# 智能家居P2P通信服务器 概要设计

修改记录

版本1.1

6.1修改包头副版本号为1

6.2新增请求头longConnection项

四 修改可行性分析

## 一、概述

P2P是一种网络新技术，依赖网络中参与者的计算能力和带宽，而不是把依赖都聚集在较少的几台服务器上。可以穿越防火墙、可以以私有地址同外界进行双向的数据传输。

## 二、术语

打洞：从B向A方向打洞，即为B向A发送一个数据包。

客户列表：服务端保存的每个客户端（主控端 + 被控端）的IP和端口。

主控端：控制程序

被控端：智能家居设备

UPNP：路由广泛支持的一种协议，通过此协议可以在路由上将任意客户端的端口完全暴露到外网，避免了传统P2P的穿透过程。

## 三、系统结构

对一般P2P来说，客户端互相之间是对等关系，但是考虑到智能家居实际场景下，智能家居设备都是被动受控，而主控端（手机/PC）是主动请求控制智能家居设备，因此主控端需要账号/密码进行身份/权限验证，而被控端不能主动发起连接请求（也可支持），不需要进行账号验证。见图3.1

3、获取绑定设备列表

2．身份/权限验证

绑定账号和设备

5、命令B向A

进行“打洞”

主控客户端

（ClientA）

被控客户端

（ClientB）

服务器端

（Server）

4、（首次）请求

连接客户端B

7、连接

客户端B

6、发送数据包

（打洞）

1、账号登陆、注册客户端IP、端口

1、注册客户端IP、端口

数据库

图3.1

建立连接的示例过程见下图3.2

3、向187.34.1.56:53000

发送一个数据包

服务器端

18.181.0.31:1235

被控端A

10.0.0.1:1234

路由器（NAT）

155.99.25.11

客户列表

clientA

155.99.25.11:62000

内网IP、端口

注册过程

端口转换

1234->62000

IP转换

10.0.0.1->155.99.25.11

路由端口62000只接收18.181.0.31的数据

服务器端

18.181.0.31:1235

被控端A

10.0.0.1:1234

路由器A（NAT）

155.99.25.11

客户列表

clientA/MAC

155.99.25.11:62000

ClientB/Telnum

187.34.1.56:53000

不同网段客户端间连接过程

1、B请求连接A

2、命令A向B打洞

主控端B

10.0.0.1:1234

路由器B（NAT）

187.34.1.56

其他IP

187.34.1.56

4、B->A

建立连接

数据库

账号列表

clientB

权限

绑定设备

图3.2

## 四、可行性分析

**通信协议选择**

P2P可以采用TCP或UDP两种方式进行通信，考虑到TCP穿透较复杂，且部分路由不支持。因此在1.0版本中：客户端与服务端间的打洞协助部分采用UDP方式进行通信。

但是丢包重发是个麻烦事，并且经过调研与实践，发现通过UPNP方式（也是目前其他厂商的方法），可以规避传统复杂穿透，是直接将端口暴露出去。这种情况下TCP不再复杂。因此在1.1版本中：暂时摒弃UDP方式，改由TCP实现。将来再考虑UDP与TCP共存问题。

**心跳探测**

1.0版本中，客户端通过NAT连接服务器，会保留一个session，若长时间不连接会释放，因此需要维持一个”心跳”。我们用自定义的心跳包来实现。

在1.1版本中我们预计会同时引入TCP和UPNP两种模式，对于传统单纯TCP穿透，仍需要维持心跳包。而对于UPNP模式，暴露在外网的端口不需要心跳包来维持。但是服务器仍需要清理那些异常丢失的连接，因此仍不能避免维持心跳。

TCP通常自带KEEPLIVE，由服务端主动打开此选项（客户端也可，但是有赖于客户端自觉），增加了服务端的压力。因此还是通过自定义心跳包实现，由客户端发送，服务端仅定时删除dead客户端。

**用户功能**

账号申请，权限管理，设备绑定可以通过TCP或HTTP或通过java web来实现

**NAT穿透类型**

目前有四种NAT

1. 完全型锥形full cone
2. 限制型锥形restricted cone
3. 端口限制型锥形port restricted cone
4. 对称型symmetric

互相之间连通图如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 完全型锥形full cone | 限制型锥形restricted cone | 端口限制型锥形  port restricted cone | 对称型  symmetric |
| 完全型锥形  full cone | 是 | 是 | 是 | 是 |
| 限制型锥形restricted cone | 是 | 是 | 是 | 是 |
| 端口限制型锥形  port restricted cone | 是 | 是 | 是 | 否 |
| 对称型  symmetric | 是 | 是 | 否 | 否 |

