文件状态：[ √ ] 草稿 [ ]通过评审正式发布 [ ]未通过评审需修改

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GOSCAM-logo  深圳市高斯贝尔家居智能电子有限公司 | 文档编号 | NO.001 | 版本 | V1.0 | 密级 | 高 |
| 项目名称 | uLife 4.0 | | | | |
| 产品型号 | uLife 4.0 | | | | |
| 文件作者 | 贺磊 | | | | |

**P2P穿透方案协议文档**

|  |  |  |  |
| --- | --- | --- | --- |
| 更改说明 | 修改者 | 修改完成时间 | 版本号 |
| 1、初版 |  | 2017/11/22 | V1.1 |
|  |  |  |  |
|  |  |  |  |

目录

[摘要 2](#_Toc15079)

[名词解释 2](#_Toc21795)

[关键词 2](#_Toc29430)

[一、基本概念 3](#_Toc10907)

[1.1 NAT的种类 3](#_Toc3136)

[二、协议结构(STUN报文结构) 3](#_Toc22979)

[2.1消息描述 3](#_Toc13657)

[2.2消息格式 3](#_Toc17384)

[三、协议内容 4](#_Toc11062)

[3.1 消息类型 4](#_Toc6021)

[3.2 消息属性 7](#_Toc6751)

[四、交互流程 11](#_Toc28465)

[4.1 术语 11](#_Toc31372)

[4.2 p2p 12](#_Toc5431)

[4.3 Relay 12](#_Toc16493)

# **摘要**

本文介绍的是一种基于UDP的p2p穿透方案，基于PJNATH库,ICE方案来实现的。主要介绍P2P,RELAY两种模式的交互流程。

# **名词解释**

|  |  |
| --- | --- |
| 类型 | 描述 |
| NAT | 网络地址转换 |
| [PJNATH](http://www.pjsip.org/pjnath/docs/html/index.htm) | PJSIP NAT Helper,一个包含基于标准的NAT穿越解决方案的实现的库 |
| ICE | 用于发现两个端点之间的通信路径的协议。PJNATH中的实现符合[draft-ietf-mmusic-ice-19.txt](http://www.ietf.org/internet-drafts/draft-ietf-mmusic-ice-19.txt)草案 |
| STUN | Simple Traversal of User Datagram Protocol Through Network Address Translators简单的UDP穿透NAT |
| TURN | Traversal Using Relays around NAT：Relay Extensions to Session Traversal Utilities for NAT（STUN）中继穿透NAT：STUN的中继扩展 |
| RTO | Retransmission TimeOut，超时重传时间 |
| ATO | Delay ack timeout,延迟ack超时时间 |
| RTT | Round Trip Time，数据包从发出去到回来的时间 |
| CWND | Congestion Window,拥塞窗口 |
| ssthresh | slow start threshold,慢启动门槛 |

# **关键词**

协议结构、协议内容

# 一、基本概念

## 1.1 NAT的种类

NAT对待UDP的实现方式有4种，分别如下：

1.    Full Cone NAT

完全锥形NAT，所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号，并且任何一个外网主机都可以通过这个映射的外网IP和端口号向这台内网主机发送包。

2.    Restricted Cone NAT

限制锥形NAT，它也是所有从同一个内网IP和端口号发送过来的请求都会被映射成同一个外网IP和端口号。与完全锥形不同的是，外网主机只能够向先前已经向它发送过数据包的内网主机发送包。

3.    Port Restricted Cone NAT

端口限制锥形NAT，与限制锥形NAT很相似，只不过它包括端口号。也就是说，一台IP地址X和端口P的外网主机想给内网主机发送包，必须是这台内网主机先前已经给这个IP地址X和端口P发送过数据包。

4.    Symmetric NAT

对称NAT，所有从同一个内网IP和端口号发送到一个特定的目的IP和端口号的请求，都会被映射到同一个IP和端口号。如果同一台主机使用相同的源地址和端口号发送包，但是发往不同的目的地，NAT将会使用不同的映射。此外，只有收到数据的外网主机才可以反过来向内网主机发送包。

# 二、协议结构(STUN报文结构)

## 2.1消息描述

消息由2部份组成，其中包括：消息头 + 消息属性，如下图所示：

|  |
| --- |
| 消息头 |
| 消息属性 |

## 2.2消息格式

* 消息头：STUN消息头为20字节，后面紧跟0或多个属性。STUN头部包含一STUN消息类型、magic cookie、事务ID和消息长度，详情如下图所示：

|  |  |  |
| --- | --- | --- |
| 00 | STUN消息类型 | 消息长度 |
| 魔术字 | | |
| 事务ID（96位） | | |

每个STUN消息的最高位前2位必须为0。当STUN协议为多个协议多路复用时若使用的是同一个端口，这可以用于与其他协议区分STUN数据包。

消息类型确定消息的类别（如请求、成功回应、失败回应、标志）。虽然这里有四种消息类型，但可以分为2类事务：请求/响应事务、标志事务。

消息类型字段可进一步划分为下面结构：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M11 | M10 | M9 | M8 | M7 | C1 | M6 | M5 | M4 | C0 | M3 | M2 | M1 | M0 |

消息类型定义如下：

0b00，表示请求

0b01，表示标志

0b10，表示成功响应

0b11，表示错误响应

魔术字域必须包含固定的值0x2112A442。在RFC3489中，该域是事务ID的一部分。配置魔术字允许服务器检测客户是否理解某些在改进的版本中增加的属性。另外，还可用于STUN多路复用时与其他协议的包进行区分。

