元海智联P2P SDK编程指南

目录

[1 简介 2](#_Toc497382904)

[2 基本流程 2](#_Toc497382905)

[2.1 P2P 客户端调用流程 3](#_Toc497382906)

[2.2 P2P 设备端调用流程 3](#_Toc497382907)

[3 P2P SDK函数说明 4](#_Toc497382908)

[3.1 类型说明 4](#_Toc497382909)

[3.2 初始化P2P SDK 5](#_Toc497382910)

[3.3 卸载P2P SDK 5](#_Toc497382911)

[3.4 连接TURN服务器 5](#_Toc497382912)

[3.5 断开TURN 服务器 6](#_Toc497382913)

[3.6 连接TURN服务器的回调函数 6](#_Toc497382914)

[3.7 turn服务断开的回调函数 6](#_Toc497382915)

[3.8 连接P2P设备 6](#_Toc497382916)

[3.9 断开P2P设备 7](#_Toc497382917)

[3.10 连接远程用户的回调函数 7](#_Toc497382918)

[3.11 设备端收到P2P连接的回调函数 8](#_Toc497382919)

[3.12 远程连接断开的回调函数 8](#_Toc497382920)

[3.13 P2P连接收到数据的回调函数 8](#_Toc497382921)

[3.14 创建TCP连接代理 9](#_Toc497382922)

[3.15 销毁TCP连接代理 9](#_Toc497382923)

[3.16 TCP连接代理的回调函数 10](#_Toc497382924)

[3.17 发送业务数据 10](#_Toc497382925)

[4 多服务器负载均衡 11](#_Toc497382926)

[4.1 基础 11](#_Toc497382927)

[4.2 负载均衡函数说明 11](#_Toc497382928)

[P2P 设备端请求分配TURN 服务器 11](#_Toc497382929)

[P2P 客户端查找远程设备的TURN 服务器 12](#_Toc497382930)

[负载均衡分派结果回调函数 12](#_Toc497382931)

[5 错误码 13](#_Toc497382932)

# 简介

元海智联P2P SDK（简称 P2PSDK）是基于RFC标准STUN、TURN、 ICE通信协议开发的，融合并修改了三个开源代码库PJNATH 、UDT、MiniUPnP。P2PSDK为网络考勤机、网络摄像机等网络产品的提供一个基础的网络通讯模块，实现端到端的P2P直接传输数据。

# 基本流程

P2P SDK涉及到三个编程角色：

1. P2P 客户端，主动发起P2P连接的一端。
2. P2P 设备端，被动接收P2P连接的一端。
3. TURN 服务器端，实现STUN、TURN协议的服务器端。

P2P SDK可以使用两种方式进行通信

1. 使用代理方式
2. 使用P2P SDK直接发送和接收数据

为了简化开发，P2P SDK可以使用TCP代理模式，第三方程序基本不需要改动。原有的基本结构如下：

TCP设备端

TCP客户端

使用了P2P SDK以后基本结构如下

TCP设备端

P2P设备端代理

P2P客户端代理

TCP客户端

## P2P 客户端调用流程

卸载SDK

初始化SDK

连接TURN 服务器

创建TCP连接代理

连接P2P设备

断开TURN 服务器

断开P2P设备

销毁TCP连接代理

TCP连接代理自动转发数据到P2P服务器

或者使用P2P SDK 发送和接收数据

## P2P 设备端调用流程

初始化SDK

卸载SDK

断开TURN 服务器

连接TURN 服务器

SDK自动接收并处理P2P连接或者使用P2P SDK 发送和接收数据

# P2P SDK函数说明

## 类型说明

|  |  |
| --- | --- |
| 类型 | 说明 |
| p2p\_transport | P2P客户端与TURN 服务器端的一个连接。 |
| p2p\_transport\_cfg | typedef struct \_p2p\_transport\_cfg  {  //TURN服务器地址  char\* server;  //TURN服务器端口  unsigned short port;  //终端类型  #define P2P\_DEVICE\_TERMINAL 0 设备端 #define P2P\_CLIENT\_TERMINAL 1 客户端，手机，PC等  unsigned char terminal\_type;  //p2p用户  char\* user;  //p2p密码  char\* password;  //用户数据,回调函数时原样返回  void \*user\_data;  //使用TCP连接TURN服务器，默认使用UDP连接TURN服务器  int use\_tcp\_connect\_srv;  //代理地址，如果不传值，默认127.0.0.1  char\* proxy\_addr;  //回调函数  const p2p\_transport\_cb \*cb;  }p2p\_transport\_cfg;  P2P客户端和TURN服务器的信息。 |
| p2p\_transport\_cb | P2P SDK 回调函数  on\_create\_complete 连接服务器的回调函数。  on\_disconnect\_server如果连接服务器成功，客户端保持P2P连接，然后因为网络原因或者服务器程序崩溃，会回调这个函数。  on\_connect\_complete P2P连接远程用户的回调函数。  on\_connection\_disconnect P2P连接远程用户成功后，客户端保持P2P连接，然后因为网络原因或者远端用户关闭，会回调这个函数。  on\_accept\_remote\_connection 收到一个远程用户的连接。  on\_connection\_recv 收到远程用户的数据。  on\_tcp\_proxy\_connected 设备端收到一个TCP代理连接 |

