**P2P模块详细设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 内容 | 作者 | 时间 |
| 1.0 | P2P模块详细方案初稿 | 李纯 | 2015/3/6 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 概述 2](#_Toc413765247)

[2 设计思想 2](#_Toc413765248)

[3 示意图 3](#_Toc413765249)

[3.1 STUN通信示意图 3](#_Toc413765250)

[3.2 核心模块调用关系图 3](#_Toc413765251)

[4 接口定义 3](#_Toc413765252)

# 概述

P2P技术属于覆盖层网络(Overlay Network)的范畴，是相对于客户机/服务器(C/S)模式来说的一种网络信息交换方式。在C/S模式中，数据的分发采用专门的服务器，多个客户端都从此服务器获取数据。这种模式的优点是：数据的一致性容易控制，系统也容易管理。但是此种模式的缺点是：因为服务器的个数只有一个(即便有多个也非常有限)，系统容易出现单一失效点；单一服务器面对众多的客户端，由于CPU能力、内存大小、网络带宽的限制，可同时服务的客户端非常有限，可扩展性差。P2P技术正是为了解决这些问题而提出来的一种对等网络结构。在我们这个流媒体服务器模型中，存在两端及设备端和APP端，为了统一接口实现，为以上两端实现统一调用接口

# 2 设计思想

此设计的流媒体模型主要是为了减轻服务器端的压力，流数据能在设备端和APP端进行传输。所以需要采用P2P中点对点穿透技术来解决。目前行业上有多种技术可以解决点对点通信方案，经过调研大概如下，

ALG技术：传统的NAT技术只能检测网络层与传输层地址，你FTP不是在应用层传送端口地址吗？所以ALG技术就是一种应用层地址识别技术，根据不同的协议进行检测，然后将发现的地址时行修改并通知NAT做相应的映射记录，从而实现成功通信．对于该技术，对于互联网每一种新的协议支持都要更新其设备要不无法识别．扩展性差．对于个别协议该技术无法解决问题，如IPSEC协议．

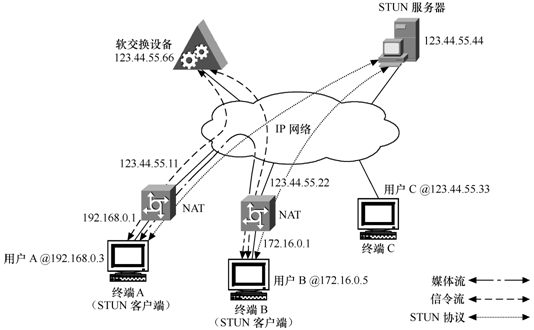
UPnP技术：网关对外提供标准接口，可以动态配置端口映射、动态打开端口、获取外部地址等功能，client可利用这些功能实现穿越；缺点：不支持多层私网，存在安全问题；并且需要路由器或者网关支持此协议

STUN技术：通过STUN协议与第三方服务器建立连接，推测出客户端的NAT类型．进而通信。不依赖应用层协议及网关和路由器的限制

综上所述， 项目采用stun协议作为点对点的通信的基础协议。通过stun协议设备端和APP端可以获取各自的nat类型和外网映射地址。并和第三方服务器进行信令通信交换双方的stun信息。按照两端的NAT类型实现一套穿透算法打通P2P通道，传输方式采用UDP

# 3 示意图

## 3.1 STUN通信示意图



注释

1. 上述软交换设备对应项目为smartcloud服务器

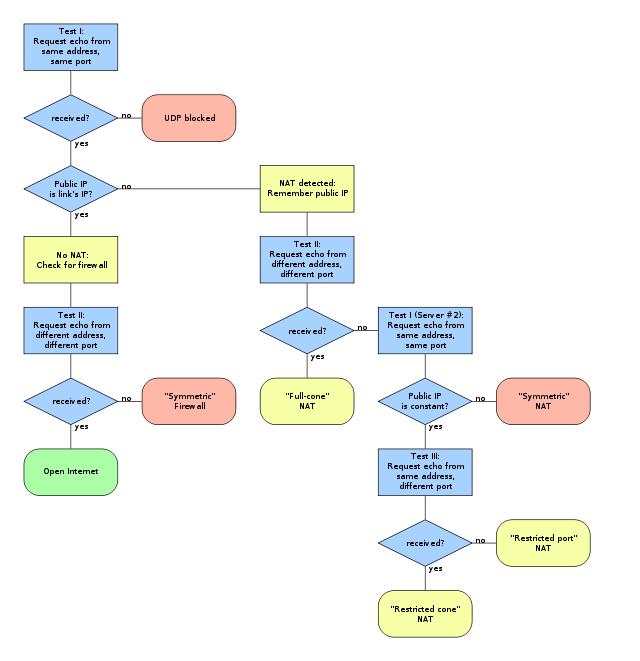
## 3.2 核心模块调用关系图



# 4 接口定义

|  |  |
| --- | --- |
| 函数接口定义 | int GetLocalStun(char\* stunip, \_J\_STUN\_INFORMATION\* stuninfo);  通过stun服务器获取本地stun信息 |
| 函数参数说明 | stunip: stun服务器的IP地址  stuninfo: 返回调用者的stun信息 |
| 返回值 | 返回0 成功，其它则错误 |