96位的事务ID用于唯一的识别STUN事务。对于请求/响应事务，事务ID由STUN客户端来选择；对于标志事务，由代理（代理指支持STUN的客户端或服务器）来选择并发送。它主要服务于与请求相关的响应，因此它也扮演着一个帮助阻止确定类型的攻击的角色。服务器使用事务ID来唯一的标识出所有客户端的每一个事务。事务ID本身必须是唯一的，并且随机的从0到2的96-1次方中选择。重新发送相同的请求时，也必须使用新的事务ID。成功或错误响应必须携带与相对应的请求相同的事务ID。

消息长度字段不包括20字节的STUN头部。所有的STUN属性必须填充为4字节的倍数。消息长度字段的最后2位总是为0，这为区分STUN包与其他协议的包提供了另外一种方法

* 消息属性：

STUN头之后是0或多个属性。每个属性都采用TLV编码，16位的类型、16位的长度及可变长度的值。每个STUN属性必须是4字节边界对齐

|  |  |
| --- | --- |
| 属性类型 | 属性长度 |
| 属性值 | |
| ... | |

属性空间被划分为2个范围。属性的类型值在0x0000到0x7fff是强制理解属性，这意味着除非STUN 代理能够理解这些属性，否则将不能正常处理包含该属性的消息；属性的类型值在0x8000到0xffff 范围是可选理解属性，这意味着如果STUN代理不能理解它们的话这些属性可以被忽略

# 三、协议内容

## 3.1 消息类型

消息类型除了包括标准协议STUN,TURN,ICE中已经定义的，还包含自定义消息类型，用于辅助穿透用的，具体如下。

|  |  |  |
| --- | --- | --- |
| 名称 | 值 | 解释 |
| /////////////标准协议中定义的 | | |
| PJ\_STUN\_BINDING\_REQUEST | 0x0001 | 捆绑请求 |
| PJ\_STUN\_BINDING\_RESPONSE | 0x0101 | 捆绑请求的正确响应 |
| PJ\_STUN\_BINDING\_ERROR\_RESPONSE | 0x0111 | 捆绑请求的错误响应 |
|  |  |  |
| PJ\_STUN\_BINDING\_INDICATION | 0x0011 | Binding Indication (ICE)，用于保活 |
|  |  |  |
| PJ\_STUN\_SHARED\_SECRET\_REQUEST | 0x0002 | STUN SHARED-SECRET reqeust  共享私密请求 |
| PJ\_STUN\_SHARED\_SECRET\_RESPONSE | 0x0102 | Successful response to STUN SHARED-SECRET reqeust |
| PJ\_STUN\_SHARED\_SECRET\_ERROR\_RESPONSE | 0x0112 | Error response to STUN SHARED-SECRET reqeust |
|  |  |  |
| PJ\_STUN\_ALLOCATE\_REQUEST | 0x0003 | STUN/TURN Allocate Request  申请Allocation |
| PJ\_STUN\_ALLOCATE\_RESPONSE | 0x0103 | Successful response to STUN/TURN Allocate Request |
| PJ\_STUN\_ALLOCATE\_ERROR\_RESPONSE | 0x0113 | Failure response to STUN/TURN Allocate Request |
|  |  |  |
| PJ\_STUN\_REFRESH\_REQUEST | 0x0004 | STUN/TURN REFRESH Request  保活/激活 ALLOCATION |
| PJ\_STUN\_REFRESH\_RESPONSE | 0x0104 | Successful response to STUN REFRESH request |
| PJ\_STUN\_REFRESH\_ERROR\_RESPONSE | 0x0114 | Error response to STUN REFRESH request |
|  |  |  |
| PJ\_STUN\_SEND\_INDICATION | 0x0016 | TURN Send indication  ICE中用于探测连通性 |
| PJ\_STUN\_DATA\_INDICATION | 0x0017 | TURN Data indication  Send Indication的回复 |
|  |  |  |
| PJ\_STUN\_CREATE\_PERM\_REQUEST | 0x0008 | TURN CreatePermission request  创建Permission |
| PJ\_STUN\_CREATE\_PERM\_RESPONSE | 0x0108 | TURN CreatePermission successful response |
| PJ\_STUN\_CREATE\_PERM\_ERROR\_RESPONSE | 0x0118 | TURN CreatePermission failure response |
|  |  |  |
| PJ\_STUN\_CHANNEL\_BIND\_REQUEST | 0x0009 | STUN/TURN ChannelBind Request  通道绑定请求 |
| PJ\_STUN\_CHANNEL\_BIND\_RESPONSE | 0x0109 | Successful response to STUN ChannelBind request |
| PJ\_STUN\_CHANNEL\_BIND\_ERROR\_RESPONSE | 0x0119 | Error response to STUN ChannelBind request |
|  | | |
| ///////////////////自定义类型 | | |
| PJ\_STUN\_P2P\_CONNNECT\_REQUEST | 0x1001 | STUN p2p connect request  P2P连接发起 |
| PJ\_STUN\_P2P\_CONNNECT\_RESPONSE | 0x1101 | Successful response to STUN p2p connect request |
| PJ\_STUN\_P2P\_CONNNECT\_ERROR\_RESPONSE | 0x1111 | Error response to STUN p2p connect request |
|  |  |  |
| PJ\_STUN\_P2P\_EXCHANGE\_INFO\_REQUEST | 0x1002 | STUN p2p exchange information request  交换两端信息  ICE中用于交换双方candidate信息 |
| PJ\_STUN\_P2P\_EXCHANGE\_INFO\_RESPONSE | 0x1102 | Successful response to STUN exchange information request |
| PJ\_STUN\_P2P\_EXCHANGE\_INFO\_ERROR\_RESPONSE | 0x1112 | Error response to STUN exchange information request |
|  |  |  |
| PJ\_STUN\_P2P\_DISCONNNECT\_REQUEST | 0x1003 | STUN p2p disconnect request  P2P连接断开 |
|  |  |  |
| PJ\_STUN\_P2P\_CLOSE\_RELAYED\_SOCKET | 0x1004 | STUN p2p close relayed socket |
| PJ\_STUN\_P2P\_CLOSE\_RELAYED\_SOCKET\_RESPONSE | 0x1104 | Response STUN p2p close relayed socket |
|  |  |  |
| PJ\_STUN\_P2P\_REPORT\_SESSION\_INFO | 0x1005 | STUN p2p report session information  报告自身结果 |
| PJ\_STUN\_P2P\_REPORT\_SESSION\_INFO\_RESPONSE | 0x1105 | Response STUN p2p report session information |

