# 智能家居P2P通信服务器 详细设计

修改记录

版本1.1

4.2.1增加json项gateway

4.3.6增加json项broadcast

5.1.1 删除操作类型5，6。其功能合并到操作类型1

4.1.7 增加群发反馈，群发识别。

4.1.7 现在会反馈一个包头。

4.1.6 删除此项，目前支持多主控端控制同一个设备。

图2.3 新增

图2.1 客户端账号登录改为P2P服务端实现。

4.2.1 增加json项username，password

4.1.1-4.1.3 增加TCP处理情况

## 概述

**术语** 主控端：手机APP，pc等智能机。

被控端：智能家居设备

客户列表：服务端上记录的已登陆用户，包括主控端和被控设备。

## 二、详细流程

为了业务逻辑清晰，以及方便测试，将P2P功能和业务功能分离。P2P服务端只负责协助客户端打洞和维护客户信息列表。业务服务端负责全部数据库操作，账号管理等。

可参考《P2P智能家居设备通信设计.docx》

1.账号登录

服务端

（业务）

服务端

（P2P）

维持心跳

5、获取绑定设备列表

2．身份/权限验证

绑定账号和设备

主控客户端

（ClientA）

7、（首次）请求

连接被控端

3、登入并上报信息

数据库

4．维护客户信息列表

必要时写入数据库

客户列表（表3.1）

clientA/MAC

IP:PORT

6．查询绑定设备

图2.1 主控端与服务端流程关系

维持心跳

5、提供目标地址

命令打洞

被控客户端

（ClientB）

1、登入并上报信息

数据库

服务端

（业务）

服务端

（P2P）

4．维护客户信息列表

必要时写入数据库

2．验证设备id

6、反馈打洞结果

3．查询设备id

图2.2 被控端与服务端流程关系

账号管理，

设备绑定。

维持心跳

客户端

（Client）

账号登录，登出

数据库

服务端

（同步）

服务端

（业务）

同步

同步账号信息，

登录日志信息

WEB端

（HTTP）

业务服务端

服务端

（P2P）

获取绑定关系

验证设备ID

查询账号,设备

在线状态

服务端

图2.3 业务服务端流程关系

## 三、数据结构

表3.1客户端信息列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 |  |  | 说明 |
| strWanIP |  |  |  | 外网IP |
| nWanPort |  |  |  | 外网PORT |
| strMac |  |  |  | MAC |
| strLanIP |  |  |  | 内网IP |
| nLanPort |  |  |  | 内网PORT |
| strLanMask |  |  |  | 子网掩码 |
| strGateWay |  |  |  | 网关地址 |
| bMainCtrl |  |  |  | 是否主控端 |
| tmCreate |  |  |  | 创建时间 |
| tmUpdate |  |  |  | 更新时间 |
| strRemark |  |  |  | 备注 |
| ~~bUsedByOther~~ |  |  |  | ~~是否已被其他主控端占用~~ |
| strDevId |  |  |  | 设备ID |
| nDevType |  |  |  | 设备类型 |

表3.2账号信息

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
|  |  | 账号名 |
|  |  | 密码（MD5） |
|  |  | 绑定设备 |

## 四、基本功能

### 4.1服务器端（P2P）

#### 4.1.1记录客户端登入、登出，维护客户端信息列表。

**UDP**

客户端列表在内存中使用key-value维护，便于查找。有需要时写入数据库。~~Key采用MAC地址（手机号也可）~~，key采用sessionid(64位int递增)，value即为客户信息，见表2.1

数据库可采用Mysql，连接方式选择ADO/ODBC

登入时插入或更新客户端列表，登出时在客户端列表中删除。主控端登入时要验证账号密码。被控端无账号，但可验证设备id或mac地址是否属于我们的产品。

登入成功反馈json部分：

{

“sessionid”:”递增id”,

“guid”:”唯一标识”