## 初始化P2P SDK

函数： int p2p\_init()

参数： 无

返回值：0表示成功，非0表示失败。

说明： 初始化P2P SDK，调用P2P SDK其他函数之前必须调用此函数。

## 卸载P2P SDK

函数： void p2p\_uninit()

参数： 无。

返回值：无。

说明： 卸载P2P SDK，调用此函数后，不可以再调用P2P SDK的其他函数了。

## 连接TURN服务器

函数： int p2p\_transport\_create(p2p\_transport\_cfg\* cfg,

p2p\_transport \*\*transport);

参数： 1、p2p\_transport\_cfg\* cfg，P2P客户端和TURN的信息。

2、p2p\_transport\*\* transport，P2P客户端与TURN 服务器端的一个连接。

返回值：int，0 成功，非0失败。

说明： 连接TURN服务器，通过p2p\_transport\_cfg里的on\_create\_complete回调函数异步返回连接结果。

## 断开TURN 服务器

函数： void p2p\_transport\_destroy(p2p\_transport \*transport);

参数： p2p\_transport \*transport，p2p\_transport\_create函数所创建的连接对象指针。

返回值：无。

说明： 断开TURN服务器的连接。

## 连接TURN服务器的回调函数

函数： void (\*on\_create\_complete)(p2p\_transport \*transport,

int status,

void \*user\_data)

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int status，连接结果，0 连接成功，非0 连接失败

3、void \*user\_data， 用户数据。

返回值：无。

说明： 连接TURN服务器的回调函数，p2p\_transport\_create函数真正的执行结果。

## TURN服务断开的回调函数

函数： void (\*on\_disconnect\_server)(p2p\_transport \*transport,

int status,

void \*user\_data);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int status，连接结果，0 连接成功，非0 连接失败

3、void \*user\_data， 用户数据。

返回值：无。

说明： 与TURN服务器的连接断开的回调函数

## 连接P2P设备

函数： int p2p\_transport\_connect

(p2p\_transport \*transport,

const char\* remote\_user,

void \*user\_data,

int conn\_flag,

int\* connection\_id)

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、const char\* remote\_user，P2P用户名，一般是设备的UID。

3、void\* user\_data，用户数据，回调函数执行时原样返回。

4、int conn\_flag, 连接标识，与设备端on\_accept\_remote\_connection回调函数conn\_flag参数对应。调用者自定义该连接的标识。

5、int\* connection\_id，连接ID，程序内部动态生成，P2P连接的唯一标识。

返回值：int，0 成功，非0失败。

说明： 连接P2P设备，通过p2p\_transport\_cfg里的on\_connect\_complete回调函数异步返回连接结果。

## 断开P2P设备

函数： void p2p\_transport\_disconnect( p2p\_transport \*transport,

int connection\_id)

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，p2p\_transport\_connect函数生成的唯一标识。

返回值：无。

说明： 断开与P2P设备的连接。

## 连接远程用户的回调函数

函数： void (\*on\_connect\_complete)( p2p\_transport \*transport,

int connection\_id,

int status,

void \*transport\_user\_data,

void \*connect\_user\_data);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，p2p\_transport\_connect函数生成的唯一标识。

3、int status，连接结果，0 连接成功，非0 连接失败

4、void \* transport\_user\_data，p2p\_transport\_cfg里的用户数据。

5、void \*connect\_user\_data，p2p\_transport\_connect指定的用户数据

返回值：无。

说明： 连接远程用户的回调函数，p2p\_transport\_connect函数真正的执行结果。

## 设备端收到P2P连接的回调函数

函数： void (\*on\_accept\_remote\_connection)(p2p\_transport \*transport,

int connection\_id,

int conn\_flag,

void \*transport\_user\_data)

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，P2P连接的唯一标识。

3、int conn\_flag，连接标识,参考p2p\_transport\_connect函数

4、void \* transport\_user\_data，p2p\_transport\_cfg里的用户数据

返回值：无。

说明： 设备端收到一个P2P连接

## 远程连接断开的回调函数

函数：void (\*on\_connection\_disconnect)(p2p\_transport \*transport,

int connection\_id,

void \*transport\_user\_data,

void \*connect\_user\_data);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，客户端p2p\_transport\_connect函数生成的唯一标识，设备端on\_accept\_remote\_connection函数的第二个参数。