## 3.2 消息属性

消息属性空间分为可选部分与强制部分，值超过0x7fff的属性是可选的，即客户或服务器即使不认识该属性也能够处理该消息；值小于或等于0x7fff的属性是强制理解的，即除非理解该属性，否则客户或服务器就不能处理该消息。消息属性除了包括标准协议STUN,TURN,ICE中已经定义的，还包含自定义消息属性，用于辅助穿透用的，具体如下。

|  |  |  |
| --- | --- | --- |
| 名称 | 值 | 解释 |
| ////STUN | | |
| MAPPED-ADDRESS | PJ\_STUN\_ATTR\_MAPPED\_ADDR = 0x0001 | 映射过的IP地址和端口 |
| XOR-MAPPED-ADDRESS | PJ\_STUN\_ATTR\_XOR\_MAPPED\_ADDR = 0x0020 | XOR-MAPPED-ADDRESS属性与MAPPED-ADDRESS属性是相同的，除了这映射后的地址经过了异或处理。（注意，异或运算是其自身的逆运算，再异或一下就可以得出真实的MAPPED-ADDRESS） |
| RESPONSE-ADDRESS | PJ\_STUN\_ATTR\_RESPONSE\_ADDR = 0x0002 | 响应的目的地址 |
| CHANGE-REQUEST | PJ\_STUN\_ATTR\_CHANGE\_REQUEST = 0x0003 | 客户使用32位的CHANGE-REQUEST属性来请求服务器使用不同的地址或端口号来发送响应 |
| SOURCE-ADDRESS | PJ\_STUN\_ATTR\_SOURCE\_ADDR = 0x0004 | SOURCE-ADDRESS属性出现在捆绑响应中，它表示服务器发送响应的源IP地址和端口 |
| CHANGED-ADDRESS | PJ\_STUN\_ATTR\_CHANGED\_ADDR = 0x0005 | 如果捆绑请求的CHANGE-REQUEST属性中的“改变IP”和“改变端口”标志设置了，则CHANGED-ADDRESS属性表示响应发出的IP地址和端口号。 |
| USERNAME | PJ\_STUN\_ATTR\_USERNAME = 0x0006 | USERNAME属性用于消息的完整性检查，用于消息完整性检查中标识共享私密。USERNAME通常出现在共享私密响应中，与PASSWORD一起。当使用消息完整性检查时，可有选择地出现在捆绑请求中 |
| PASSWORD | PJ\_STUN\_ATTR\_PASSWORD = 0x0007 | PASSWORD属性用在共享私密响应中，与USERNAME一起。PASSWORD的值是变长的，用作共享私密，它的长度必须是4字节的倍数，以保证属性与边界对齐 |
| MESSAGE-INTEGRITY | PJ\_STUN\_ATTR\_MESSAGE\_INTEGRITY = 0x0008 | MESSAGE-INTEGRITY属性包含STUN消息的HMAC-SHA1，它可以出现在捆绑请求或捆绑响应中；MESSAGE-INTEGRITY属性必须是任何STUN消息的最后一个属性。它的内容决定了HMAC输入的Key值 |
| ERROR-CODE | PJ\_STUN\_ATTR\_ERROR\_CODE = 0x0009 | ERROR-CODE属性出现在捆绑错误响应或共享私密错误响应中。它的响应号数值范围从100到699。  下面的响应号，与它们缺省的原因语句一起，目前定义如下：  400（错误请求）：请求变形了。客户在修改先前的尝试前不应该重试该请求。  401（未授权）：捆绑请求没有包含MESSAGE-INTERITY属性。  420（未知属性）：服务器不认识请求中的强制属性。  430（过期资格）：捆绑请求没有包含MESSAGE-INTEGRITY属性，但它使用过期  的共享私密。客户应该获得新的共享私密并再次重试。  431（完整性检查失败）：捆绑请求包含MESSAGE-INTEGRITY属性，但HMAC验  证失败。这可能是潜在攻击的表现，或者客户端实现错误  432（丢失用户名）：捆绑请求包含MESSAGE-INTEGRITY属性，但没有  USERNAME属性。完整性检查中两项都必须存在。  433（使用TLS）：共享私密请求已经通过TLS（Transport Layer Security，即安全  传输层协议）发送，但没有在TLS上收到。  500（服务器错误）：服务器遇到临时错误，客户应该再次尝试。  600（全局失败）：服务器拒绝完成请求，客户不应该重试 |
| UNKNOWN-ATTRIBUTES | PJ\_STUN\_ATTR\_UNKNOWN\_ATTRIBUTES = 0x000A | UNKNOWN-ATTRIBUTES属性只存在于其ERROR-CODE属性中的响应号为420的捆绑错误响应或共享私密错误响应中 |
| REFLECTED-FROM | PJ\_STUN\_ATTR\_REFLECTED\_FROM = 0x000B | REFLECTED-FROM属性只存在于其对应的捆绑请求包含RESPONSE-ADDRESS属性的捆绑响应中。属性包含请求发出的源IP地址，它的目的是提供跟踪能力，这样STUN就不能被用作DOS攻击的反射器 |
| XOR-REFLECTED-FROM | PJ\_STUN\_ATTR\_XOR\_REFLECTED\_FROM = 0x0023 | REFLECTED-FROM异或处理 |