NAT类型测试：<http://cc.rtmfp.net/>

检测了十几个公司，家庭路由器，其中1/3是4型，大部分是3型，少量是2型。样本较少，可能不准确。

说明市场主流是2，3型路由，而随着安全性要求提高，第四种路由也越来越多。为了确保能进行P2P打穿，至少要保证被控端设备处是前两种路由。实在不行就只能服务器中转。

1.1版本中，通过引入UPNP，可有效避免NAT穿透。但当用户不愿开启路由的UPNP支持时，我们可将NAT穿透作为备选。最坏情况是通过服务器中转。

## 五、基本功能

### 5.1服务器端

P2P协助通信部分：

1. 记录客户端登入、登出，维护客户端列表（IP + 端口号）。
2. 维持与客户端发来”心跳”。
3. 为主控端提供被控端（已绑定设备）列表。
4. 接收主控端连接被控端的请求。
5. 命令被控端向主控端打洞。
6. 为客户端间提供中转传输

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. 设备绑定管理

以上红色部分不在P2P服务器功能中

### 5.2客户端

主控端：

1. 登入，登出服务器，注册本机信息（IP + 端口号）
2. 维持与服务端”心跳”
3. 维持与被控端”心跳”
4. 从服务器获取可用的被控端（已绑定设备）列表
5. 向服务器请求连接被控端
6. 向被控端发起连接
7. 请求服务器中转请求
8. 账号申请，找回
9. 绑定设备的增删改查

以上红色部分不请求P2P服务器

被控端：

1. 登入，登出服务器，注册本机信息（IP + 端口号）
2. 维持与服务端”心跳”
3. 维持与主控端发来”心跳”
4. 接受服务器向主控端打洞指令
5. 向主控端打洞
6. 请求服务器中转应答

## 六、封包结构

包头采用网络字节序，注意int到unsigned char[4], short到unsigned char[2]的转换。

### 6.1 UDP

考虑MTU过长导致分包，暂定请求包长度不超过1K，包头32字节

UDP包一次最多64K，应答包长度也不宜过长，包头32字节。尤其主控端获取已绑定被控端信息列表时要注意。

**请求头**

考虑与TCP方式风格一致，采用包头+JSON数据。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| cbSize | UChar数组 | 2 | 结构体大小 32 |
| nMajorVer | UChar数组 | 1 | 主版本号 1 |
| nMinorVer | UChar数组 | 1 | 副版本号 1 |
| szMagicStr | UChar数组 | 8 | $bluev$ |
| nOpClass | UChar数组 | 2 | 操作类型 |
| nOpCode | UChar数组 | 2 | 操作码 |
| nDataLen | UChar数组 | 2 | 后面的数据长度 |
| bnRes14 | UChar数组 | 14 | 补齐 |

**应答头**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| cbSize | UChar数组 | 2 | 结构体大小 32 |
| nMajorVer | UChar数组 | 1 | 主版本号 1 |
| nMinorVer | UChar数组 | 1 | 副版本号 1 |
| szMagicStr | UChar数组 | 8 | must be $bluev$ |
| nOpClass | UChar数组 | 2 | 操作类型 |
| nOpCode | UChar数组 | 2 | 操作码 |
| nDataLen | UChar数组 | 2 | 后面的数据长度 |
| nRetVal | UChar数组 | 2 | 返回值(大于等于0表示成功，小于0表示失败。错误原因参见详细设计文档错误代码部分) |
| bnRes | UChar数组 | 12 | reserved |

**此处只提供包头定义，操作类型，操作码以及包头后跟的json数据格式等见详细设计文档**

### 6.2 TCP（暂不实现）

包头32字节，与UDP包头大小一致，含义略有不同，详见包头定义

**请求头**

与UDP方式风格一致，采用包头+JSON数据。

包头中包含必要信息，TCP时有助服务端筛选和过滤非法请求，不必等到接收完再解析。

数据使用JSON便于解析。

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| cbSize | UChar数组 | 2 | 结构体大小 32 |
| nMajorVer | UChar数组 | 1 | 主版本号 1 |
| nMinorVer | UChar数组 | 1 | 副版本号 0 |
| szMagicStr | UChar数组 | 8 | $bluev$ |
| nOpClass | UChar数组 | 2 | 操作类型 |
| nOpCode | UChar数组 | 2 | 操作码 |
| nDataLen | UChar数组 | 4 | 后面的数据长度 |
| reuseConnection | UChar数组 | 1 | 临时长连接，不建议使用 |
| longConnection | UChar数组 | 1 | 真正的长连接，服务端不主动close |
| bnRes | UChar数组 | 10 | 补齐 |

**应答头**

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 长度 | 说明 |
| cbSize | UChar数组 | 2 | 结构体大小 32 |
| nMajorVer | UChar数组 | 1 | 主版本号 1 |
| nMinorVer | UChar数组 | 1 | 副版本号 0 |
| szMagicStr | UChar数组 | 8 | must be $bluev$ |
| nDataLen | UChar数组 | 4 | 后面的数据长度 |
| nRetVal | UChar数组 | 2 | 返回值 |
| hasException | UChar数组 | 1 | 是否出现异常 |
| bnRes | UChar数组 | 13 | Reserved |

**此处只提供包头定义，操作类型，操作码以及包头后跟的json数据格式等见详细设计文档**