}

登出成功不反馈json，只返回包头

**TCP**

客户端列表改为HASHMAP，sessionid作为key

Sessionid不再是递增数字，改为UUID。每次登陆先从数据库进行身份验证（主控端为账号密码，被控端为deviceid），成功后得到这条记录的uuid。

登入成功反馈json部分：

{

“sessionid”:” 唯一标识”,

}

登出成功同上。

#### 4.1.2维持与客户端发来”心跳”

**UDP**

经实际测试，UDP模式下，NAT维护的端口映射在一分钟内无通信就会失效。必须维持一个心跳。暂定20秒。为减少服务端工作，由客户端定时发送。

收到心跳包后，更新客户端列表中的更新时间，并反馈一个包头。

当第一个用户登陆时，服务端发起一个线程用于检测超时用户。

**TCP**

同样需要心跳包，由客户端定时发送。不使用TCP自带的keepalive，详见概要设计——可行性分析

#### 4.1.3为主控端提供被控端（已绑定设备）列表

**UDP**

验证主控端是否已登录。

使用主控端的MAC地址或账号(sessionid)查找已绑定的设备的MAC地址(或device id)，从客户端列表中找到相应的被控端信息，发送给主控端。注意，当绑定设备过多时，可能造成json串过长，导致分包或超出UDP数据包最大值64K，一种方法是限制最大绑定数，如30 。另一种方法是采取分页，添加分页查询接口。或者索性使用TCP方式。

可使用数据库表维护主控端账号，MAC地址与被控端设备的绑定以及信息查询。

JSON部分：

{

“devs”:[

{

“ip”:”IP地址”,

“port”:端口号,

“sessionid”:”session id”,

“deviceid”:”设备id”

“issamelan”:1,

//注释：issamelan为0时，没有下面两个字段

“lanip”:”局域网IP地址”,

“lanport”:局域网端口号

}

]

}

这里只保留必备的基础信息，详细信息查询可另提供服务端，见账号管理部分

**TCP**

首先根据sessionid和SOCKET值判断是否登录

使用账号从数据库查询绑定关系。

查询绑定的设备登录状态，反馈给主控端。

JSON部分：

{

“devs”:[

{

“ip”:”IP地址”,

“port”:端口号,

“sessionid”:”uuid”,

//“deviceid”:”设备id”,

“isupnp”:1,

//注释：isupnp为0时，没有下面两个字段

“upnpip”:”UPNP获取的路由外网IP”,

“upnpport”:UPNP端口号,

“issamelan”:1,

//注释：issamelan为0时，没有下面两个字段

“lanip”:”局域网IP地址”,

“lanport”:局域网端口号

}

]

}

注意：TCP的sessionid，已经改为UUID。

#### 4.1.4接收主控端连接被控端的请求

解析json（见4.2.5）:

1验证主控端身份，是否已登陆。

2验证被控端是否已与主控端绑定。

3验证被控端是否可连通。

4验证被控端是否已被他人占用，以免同时控制产生冲突。

反馈一个包头。不保证打通，主控端可持续请求。

#### 4.1.5命令被控端向主控端打洞

将主控端信息（ip，port等）发给被控端，并命令其向主控端打洞。主控端每请求一次，服务端就向被控端请求一次。双方反馈连接建立则更新被控设备占用标记

Json格式：

{

“mainctrl”:

{

“ip”:”ip”,

“port”:”port”,

“sessionid”:”主控端的session id”

}

}

当主控端与被控端在同一局域网内时，可不进行打洞协助。如何判断在同一路由下？可以通过内网IP + 子网掩码。此时只用提供内网IP和端口即可。

#### ~~4.1.6维护被控端占用状态~~

~~服务器协助打洞时，应设置被控端占用状态，已防其他主控端连接。~~

~~主控端登出时或与服务端心跳失效，服务器清除其绑定的被控端的占用状态。~~

~~主控端与被控端之间心跳失效，均通知服务器清除被控端占用状态。~~

支持多个主控端同时控制一个设备。

#### 4.1.7为客户端间提供中转传输（UDP）

当主控端与被控端同时处于对称型路由下，无法进行P2P传输，只能由服务端中转。

提供一个代理接口。

基于可靠性考虑，可考虑用TCP实现。

服务端解析json（分别见4.2.7和4.3.6），验证源session判断是否登陆，验证目的端是否可达。对proxydata不处理，重新生成json转发给目的端(见4.2.8和4.3.7)。

群发反馈功能：在中转模式下，同步各主控端得到的设备状态。

需要区分主控端，设备，以及设备自身。且只反馈给绑定的各主控端。

需要区分P2P建立连接，中转建立连接的各主控端，以免代理反馈群发时，与被控端直接向P2P连接的主控端群发重复。

回复客户端一个包头

账号管理部分：

1. 账号身份验证
2. 账号申请，找回
3. 权限管理
4. 绑定设备管理

以上红色部分不在P2P服务器功能中

### 4.2主控客户端

#### 4.2.1登入，登出服务器，注册本机信息

**UDP**

登入，登出目前有两种方式：

一是使用账号，密码登录业务服务器验证，成功后得到session（或P2P服务器地址），向P2P服务器登入并上报本机信息（见表3.1）。退出时先登出P2P服务器再登出业务服务器。