| SOFTWARE | PJ\_STUN\_ATTR\_SOFTWARE = 0x8022 | SOFTWARE属性用于代理发送消息时包含版本的描述。它用于客户端和服务器。它的值包括制造商和版本号。该属性对于协议的运行没有任何影响，仅为诊断和调试目的提供服务。SOFTWARE属性是个可变长度的，采用UTF-8编码的小于128个字符的序列号 |
| ALTERNATE-SERVER | PJ\_STUN\_ATTR\_ALTERNATE\_SERVER = 0x8023 | ALTERNATE-SERVER属性标识一个备份的传输地址表明一个STUN客户可以尝试的不同的STUN服务器。属性格式与MAPPED-ADDRESS相同。IP地址族必须与请求的源IP地址的相同 |
| FINGERPRINT | PJ\_STUN\_ATTR\_FINGERPRINT = 0x8028 | INGERPRINT属性可以存在于所有的STUN消息中，提供辅助区分STUN数据包与其他协议数据包的功能。属性的值为采用CRC32方式计算STUN消息直到但不包括FINGERPRINT属性的的结果，并与32位的值0x5354554e异或 |
| REALM | PJ\_STUN\_ATTR\_REALM = 0x0014 | REALM属性可能出现在请求和响应中。在请求中表示长期资格将在认证中使用。当在错误响应中出现表示服务器希望客户使用长期资格来进行认证。 |
| NONCE | PJ\_STUN\_ATTR\_NONCE = 0x0015 | NONCE属性可能出现在请求和响应消息中 |
| ////TURN | | |
| CHANNEL-NUMBER | PJ\_STUN\_ATTR\_CHANNEL\_NUMBER = 0x000C | CHANNEL-NUMBER属性包含通道的号码。属性长4字节，包含16比特的无符号整数和2字节的RFFU（Reserved For Future Use）字段，该字段必须设为0且在接收时被忽略 |
| LIFETIME | PJ\_STUN\_ATTR\_LIFETIME = 0x000D | LIFETIME属性表示服务器在没有收到refresh时维持一个allocation的持续时间。属性长4字节，包含一个32比特的无符号整数值，表示剩余多少秒终止 |
| XOR-PEER-ADDRESS | PJ\_STUN\_ATTR\_XOR\_PEER\_ADDR = 0x0012 | XOR-PEER-ADDRESS指定从TURN服务器看到的对端的地址和端口，例如如果对端是在一个NAT后面，则为对端的server-reflexive传输地址 |
| DATA | PJ\_STUN\_ATTR\_DATA = 0x0013 | DATA属性存在于所有的Send和Data indications消息中。属性的值是可变长度的，包括应用数据。如果属性的长度不上4字节的倍数，必须进行填充 |
| XOR-RELAYED-ADDRESS | PJ\_STUN\_ATTR\_XOR\_RELAYED\_ADDR = 0x0016 | XOR-RELAYED-ADDRESS存在于所有的Allocate响应中。它指定了服务器分配给客户端的地址和端口 |
| REQUESTED-TRANSPORT | PJ\_STUN\_ATTR\_REQ\_TRANSPORT = 0x0019 | 客户端通过该属性为已分配的传输地址请求一个特定的传输协议。属性的值是4字节长度的。协议字段指定了需求的协议。可以取自IPv4报头中的协议字段的值或IPv6报头的下一个报头字段的协议号。目前仅允许设置为17，即UDP。RFFU字段在传输时必须设置为0，并在接收时被忽略。保留用于未来使用 |
| DONT-FRAGMENT | PJ\_STUN\_ATTR\_DONT\_FRAGMENT = 0x001A | 客户端使用该属性来请求服务器设置IP报头中的DF（不要分片）位，当中继应用数据到对端时。该属性没有值，因此属性长度字段为0 |
| ICMP | PJ\_STUN\_ATTR\_ICMP = 0x0030 | 经典路径MTU发现算法有作用 |
| RESERVATION-TOKEN | PJ\_STUN\_ATTR\_RESERVATION\_TOKEN = 0x0022 | RESERVATION-TOKEN属性包含一个token来唯一的标识一个中继传输地址已经被服务器保留。服务器在一个成功响应中包含该属性来告诉客户端这个token，客户端在接下来的Allocate请求中包括该属性来请求服务器为这个allocation使用那个中继传输地址。属性值是8字节长 |
| TIMER-VAL | PJ\_STUN\_ATTR\_TIMER\_VAL = 0x0021 | reserved |
| BANDWIDTH | PJ\_STUN\_ATTR\_BANDWIDTH = 0x0010 | reserved |
| ////ICE |  |  |
| PRIORITY |  | 优先级，ICE中候选人的优先级 |
| USE-CANDIDATE |  | 表示检查的候选对结果要用于传输，该属性没有内容 |
| ICE-CONTROLLED |  | 该属性存在于绑定请求中，表示目前角色处于受控状态，该属性的内容是一个64位无符号整数     ，网络字节顺序，它包含一个随机数，这个数字是用来解决角色冲突的，它被当作一个tie-breaker值。在ICE会话中，对所有的流，所有的绑定请求，ICE代理必须使用相同的编号。当ICE代理重启时，可能会改变这个数字 |
| ICE-CONTROLLING |  | 跟ICE-CONTROLLED一样，不过表示当前角色处于主控状态,主控方选择最终用来通讯的候选地址对，受控方被告知哪个候选地址对用来进行通讯 |
|  |  |  |

