum());
    pDbConn->SendPdu(&pdu);
}
```

DBServConn.cpp

C++ 复制代码

```cpp
void CDBServConn::HandlePdu(CImPdu* pPdu)
{
    switch (pPdu->GetCommandId()) {
        case CID_LOGIN_RES_REGISTER:
            _HandleRegistResponse(pPdu);
            break;
     ...
    }
}

void CDBServConn::_HandleRegistResponse(CImPdu* pPdu)
{
    IM::Login::IMRegistRes msg;
    CHECK_PB_PARSE_MSG(msg.ParseFromArray(pPdu->GetBodyData(), pPdu->GetBodyLength()));

    uint16_t app_id = pPdu->GetFlag();
    string user_name = msg.user_name();
    uint32_t result = msg.result_code();
    string result_string = msg.result_string();

    CDbAttachData attach_data((uchar_t*)msg.attach_data().c_str(), msg.attach_data().length());
    log("app_id=%d, user_name=%s, result=%d, result_string=%s", app_id, user_name.c_str(), result, result_string.c_str());

    CImUser* pImUser = CImUserManager::GetInstance()->GetImUserByLoginName(app_id, user_name);
    CMsgConn* pMsgConn = NULL;
    if (!pImUser) {
        log_error("ImUser for user not exist, app_id=%u, user_name=%s", app_id, user_name.c_str());
        return;
    }
    else {
        pMsgConn = pImUser->GetUnValidateMsgConn(attach_data.GetHandle());
        if (NULL == pMsgConn) {
            log_error("no such conn, app_id=%u, user_name=%s", app_id, user_name.c_str());
            return;
        }
    }

    // 给用户发送响应
```

```cpp
        msg.clear_attach_data();
        CImPdu pdu2;
        pdu2.SetPBMsg(&msg);
        pdu2.SetServiceId(SID_LOGIN);
        pdu2.SetCommandId(CID_LOGIN_RES_REGIST);
        pdu2.SetSeqNum(pPdu->GetSeqNum());
        pMsgConn->SendPdu(&pdu2);

        if (result != 0) {
            pMsgConn->Close();
        }
    }
```

3.3 db_proxy_server

HandlerMap.cpp

C++ 复制代码

```cpp
m_handler_map.insert(make_pair(uint32_t(CID_LOGIN_REQ_REGISTER), DB_PROXY::registerUser));
```

business/UserAction.cpp

C++ 复制代码

```cpp
void registerUser(CImPdu *pPdu, uint32_t conn_uuid)
{
    IM::Login::IMRegistReq msg;
    IM::Login::IMRegistRes msgResp;
    if (msg.ParseFromArray(pPdu->GetBodyData(), pPdu->GetBodyLength()))
    {
        CImPdu *pPduRes = new CImPdu;
        IM::BaseDefine::ResultType nResultType = IM::BaseDefine::ResultType::REFUSE_REASON_NONE;
        DBUserInfo_t cUser;
        uint32_t nAppId = msg.app_id();

        cUser.nId = msg.user_id();
        cUser.strName = msg.user_name();
        cUser.strPass = msg.password();
        cUser.nSex = msg.sex();
        cUser.strNick = msg.nick();
        cUser.strAvatar = msg.avatar();
        cUser.strTel = msg.phone();
        cUser.strEmail = msg.email();
        cUser.strAddress = msg.address();

        log("appId=%d,name=%s,pass=%s,email=%s,company=%s,address=%s", nAppId, cUser.strName.c_str(), cUser.strPass.c_str(),
       cUser.strEmail.c_str(), cUser.strCompany.c_str(), cUser.strAddress.c_str());

        bool bRet = CUserModel::getInstance()->isUserExist(nAppId, cUser.nId, cUser.strName);
        if (bRet)
        {
            if (0 != cUser.nId)
            {
                DBUserInfo_t oldUser;
                // 当使用指定ID同步账户时，若账户名称和头像不为空，并且与之前保存的不一样，则需要更新
                if (CUserModel::getInstance()->getUser(nAppId, cUser.nId, oldUser))
                {
                    if ((!cUser.strName.empty() && oldUser.strName != cUser.strName) || (!cUser.strAvatar.empty() && oldUser.strAvatar != cUser.strAvatar))
                    {
                        if (!cUser.strName.empty() && oldUser.strName != cUser.strName)
```

```
                            oldUser.strName = cUser.strName;

                        if (!cUser.strAvatar.empty() && oldUser.strAvatar != cUser.strAvatar)
                            oldUser.strAvatar = cUser.strAvatar;

                        log("update newUserName:%s oldUserName:%s newAvatar:%s oldAvatar:%s.", cUser.strName.c_str(),
                            oldUser.strName.c_str(), cUser.strAvatar.c_str(), oldUser.strAvatar.c_str());

                        if (!CUserModel::getInstance()->updateUser(nAppId, oldUser))
                            nResultType = IM::BaseDefine::ResultType::REFUSE_REASON_DB_VALIDATE_FAILED;
                    }
                }
                else
                {
                    log_error("Get user failed, its userID is %d", cUser.nId);
                    nResultType = IM::BaseDefine::ResultType::REFUSE_REASON_DB_VALIDATE_FAILED;
                }
            }
            else
            {
                nResultType = IM::BaseDefine::ResultType::RESULT_REPEAT_OP;
            }
        }
        else
        {
            bRet = CUserModel::getInstance()->insertUser(nAppId, cUser);

            if (bRet)
                nResultType = IM::BaseDefine::ResultType::REFUSE_REASON_NONE;
            else
                nResultType = IM::BaseDefine::ResultType::REFUSE_REASON_DB_VALIDATE_FAILED;
        }

        log("nUserId=%d,userName=%s,nResultType=%d", cUser.nId, msg.user_name().c_str(), nResultType);
        msgResp.set_user_id(cUser.nId);
        msgResp.set_user_name(cUser.strName.c_str());
        msgResp.set_result_code(nResultType);
```

```
            msgResp.set_attach_data(msg.attach_data());
            pPduRes->SetPBMsg(&msgResp);
            pPduRes->SetFlag(nAppId);
            pPduRes->SetSeqNum(pPdu->GetSeqNum());
            pPduRes->SetServiceId(IM::BaseDefine::SID_BUDDY_LIST);
            pPduRes->SetCommandId(IM::BaseDefine::CID_LOGIN_RES_REGISTER); // 返回注册的用户ID
            CProxyConn::AddResponsePdu(conn_uuid, pPduRes);
        }
        else
        {
            log_error("parse pb failed");
        }
    }
```

3.4 测试文件

在test目录下的test_client.c 添加支持注册。