二是直接用账号密码登录P2P服务器，由P2P服务器代理登录业务服务器。退出时登出P2P服务器。

登入请求Json部分：

{

“lanip”:”内网IP”,

“lanport”:内网端口,

“mask”:”子网掩码”,

“gateway”:”网关”,

“ismainctrl”:是否主控端， //1 主控端 0 被控端

“devid”:”设备id”， //被控端时存在此节点

“mac”:”mac地址”， //主控端时存在此节点

“username”:”账号” //主控端时存在此节点

“password”:”密码MD5 的16进制表示” //主控端时存在此节点

}

登出请求json部分：

{

“sessionid”:””,

“guid”:””

}

**TCP**

登入同UDP。

登出请求json部分：

{

“sessionid”:”UUID”,

}

#### 4.2.2维持与服务端”心跳”

**UDP**

定时向服务端发送”心跳”包。未收到确认则重发。

记录成功的心跳确认时间，未收到确认则每30秒以指数退避(1,3,6,12,24)发送心跳

Json部分：

{

“sessionid”：”自己的sessionid”

}

**TCP**

同udp

#### 4.2.3维持与被控端”心跳”

定时向被控端发送”心跳”包。为收到确认则重发。

同上

#### 4.2.4从服务器获取可用的被控端（已绑定设备）列表

**UDP**

请求服务器获取已绑定设备的信息列表。

Json部分：

{

“sessionid”:”11”,

“guid”:”xxx”

}

**TCP**

Json部分：

{

“sessionid”:”uuid”

}

#### 4.2.5向服务器请求连接被控端

**UDP**

向服务器发送需要连接的被控端信息

{

“devinfo”:

{

“sessionid”:”设备session id”,

“devid”:”设备dev id”,

“issamelan”:1, //同一内网私下已联通，此标记仅通知服务器更改占用位

//服务器会验证是否真的同一内网，真则不协助，假则不改占用位

//主控端必须先确认联通再设置为1

//若想强制打洞，那么就设置0

} ,

“session” //自己的会话session

{

“sessionid”:”session id”,

“guid”:”guid”

}

}

#### 4.2.6向被控端发起连接

直接向被控端发起连接。包头+json

JSON部分：

{

“sessionid”:”主控端sessionid”

}

#### 4.2.7请求服务器中转请求

**UDP**

{

“devinfo”:

{

“sessionid”:”设备session id”,

“devid”:”设备dev id”,

} ,

“proxydata”:”base64编码”,

“session” //自己的会话session

{

“sessionid”:”session id”,

“guid”:”guid”

}

}

注意：服务端不对proxydata进行处理，需要被控端自己base64解码

**TCP**

{

“devinfo”:

{

“sessionid”:”设备session id”,

//“devid”:”设备dev id”,

} ,

“proxydata”:”base64编码”,

“session” //自己的会话session

{

“sessionid”:”session id”,

}

}

#### 4.2.8接收服务器中转应答

**UDP**

{

“sessionid”:”设备session id”

“proxydata”:”base64编码”,

“stamp”:”1970年至今的秒数”

}

**TCP**

{

“sessionid”:”设备session id”

“proxydata”:”base64编码”,

}

业务部分：

1. 账号申请，验证，找回
2. 绑定设备增删改查

以上红色部分不请求P2P服务器

### 4.3被控客户端

#### 4.3.1登入，登出服务器，注册本机信息

**UDP**

无账号登入，将本机信息（见表2.1）上报服务器。退出时登出服务器。

Json同4.2.1

**TCP**

登入请求Json部分：

{

“lanip”:”内网IP”,

“lanport”:内网端口, //本地server的端口

“mask”:”子网掩码”,

“gateway”:”网关”,

“ismainctrl”:是否主控端， //1 主控端 0 被控端

“devid”:”设备id”， //被控端时存在此节点

“mac”:”mac地址”， //主控端时存在此节点

“username”:”账号” //主控端时存在此节点

“password”:”密码MD5 的16进制表示” //主控端时存在此节点

“isupnp”:1, //被控端时存在此节点

//注释：isupnp为0时，没有下面这个字段

“upnpip”:”UPNP获取的路由外网IP”, //被控端时存在此节点

“upnpport”:UPNP端口号 //被控端时存在此节点

}

注意：lanport含义不同于UDP。以前是表示UDP client的本地端口（既可以当server，也可以当client），现在改为TCP client，本地这个端口已被与服务器通信占用，上报这个端口没有意义，需要再开一个本地的tcp server，来接收同一内网的app访问。因此在这里表示本地tcp server的端口。