3、void \* transport\_user\_data，p2p\_transport\_cfg里的用户数据。

4、void \*connect\_user\_data，客户端 p2p\_transport\_connect指定的用户数据，设备端为NULL

返回值：无。

说明： P2P连接断开了

## P2P连接收到数据的回调函数

函数： void (\*on\_connection\_recv)(p2p\_transport \*transport,

int connection\_id,

void \*transport\_user\_data,

void \*connect\_user\_data,

char\* data,

int len);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，P2P连接的唯一标识。

3、void \* transport\_user\_data，p2p\_transport\_cfg里的用户数据

4、void \*connect\_user\_data，客户端 p2p\_transport\_connect指定的用户数据，设备端为NULL

5、char\* data,收到的数据指针

6、int len，收到的数据长度

返回值：无。

说明： P2P连接收到数据

## 创建TCP连接代理

函数： int p2p\_create\_tcp\_proxy(p2p\_transport \*transport,

int connection\_id,

unsigned short remote\_listen\_port,

unsigned short\* local\_proxy\_port)

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，p2p\_transport\_connect函数生成的唯一标识。

3、unsigned short remote\_listen\_port，P2P设备的监听端口，即P2P设备端实际业务数据的socket listen端口。

4、unsigned short\* local\_proxy\_port，指定TCP连接代理的socket listen端口，若指定的端口被系统占用则由程序内部随机生成。

返回值：int，0 成功，非0失败。

说明： 在P2P客户端监听一个socket，创建TCP代理，接收用户的TCP连接。使用P2P连接，在P2P客户端和P2P设备之间转发TCP数据。

## 销毁TCP连接代理

函数： (void) p2p\_destory\_tcp\_proxy(p2p\_transport \*transport,

int connection\_id,

unsigned short local\_proxy\_port);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，p2p\_transport\_connect函数生成的唯一标识。

3、unsigned short local\_proxy\_port，p2p\_create\_tcp\_proxy 创建的TCP监听端口。

返回值：无。

说明： 注销TCP连接代理，将断开所有socket listen端口上的用户TCP连接。

## TCP连接代理的回调函数

函数： void (\*on\_tcp\_proxy\_connected)(p2p\_transport \*transport,

unsigned short port,

char\* addr);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、unsigned short port， TCP端口

3、char\* addr,TCP 地址，客户端的公网地址。

返回值：无。

说明： 设备端收到一个TCP连接代理。

## 发送业务数据

函数： int p2p\_transport\_send(p2p\_transport \*transport,

int connection\_id,

char\* buffer,

int len,

p2p\_send\_model model,

int\* error\_code);

参数： 1、p2p\_transport\* transport，p2p\_transport\_create函数所创建的连接对象指针。

2、int connection\_id，连接ID，p2p\_transport\_connect函数生成的唯一标识。

3、char\* buffer,需要发送的数据缓冲区

4、int len,需要发送的数据缓冲区长度

5、p2p\_send\_model model，发送模式，阻塞和非阻塞

6、int\* error\_code,错误码，0 成功，非0其他错误

返回值：成功则返回已发送的数据长度，失败返回-1,如果连接已经无效，返回0

非阻塞模式，如果缓冲区满了，返回-1，error\_code 等于70027

说明： 使用这个函数发送实际的业务数据

# 多服务器负载均衡

## 基础

P2P TURN SERVER可以支持多服务器部署进行负载均衡。

P2P SDK涉及到三个编程角色：

1. P2P 客户端，主动发起P2P连接的一端。
2. P2P 设备端，被动接收P2P连接的一端。
3. TURN 服务器端，实现STUN、TURN协议的服务器端。

若果要多服务器部署，必须增加一个新的角色：P2P负载均衡分派服务器。所有的TURN 服务器端都连接到P2P负载均衡分派服务器，并上报自己的负载信息。

负载均衡分派服务器也可以部署多个，每个负载均衡分派服务器都是平等的，可以独立运行。

P2P设备端连接过程如下：

1. 请求P2P负载均衡分派服务器分配一个最优TURN 服务器端
2. P2P 设备器端使用P2P负载均衡分派服务器返回的TURN 服务器端信息，连接TURN 服务器端
3. TURN 服务器端将信息保存到全局存储区域（分布式数据库）

P2P 客户端连接过程如下：

1. 请求P2P负载均衡分派服务器，在全局存储区域（分布式数据库）查找P2P 设备端所连接的TURN 服务器端
2. P2P 客户端端使用P2P负载均衡分派服务器返回的TURN 服务器端信息，连接TURN 服务器端
3. 向TURN 服务器端发起连接P2P 设备端的请求

