**好友系统设计文档**

**概况：**本好友系统包括添加好友，添加黑名单，删除好友，删除黑名单，向好友发信息，推荐好友几个功能。所有关系均为单向关系，即你的好友的好友列表里未必有你。

**内存结构：**

好友系统结构由friend\_t, black\_t, recent\_t, temp\_t, recommendation\_t 五个struct和一个管理类Friendinfo组成。

friend\_t, black\_t分别存放自己某个好友和某个黑名单中人的详细信息。recent\_t存放自己最近联系人信息，recommendation存放的是向玩家推荐的好友的信息， temp\_t存放一个临时数据，当玩家申请加某人为好友，恰巧该人不在线时，无法立刻推送好友申请消息时，会在该人的数据库中插入一条临时数据，上线时，该临时数据读入temp\_t，并在数据库中删除，然后根据temp\_t中的信息向已上线的玩家推送对方申请好友的消息。

Friendinfo是一个管理类，用它来管理一个玩家的好友，黑名单，临时名单和最近联系人名单。它有四个成员变量，分别是三个map,

std::map<userid\_t, friend\_t> m\_friends;

std::map<userid\_t, black\_t> m\_blacklist;

std::map<userid\_t, temp\_t> m\_templist;

和一个list std::list<recent\_t> l\_recentlist;

其他成员方法均是对这几个数据结构的操作。

**数据库结构：**

玩家登录时，会将数据库的信息读入内存。

数据库中存储了5个字段  
userid 和friendid共同为key, 确定了两个玩家之间会有某种关系（当然未必就是好友）。

后边三个字段确定了关系类型。Is\_friend 为1说明是好友，is\_black为1说明是黑名单中人，is\_recent为1说明在最近联系人列表中，is\_temp为1说明只是临时信息（前边已经介绍）。

其中is\_friend和is\_black互斥，即不能同时具有这两层关系

**协议部分（代码很长，这里只写流程，详情看代码和注释）：**

cs\_0x0601\_refresh\_friend\_data && sc\_0x0601\_refresh\_friend\_data：

刷新好友（包括好友，黑名单，最近联系人，临时列表）列表协议，在上线时和打开好友面板时，前端会发此协议给后台。

后台收到此协议后，从内存中读出好友，黑名单，最近联系人，临时列表。首先将这个id打包协议发向switch读取玩家是否在线信息，并存入内存，再将这些id打包协议发向cache服务器读取其他详细的信息(例如vip等级，战力，性别等)，并回包通知客户端。另外，如果玩家temp\_t列表中存在id，说明在自己下线时有人申请自己为好友，这时需要给该玩家推送一条对方申请自己为好友的消息，并将内存中的temp\_t信息删除 。

对应的switch中拉取信息的逻辑，可以查看switch服务器中的代码。

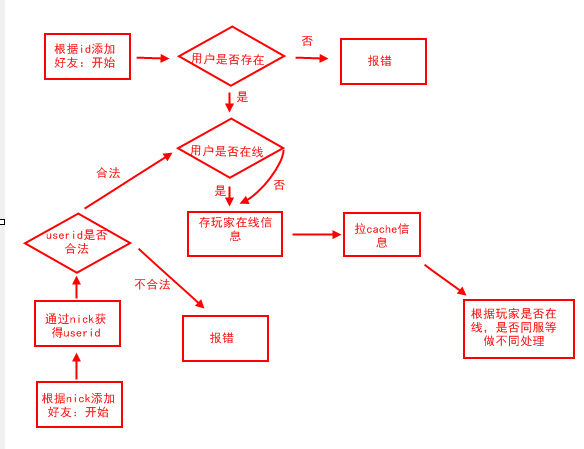
cs\_0x0602\_sync\_friend\_data && sc\_0x0602\_sync\_friend\_data:

同步好友（包括好友，黑名单，最近联系人）列表协议，当有任何改动列表的事件发生时，例如添加好友，删除黑名单等，就将更新后的列表打包到此协议中，返回给客户端

cs\_0x0603\_add\_friend && sc\_0x0603\_add\_friend:

添加好友（逻辑较为复杂，详细看代码和注释）。

流程



拉取到cache后，就可以将对方添加为自己好友（分别同步客户端和db），因为是单向好友，所以不论对方是否同意，自己的好友列表中都会有对方（对方好友列表中未必有自己）。