# 四、交互流程

标准的消息类型，和属性，穿透流程可查看标准文档，Old STUN (RFC 3489)，STUN RFC 5389，TURN RFC5766，ICE5245,可以到该网址https://datatracker.ietf.org/doc，搜索相应标准文档。PJNATH的文档参考http://www.pjsip.org/pjnath/docs/html/index.htm。下面介绍本文档所实现的P2P穿透流程。

## 4.1 名词解释

为理解本节后面的内容，先对一些字段做一些解释.

|  |  |
| --- | --- |
| Allocation | 通过Allocate请求将中继传输地址提供给客户端，除了中继状态外，还有许可和超时定时器等 |
| Permission | 一个对端允许使用它的IP地址和传输协议来发送数据到TURN服务器，服务器只为从对端发来的并且匹配一个已经存在的许可的流量中继到相应的客户端 |
| Peer Transport Address | 服务器看到的对端的传输地址，当对端是在NAT后面，则是对端的服务器反射传输地址 |
| Server-Reflexive Transport Address | NAT公网侧的传输地址，该地址由NAT分配，相当于一个特定的主机传输地址 |
| Realm | 服务器内用于描述服务器或内容的一个字符串，这个realm告诉客户端哪些用户名和密码的组合可用于认证请求 |
| Nonce | 服务器随机选择的一个字符串，包含在报文摘要中。为了防止中继攻击，服务器应该有规律的改变这个nonce |
| Refresh | 用来保活一个allocation，当refresh返回错误时，需要重新申请Allocation |

## 4.2 p2p

ICE通过结合多个协议（例如STUN和TURN）来工作，并为通信提供多个候选路径，从而最大化成功的机会，但同时也有能力优先考虑候选人，从而使得更昂贵的替代方案（即中继）只能作为最后的手段，否则失败。ICE谈判过程涉及几个阶段：

1.候选人聚会，这里客户端首先会找出所有可用于通信的地址。它可 以找到三种类型的候选地址：1.候选主机的物理网卡的地址2.从STUN 解析出来的服务器反射地址3.从TURN中继分配的中继地址。

2.候选人的优先级,通常，中继候选者将使用最低优先级，因为它是 最昂贵的

3.配对候选人，编码(SDP)这些候选人,发送给远端的PEER，开始进行谈判。它将每个本地候选人与从远程PEER收到的每个远程候选人进行配对。

4.检查每个候选对的连通性,连通性的检查使用STUN Binding request,是一个四次握手过程,双方都需要发送和应答该请求。

穿透的前提是每个终端跟服务器都有一个会话，用来传输各种命令。首先，根据通信五元组，客户端通过一个IP地址和端口的组合来与服务器建立会话，客户端使用TURN命令在服务器上创建和操作一个ALLOCATION。一旦这个allocation创建好了，客户端能够在数据发往哪个对端的指示下发送应用数据到这个服务器，服务器将中继这些数据到合适的对端。

**1.创建ALLOCATION的流程如下**:

1. 向TURN SERVER申请ALLOCATION
2. TRUN SERVER拒绝未认证的请求,返回401(未授权)，并携带REALM指定认证的域，还包括属性NONCE,SOFTWARE
3. 再次向TURN SERVER申请ALLOCATION，携带认证信息
4. TURN SERVER成功创建ALLOCATION

具体流程图:



**2.p2p穿透大致流程如下**:

1. 向对端发起连接
2. 发送捆绑请求
3. 创建ALLOCATION
4. 交换双方候选人信息
5. 开始谈判
6. 收发数据
7. 链路保活

穿透具体流程图:

**3.Udp数据**

参见[4.4节](#_4.4  UDP协议)