**STUN**（**Simple Traversal of User Datagram Protocol through Network Address Translators (NATs)**，NAT的UDP简单穿越）是一种网络协议，它允许位于NAT（或多重NAT）后的客户 端找出自己的公网地址，查出自己位于哪种类型的NAT之后以及NAT为某一个本地端口所绑定的Internet端端口。这些信息被用来在两个同时处于 NAT路由器之后的主机之间建立UDP通信。该协议由RFC 3489定义。STUN 使用下图的算法（取自RFC 3489）来发现 NAT gateways 以及[防火墙](http://baike.baidu.com/view/3067.htm" \t "_blank)（firewalls）: 一旦路经通过红色箱子的终点时，UDP的沟通是没有可能性的。一旦通过黄色或是绿色的箱子，就有连线的可能。



注释，

1. 本SDK所采用的stun实现代码移植于开源项目 <http://sourceforge.net/projects/stun/> ，采用其中的client实现部分

|  |  |
| --- | --- |
| 函数接口定义 | int StunInfoSetting(\_J\_STUN\_INFORMATION\* stundev,  \_J\_STUN\_INFORMATION\* stunapp);  设置设备和APP 的stun 信息 |
| 函数参数说明 | stundev: 设备端的stun信息  stunapp: APP端的stun信息 |
| 返回值 | 返回0 成功，其它则错误 |

|  |  |
| --- | --- |
| 函数接口定义 | int StartP2PSession();  p2p通道会话打通 |
| 函数参数说明 |  |
| 返回值 | 返回0 成功，其它则错误 |

注释

此函数为P2P核心函数实现，具体实现流程参考下图



其中心跳包超时还未实现

|  |  |
| --- | --- |
| 函数接口定义 | int SetH264PacketMode(int PktMode)  设备端停止推送数据 |
| 函数参数说明 | PktMode: 对H264有三种模式  0 <必须使用单一 NALU 单元模式>  1 <非交错(non-interleaved)封包模式>  2 <交错(interleaved)封包模式> |
| 返回值 | 返回0 成功，其它则错误 |
| 备注 | 此函数还未实现 |

|  |  |
| --- | --- |
| 函数接口定义 | int DEVStreamPush(\_J\_STREAM\_PACKET\* streamPkt);  设备端推送音视频流数据通过RTP封装发送 |
| 函数参数说明 | streamPkt: 推送流数据的结构体 |
| 返回值 | 返回1 则发送成功，其它则错误 |
| 备注 | 此函数RTP封装部分还未实现  相关数据结构  typedef enum {  MEDIA\_VIDEO\_TYPE,  MEDIA\_AUDIO\_TYPE,  MEDIA\_MIXED\_TYPE  }J\_MEDIA\_TYPE; // 媒体类型  typedef struct {  unsigned char\* buf; //数据指针  int len; // 数据的长度  J\_MEDIA\_TYPE mtype;  }\_J\_STREAM\_PACKET; //设备推流数据结构体 |

|  |  |
| --- | --- |
| 函数接口定义 | int DEVStreamStop();  设备端停止推送数据 |
| 函数参数说明 |  |
| 返回值 | 返回0 成功，其它则错误 |

|  |  |
| --- | --- |
| 函数接口定义 | int APPStreamCallback(appStreamCallback streamCallback);  APP端设置流媒体数据回调 |
| 函数参数说明 | streamCallback: 接收流数据回调函数 |
| 返回值 | 返回0 成功，其它则错误 |
| 备注 | 此接口相关数据结构定义  typedef enum {  RTP\_STREAM\_PACKET,  RAW\_STREAM\_PACKET,  ENCODE\_STREAM\_PACKET  }P2PSTRAMTYPE;  typedef struct {  char \*p2pBuf;  int length;  P2PSTRAMTYPE type;  }p2pStream;  typedef void (\*appStreamCallback)(p2pStream \*p2pkt); |

|  |  |
| --- | --- |
| 函数接口定义 | int APPStreamStop();  APP端停止收送数据 |
| 函数参数说明 |  |
| 返回值 | 返回0 成功，其它则错误 |

|  |  |
| --- | --- |
| 函数接口定义 | void enableP2PDebug();  打开调试信息 |
| 函数参数说明 |  |
| 返回值 |  |