然后判断要添加玩家是否在线。

如果不在线，就向要添加玩家的数据库写temp临时信息（以自己id为friendid，为对方id为Userid存，因为是存对方的数据库，所以要以被添加玩家的角度存数据）。

如果在线，要判断是否同服。

如果同服，就打包sc\_0x0606\_notify\_add\_friend协议并发给对方，通知自己加了对方好友。

如果不同服，就将sc\_0x0606\_notify\_add\_friend通过switchproto::cs\_sw\_transmit\_only协议，由switch转发给对方。

**另一方面，在对方的online收到该转发包后，不会立刻给客户端，要在对方的online验证对方的黑名单中是否有自己以及对方是否已经加了自己，如果自己在对方黑名单中或者自己已经被对方所加，就丢包，而不把包发给对方。这段逻辑在online下的swich\_proto.cpp中的SwitchTransmitCmdProcessor::proc\_pkg\_from\_serv的//hack注释处**

cs\_0x0604\_send\_personal\_msg &&sc\_0x0604\_send\_personal\_msg:

向好友发聊天消息。

如果自己和对方同服且在线，就直接发消息sc\_0x0605\_notify\_personal\_message给对方。

如果不是上边情况，就从switch拉好友是否在线信息。

如果在线，就通过swtich转发sc\_0x0605\_notify\_personal\_message消息给对方。

上述情况中，

如果对方黑名单中有自己，都要丢包而不发消息给对方。

如果对方好友中没有自己，都会给对方改发好友申请信息，而不是对应的聊天消息。

如果对方不在线，都会直接丢掉包，而不发送任何消息给对方。

**在发消息后，无论是否成功，都要改变对应的recent列表，将发消息的对象放到最近联系人列表的首位**

另外，通过switch转发到对方Online后，不是立刻将包给客户端，像前边一样，也做了处理，这段逻辑也在online下的swich\_proto.cpp的SwitchTransmitCmdProcessor:: proc\_pkg\_from\_serv的//hack注释处

cs\_0x0605\_notify\_personal\_message &&sc\_0x0605\_notify\_personal\_message

后台和前端通知聊天信息使用的协议，在0x0604中用到。

cs\_0x0606\_notify\_add\_friend && sc\_0x0606\_notify\_add\_friend

后台和前端通知添加好友使用的协议，在0x0603中用到。

cs\_0x0607\_add\_blacklist && sc\_0x0607\_add\_blacklist

和0x0603添加好友非常相似，这里不重复了。

**值得一说的是，如果一个玩家被某人拉入黑名单，则会自动从该人的好友列表和最近联系人中移除（如果之前在这两个列表中）；同理，如果一个玩家被某人加入好友名单，也会自动从该人的黑名单中删除（如果之前在这个黑名单中）。**

cs\_0x0608\_remove\_friend && sc\_0x0608\_remove\_friend

删除好友或者黑名单。

根据前端传来的id，删除该好友或黑名单中人。需要同时修改内存，数据库，并通知前端。

**注意：如果删除的是好友，且最近联系人列表中也有该好友时，应该将该好友从最近联系人中也删除掉**

cs\_0x0609\_search\_friend && sc\_0x0609\_search\_friend

查找好友，逻辑和添加好友非常相似。

区别就是在最后拉取完所有信息后，不向对方发添加好友通知，而是将拉取到的好友列表返给客户端。

cs\_0x060F\_recommendation && sc\_0x060F\_recommendation

收到此协议时，首先从swich中获取一部分当前在线玩家的id（目前是获取100个）。然后在把这个id打包入协议，发送给cache拉取他们的详细信息，然后将等级与发送请求玩家最接近的10个用户返回给玩家。

**目前的做法可能存在一个待验证问题，就是switch中，玩家的id是存在map<userid, player\_t>的数据结构中的，userid是key，所以map会默认以userid从小到大的顺序给玩家排序。**

**所以拉取到的玩家会偏向于米米号最小的那批玩家。**

设计初期，本以为swich中米米号的存放，是以登录时间排序的（以为玩家是存在线性结构中）,这样推荐的永远是最新登录的玩家，一时忽略了map数据结构的特性。如果玩家上下线很频繁，则不会发生玩家总是被推荐相同人的情况，如果玩家上下线不频繁，则有可能导致推荐好友集中在米米号较小的玩家。

短期内没想到特别好的解决方案（因为还要同时保证推荐玩家的随机性），在和toby交流中，他说可以用到redis，如果遇到问题，可以与他讨论解决方案。