另：当支持UPNP时，upnpport就等同于lanport，即本地tcpserver端口。

详见《P2P智能家居设备通信设计.docx》

#### 4.3.2维持与服务端”心跳”

同4.2.2

#### 4.3.3维持与主控端发来”心跳”

只反馈确认。由主控端主动维持。

#### 4.3.4接受服务器向主控端打洞指令

**UDP**

接收服务端发来的打洞指令和主控端信息。

#### 4.3.5向主控端打洞

**UDP**

向主控端发送一个数据包。也可后跟json格式的sessionid以表示身份。

#### 4.3.6请求服务器中转应答

**UDP**

{

“mainctrl”:

{

“sessionid”:”主控端session id”,

“broadcast”:0或1(即使sessionid无效，亦群发反馈),

} ,

“proxydata”:”base64编码”,

“session” //自己的会话session

{

“sessionid”:”session id”,

“guid”:”guid”

}

}

注意：服务端不对proxydata进行处理，需要主控端自己base64解码

**TCP**

{

“mainctrl”:

{

“sessionid”:”主控端session id”,

“broadcast”:0或1(即使sessionid无效，亦群发反馈),

} ,

“proxydata”:”base64编码”,

“session” //自己的会话session

{

“sessionid”:”session id”,

}

}

#### 4.3.7接收服务器中转请求

**UDP**

{

“sessionid”:”主控端session id”,

“proxydata”:”base64编码”

}

**TCP**

{

“sessionid”:”主控端session id”,

“proxydata”:”base64编码”

}

### 4.4服务端（业务）

计划提供（暂不实现）：

1. 账号维护
2. 查询设备在线信息
3. 维护主控端与设备绑定关系

## 五、封包设计

包头见概要设计。

### 5.1.1操作类型

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 数值 | 说明 |
| OP\_CLASS\_P2P |  | 1 | 发向或来自服务器的请求 |
| OP\_CLASS\_P2PANS |  | 2 | 来自服务器的反馈 |
| OP\_CLASS\_P2PCLIENT |  | 3 | 发向或来自客户端的请求 |
| OP\_CLASS\_P2PCLIENTANS |  | 4 | 来自客户端的反馈 |
| ~~OP\_CLASS\_P2PPROXY~~ |  | ~~5~~ | ~~来自服务器的代理客户端的请求~~ |
| ~~OP\_CLASS\_P2PPROXYANS~~ |  | ~~6~~ | ~~来自服务器的代理客户端的应答~~ |
| OP\_CLASS\_USER |  | 7 | 业务服务器 |

### 5.1.2操作码

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 数值 | 说明 |
| OP\_CODE\_LOGIN |  | 1 | 登入 |
| OP\_CODE\_LOGOUT |  | 2 | 登出 |
| OP\_CODE\_GETDEVINFO |  | 3 | 获取绑定设备信息 |
| OP\_CODE\_CONNDEV |  | 4 | 连接设备 |
| OP\_CODE\_DIG |  | 5 | 打洞命令 |
|  |  |  |  |
| OP\_CODE\_KEEPALIVE |  | 7 | 心跳 |
|  |  |  |  |
| OP\_CODE\_PROXY |  | 9 | 代理请求 |
| OP\_CODE\_PROXYANS |  | 10 | 代理应答 |

### 5.1.3错误代码(暂定)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 数值 | 说明 |
| ERR\_PARSEREQ | 枚举 | -1 | 解析请求信息错误 |
| ERR\_BUILDANS | 枚举 | -2 | 构造返回信息错误 |
| ERR\_LOGIN | 枚举 | -3 | 登陆错误（内存不足） |
| ERR\_LOGOUT | 枚举 | -4 | 登出错误（未登录，重复登出，guid不符） |
| ERR\_MCTRLID | 枚举 | -5 | 主控端身份验证失败（未登录，查询登陆信息失败，非主控端） |
| ERR\_GETDEVINFO | 枚举 | -6 | 获取设备列表失败 |
| ERR\_KEEPALIVE | 枚举 | -7 | 保持心跳失败（未登录） |
| ERR\_UCTRLID | 枚举 | -8 | 被控端身份验证失败（未登录，查询登陆信息失败，非被控端） |
| ERR\_CONNDEV | 枚举 | -9 | 协助打洞失败 |
| ERR\_PROXYREQ | 枚举 | -10 | 代理请求失败 |
| ERR\_PROXYANS | 枚举 | -11 | 代理应答失败 |
| ERR\_BUILDREQ | 枚举 | -12 | 构造请求信息错误 |
| ERR\_PARSEANS | 枚举 | -13 | 解析返回信息错误 |