4**.每个请求所携带的信息解释如下**:

|  |  |  |
| --- | --- | --- |
| 请求 | 消息类型 | 属性 |
| 向TURN SERVER  申请ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  拒绝ALLOCATION | PJ\_STUN\_ALLOCATE\_ERROR\_RESPONSE | ERROR-CODE:401 Unauthorised  NONCE:随机值NID  REALM:mycompany.org  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 再次向TURN  SERVER申请  ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| TURN SERVER  成功创建  ALLOCATION | PJ\_STUN\_ALLOCATE\_RESPONSE | XOR-RELAYED-ADDRESS:中继地址,119.23.128.208:55063  XOR-MAPPED-ADDRESS:公网映射地址的异或119.123.74.229:22813  LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| REFRESH请求 | PJ\_STUN\_REFRESH\_REQUEST | SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_\_client guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| REFRESH应答 | PJ\_STUN\_REFRESH\_RESPONSE | LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| client向对端发起连接 | PJ\_STUN\_P2P\_CONNNECT\_REQUEST | P2P-CONN-ID:连接ID  P2P-TYPE:1  P2P-LOCAL-USER:p2pc\_\_\_\_client guid  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_\_client guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| device接受连接 | PJ\_STUN\_P2P\_CONNNECT\_RESPONSE | P2P-LOCAL-USER:device uid  P2P-REMOTE-USER:p2pc\_\_\_\_client guid  P2P-CONN-ID:连接ID  P2P-TYPE:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| 终端发起捆绑请求 | PJ\_STUN\_BINDING\_REQUEST | SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  捆绑成功响应 | PJ\_STUN\_BINDING\_RESPONSE | XOR-MAPPED-ADDRESS:119.123.74.229:22276  MAPPED-ADDRESS:119.123.74.229:22276  RESPONSE-ORIGIN:119.23.128.209:34780  OTHER:ADDRESS:10.27.166.149:34781  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 向TURN SERVER  申请ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  P2P-INITIATIVE:1  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  拒绝ALLOCATION | PJ\_STUN\_ALLOCATE\_ERROR\_RESPONSE | ERROR-CODE:401 Unauthorised  NONCE:随机值NID  REALM:mycompany.org  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 再次向TURN  SERVER申请  ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  P2P-INITIATIVE:1  P2P-CONN-ID:连接ID  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| TURN SERVER  成功创建  ALLOCATION | PJ\_STUN\_ALLOCATE\_RESPONSE | XOR-RELAYED-ADDRESS:中继地址,119.23.128.208:55063  XOR-MAPPED-ADDRESS:公网映射地址的异或119.123.74.229:22813  LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 双方交换候选人信息 | PJ\_STUN\_P2P\_EXCHANGE\_INFO\_REQUEST | P2P-LOCAL-INFO:SDP格式打包的候选人信息  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| 交换候选人信息应答 | PJ\_STUN\_P2P\_EXCHANGE\_INFO\_RESPONSE | P2P-LOCAL-INFO:SDP格式打包的候选人信息  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| 开始配对候选人  双方各自往对端候选人发送该消息，来完成配对。  受控方 | PJ\_STUN\_BINDING\_REQUEST | PRIORITY:优先级  ICE-CONTROLLED:受控状态  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:用于消息校验，来源于交换的SDP格式候选人信息中  MESSAGE-INTEGRITY:HMAC-SHA1  FINGERPRINT:CRC-32 |
| 开始配对候选人  双方各自往对端候选人发送该消息，来完成配对。  主控方 | PJ\_STUN\_BINDING\_REQUEST | PRIORITY:优先级  USE-CANDIDATE:  ICE-CONTROLLING:主控状态  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:用于消息校验，来源于交换的SDP格式候选人信息中  MESSAGE-INTEGRITY:HMAC-SHA1  FINGERPRINT:CRC-32 |
| 回复结果：成功 | PJ\_STUN\_BINDING\_RESPONSE | XOR-MAPPED-ADDRESS:192.168.20.63:58982  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  MESSAGE-INTEGRITY:HMAC-SHA1  FINGERPRINT:CRC-32 |
| 失败 | PJ\_STUN\_BINDING\_ERROR\_RESPONSE | ERROR-CODE:487(Role Conflict)角色冲突  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  MESSAGE-INTEGRITY:HMAC-SHA1  FINGERPRINT:CRC-32 |
| 未到达 | 无回复 |  |
| 选择最优路径，完成穿透 | | |
| 心跳 | Udp数据 P2P\_TCP\_TYPE\_HEART |  |
| 收发数据 | Udp数据P2P\_TCP\_TYPE\_DATA |  |

## 4.3 Relay

TURN与STUN的共同点都是通过修改应用层中的私网地址达到NAT穿透的效果，异同点是TURN是通过两方通讯的“中间人”方式实现穿透,即RELAY模式。

1. **RELAY模式的流程如下:**
2. 创建ALLOCATION
3. 创建Permission
4. Send Indication
5. 绑定通道，Channel Bind
6. 收发数据，Channel Data

具体流程图如下:



1. **每个请求所携带的信息解释如下**:

|  |  |  |
| --- | --- | --- |
| 请求 | 消息类型 | 属性 |
| 向TURN SERVER  申请ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  拒绝ALLOCATION | PJ\_STUN\_ALLOCATE\_ERROR\_RESPONSE | ERROR-CODE:401 Unauthorised  NONCE:随机值NID  REALM:mycompany.org  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 再次向TURN  SERVER申请  ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| TURN SERVER  成功创建  ALLOCATION | PJ\_STUN\_ALLOCATE\_RESPONSE | XOR-RELAYED-ADDRESS:中继地址,119.23.128.208:55063  XOR-MAPPED-ADDRESS:公网映射地址的异或119.123.74.229:22813  LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| REFRESH请求 | PJ\_STUN\_REFRESH\_REQUEST | SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_\_client guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| REFRESH应答 | PJ\_STUN\_REFRESH\_RESPONSE | LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| client向对端发起连接 | PJ\_STUN\_P2P\_CONNNECT\_REQUEST | P2P-CONN-ID:连接ID  P2P-TYPE:1  P2P-LOCAL-USER:p2pc\_\_\_\_client guid  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_\_client guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| device接受连接 | PJ\_STUN\_P2P\_CONNNECT\_RESPONSE | P2P-LOCAL-USER:device uid  P2P-REMOTE-USER:p2pc\_\_\_\_client guid  P2P-CONN-ID:连接ID  P2P-TYPE:1  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| 终端发起捆绑请求 | PJ\_STUN\_BINDING\_REQUEST | SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  捆绑成功响应 | PJ\_STUN\_BINDING\_RESPONSE | XOR-MAPPED-ADDRESS:119.123.74.229:22276  MAPPED-ADDRESS:119.123.74.229:22276  RESPONSE-ORIGIN:119.23.128.209:34780  OTHER:ADDRESS:10.27.166.149:34781  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 向TURN SERVER  申请ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  P2P-INITIATIVE:1  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0 |
| TURN SERVER  拒绝ALLOCATION | PJ\_STUN\_ALLOCATE\_ERROR\_RESPONSE | ERROR-CODE:401 Unauthorised  NONCE:随机值NID  REALM:mycompany.org  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 再次向TURN  SERVER申请  ALLOCATION | PJ\_STUN\_ALLOCATE\_REQUEST | REQUESTED-TRANSPORT:UDP  P2P-ASSIST:1  P2P-INITIATIVE:1  P2P-CONN-ID:连接ID  P2P-REMOTE-USER:device uid  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| TURN SERVER  成功创建  ALLOCATION | PJ\_STUN\_ALLOCATE\_RESPONSE | XOR-RELAYED-ADDRESS:中继地址,119.23.128.208:55063  XOR-MAPPED-ADDRESS:公网映射地址的异或119.123.74.229:22813  LIFTTIME:600  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| Create Permission | PJ\_STUN\_CREATE\_PERM\_REQUEST | XOR-PEER-ADDRESS:对端候选人集  ...  XOR-PEER-ADDRESS  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| 成功创建  Permission | PJ\_STUN\_CREATE\_PERM\_RESPONSE | SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| Send Indication | PJ\_STUN\_SEND\_INDICATION | XOR-PEER-ADDRESS:对端地址119.23.128.209:52904  DATA: |
| Data Indication | PJ\_STUN\_DATA\_INDICATION | DATA:  XOR-PEER-ADDRESS:对端地址119.23.128.209:52904  SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 绑定通道  Channel Bind | PJ\_STUN\_CHANNEL\_BIND\_REQUEST | Channel-Number:0x4000,可用的通道号  X0R-PEER-ADDRESS:对端地址  SOFTWARE:p2pc\_\_\_pjnath-2.3.0  USERNAME:p2pc\_\_\_CLIENT guid  REALM:mycompany.org  NONCE:NID  MESSAGE-INTEGRITY:HMAC-SHA1 |
| 绑定通道成功 | PJ\_STUN\_CHANNEL\_BIND\_RESPONSE | SOFTWARE:Citrix-3.2.5.2 ‘Marshal West’ |
| 收发数据，非STUN消息 | | |
| Channel Data |  | Channel\_data+udp数据 |
| Channel Data |  |  |

1. **Channel Data**

Channel Data=pj\_turn\_channel\_data+UDP数据

pj\_turn\_channel\_data格式:

|  |  |  |  |
| --- | --- | --- | --- |
| pj\_turn\_channel\_data总共占用4字节 | | | |
| 字段名称 | 长度(字节) | 范围 | 描述 | |
| Channel Number | 2 | Unsigned short | 网络字节序,通道号 | |
| Payload Length | 2 | Unsigned short | 网络字节序,数据长度 | |
|  |  |  |  | |
|  |  |  |  | |

UDP数据格式:参见[4.4节](#_4.4  UDP协议)

## 4.4 UDP协议

此代码总共实现了两种协议，一种叫UDT，该协议未使用，不做介绍，请自行google；另一种，用udp模拟了tcp的实现，主要参考tcp协议，用udp协议实现了可靠传输，主要实现要点：1.确保没有乱序 2.确保不丢包 3.根据网络情况进行实时流控

1. 怎么确保没有乱序，发送的数据都携带seq number，在接收端确保该number连续，不连续，通过ack机制，告知对方；对方根据情况进行重传
2. 怎么确保不丢包，超时重传，在rto时间内没有收到ack，就对相应的包进行重传；或者通过接收到的ack，来决定重传哪些包。接收端应答ack时，如果接收到不连续的包，则ack中会携带所有不连续的包的起始和结束seq number，发送端跟据这个seq number来发送需要重传的包
3. 怎么实现流控，使用类似tcp滑动窗口的机制来控制发包的速率，滑动窗口大小取决于拥塞窗口cwnd和对端接收窗口peer\_wnd的大小。拥塞窗口cwnd的变化在拥塞处理过程中体现的，协议还是实现了”延迟ACK”来提高效率，具体参见4.4.2节