## 负载均衡函数说明

### P2P 设备端请求分配TURN 服务器

函数： int p2p\_request\_dispatch\_server(char\* user,

char\* password,

char\* ds\_addr,

void\* user\_data,

DISPATCH\_CALLBACK cb,

void\*\* dispatcher);

参数： 1、char\* user 用户名。

2、char\* password 密码

3、char\* ds\_addr, 负载均衡分派服务器地址，多个地址以分号间隔，例如 服务器1:服务器1端口; 服务器2:服务器2端口, 192.168.0.1:9999; 192.168.0.2:9999

4、void\* user\_data，用户数据，回调函数执行时原样返回

5、回调函数

6、void\*\* dispatcher，请求对象。完成后调用destroy\_dispatch\_requester释放内存

返回值：int，0 成功，非0失败。

说明： 请求P2P负载均衡分派服务器分配一个最优TURN 服务器端

### P2P 客户端查找设备的TURN 服务器

函数： int p2p\_query\_dispatch\_server (char\* dest\_user,

char\* ds\_addr,

void\* user\_data,

DISPATCH\_CALLBACK cb,

void\*\* dispatcher);

参数： 1、char\* dest\_user 远程用户

2、char\* ds\_addr, 负载均衡分派服务器地址，多个地址以分号间隔，例如 服务器1:服务器1端口; 服务器2:服务器2端口, 192.168.0.1:9999; 192.168.0.2:9999

3、void\* user\_data，用户数据，回调函数执行时原样返回

4、回调函数

5、void\*\* dispatcher，请求对象。完成后调用destroy\_dispatch\_requester释放内存

返回值：int，0 成功，非0失败。

说明： 查找设备端登录到哪一个TURN服务器

### 销毁分派请求

函数：void destroy\_dispatch\_requester(void\* dispatcher)

参数：void\* dispatcher, p2p\_request\_dispatch\_server函数或p2p\_query\_dispatch\_server函数返回的对象。

返回值：无

说明: 销毁分派请求，回收内存。

### 负载均衡分派结果回调函数

函数： typedef void (\*DISPATCH\_CALLBACK)(int status, void\* user\_data, char\* server, unsigned short port, unsigned int server\_id);

参数： 1、int status，0接成功，非0失败

2、void\* user\_data，用户数据，回调函数执行时原样返回

3、char\* server，TURN 服务器地址

4、unsigned short port, TURN 服务器端口

5、unsigned int server\_id，TURN 服务器ID

返回值：无

说明： p2p\_query\_dispatch\_server和p2p\_request\_dispatch\_server的实际结果

# 错误码

* 70001 未知错误
* 70002 操作正在进行中
* 70003 SOCKET数量超过限制
* 70004 无效的参数
* 70005 主机名太长
* 70006 对象没有找到
* 70007 内存不足
* 70008 内部BUG
* 70009 超时
* 70010 对象太多
* 70011 对象正忙
* 70012 指定的操作不支持
* 70013 无效的操作
* 70014 操作被取消
* 70015 对象已经存在
* 70016 到达文件尾部
* 70017 超过了指定的大小
* 70018 找不到指定的主机
* 70019 小于指定的大小
* 70020 操作被忽略
* 70021 IPV6 不支持
* 70022 无效的IP族
* 70023 对象不存在
* 70024 socket已关闭
* 70025 创建socket pair 失败
* 70026 UDT 连接失败
* 370001 无效的STUN消息
* 370002 无效的STUN消息长度
* 370003 无效的STUN消息类型
* 370004 STUN请求超时
* 370021 太多的STUN属性
* 370022 无效的STUN属性长度
* 370023 重复的的STUN属性
* 370030 STUN 电子指纹信息验证失败
* 370031 无效的STUN MESSAGE-INTEGRITY属性
* 370032无效的STUN FINGERPRINT属性
* 370040 STUN (XOR-)MAPPED-ADDRESS属性没找到
* 370041 STUN 不支持IPV6
* 370042 无效的STUN 地址
* 370050 无效的STUN 服务器
* 370060 STUN 对象已经被销毁
* 370080 无效的ICE 会话
* 370081 ICE检测正在进行
* 370082 ICE检测失败
* 370083 没有有效的ICE候选对象
* 370086 无效的ICE组件ID
* 370087 无效的IC候选ID
* 370088 源地址不匹配
* 370090 ICE SDP 无效
* 370091 ICE SDP中候选对象无效
* 370092 没有关于srflx候选对象
* 370093 被控端代理超时
* 370120 无效的TURN transport
* 370201 P2P客户端被踢出
* 470001 连接数据库失败
* 470002 查询数据库失败
* 470003 无效的用户名和密码
* 470004 找不到有效的TURN服务器
* 470005 用户离线