### 4.4.1 报文格式

Udp数据报报文格式=消息头+消息体

消息头格式:

|  |  |  |  |
| --- | --- | --- | --- |
| 消息头总共占用16字节 | | | |
| 字段名称 | 长度(字节) | 范围 | 描述 | |
| Type | 1 | 0xA1: 数据  0xA2: 响应  0xA3: 窗口大小  0xA4: 心跳 | 数据类型 | |
| seq | 4 | Unsigned int  (0~4294967295) | 网络字节序,包序号 | |
| len | 2 | 0~MSS-消息头  (1360-16) | 网络字节序,数据长度 | |
| Wnd\_size | 2 | Unsigned short  ( 0——65535) | 网络字节序,窗口大小 | |
| ack | 4 | Unsigned int  (0~4294967295) | 网络字节序,应答 | |
| padding | 3 | Array[3] | 填充字节，暂时无用,为0 | |

消息体:

二进制数据

### 4.4.2 协议实现

**1.流控**

根据窗口wnd的大小来实现流控，wnd的大小表示每次最大可以发送的udp报文的包数。具体计算方式如下：

Wnd = [min(Peer\_wnd\_size,Cwnd) - (cur.seq-begin.seq)] \* 2;

Wnd = min(Wnd,Sendpkg);

以上个名词的解释如下表:

|  |  |  |
| --- | --- | --- |
| 名词 | 值 | 解释 |
| Wnd | ？ | 当前可发送的最大udp报文包数 |
| cur.seq | ？ | 当前发送的包序号 |
| begin.seq | ？ | 第一包发送的包序号 |
| Sendpkg | ？ | 一次发送的包数 |
| Peer\_wnd\_size | 随时改变 | 对端可接收的窗口大小 |
| Cwnd | 随时改变 | 拥塞窗口大小 |
| Peer\_wnd\_size | P2P\_TCP\_RECV\_BUFFER\_COUNT - recved\_user\_data\_count | 对端窗口大小 |
| recved\_user\_data\_count | 最大连续序号 - 拷贝完的数据 | 收到的数据包数 |
| Ack | 最后收到的连续的seq+1 | ack |
| P2P\_TCP\_RECV\_BUFFER\_COUNT | 64 | 接收窗口大小，一个宏 |
| MIN\_SSTHRESH | 16 | Ssthresh最小值，一个宏 |
| CWND\_INIT\_VAL | 10 | Cwnd初始值，一个宏 |
| MIN\_FAST\_ACK | 2 | Fast ack 最小值，一个宏 |
| MAX\_FAST\_ACK | 4 | Fast ack 最大值，一个宏 |
| DELAY\_ACK\_MIN | 40 | 延迟ack时间，ato = 40ms，一个宏 |
|  |  |  |
|  |  |  |

**2.拥塞控制**

拥塞控制包括三部分: 1.慢启动 2.拥塞避免 3.快速恢复。

拥塞处理的具体过程为：

首先进入慢启动，在慢启动时有两种情况:

1. 当cwnd达到一个阀值ssthresh时，进入拥塞避免;
2. 当超时重传，进入慢启动。

在拥塞避免时，会有两种情况

1. 当收到3次相同ack时，进入快速重传；
2. 当超时重传时，进入慢启动。

在快速重传时，有两种情况:

1. 当接收到新的ack，即重传的包对端已经收到，重新进入拥塞避免；
2. 当出现超时重传时，同样进入慢启动.

下表列出一些初始值：

|  |  |  |
| --- | --- | --- |
| 名词 | 值 | 解释 |
| cwnd | 10 | 拥塞窗口 |
| ssthresh | 64 | 慢启动门槛 |
| rto | 200ms | 超时重传时间 |
| ato | 40ms | 延迟ack时间 |
| peer\_wnd\_size | 64 | 对端窗口,初始大小为64 |
| send\_seq | 1 | 发送序列号，初始值为1 |
| last\_recv\_ack | 最后收到的连续的seq+1 | ack |
| sack | 接收到的乱序数据包的所有起始位置。例如：  Order list：  Seq = 1，  Seq = 2，  Disorder list：  Seq = 4，  Seq = 5，  Seq = 7，  Seq = 8，  Seq = 9，  Seq = 10  则此时sack=4，7，10 | 告诉对端哪些包没有接收到 |

**慢启动:**

开始时，首先进入慢启动模式，在该模式中每经过一个rtt，cwnd+1，由于开启了delay ack，所以cwnd=cwnd+last\_recv\_ack。

慢启动时序图如下：



**拥塞避免:**

在慢启动下,当cwnd>ssthresh时，即达到慢启动阀值时进入拥塞避免，此时每经过一个rtt，cwnd++，当收到相同的ack时cwnd保持不变，当连续3此收到相同的ack，进入快速恢复模式.

正常拥塞避免时序图:

 拥塞避免进入快速重传时序图如下:



**快速恢复:**

在拥塞避免下,当连续3此收到相同的ack，进入快速恢复模式,此时初始状态为:ssthresh=max(cwnd/2,MIN\_SSTHRESH); cwnd=1, 然后快速重传丢掉的包。该模式下每经过一个rtt，cwnd++，当收到新ack时，重新进入拥塞避免，cwnd=sshthresh。

快速恢复进入拥塞避免时序图如下:

参考上图(拥塞避免进入快速重传时序图)

**超时重传:**

在以上三个过程中，都有可能出现超时的情况,一旦出现超时的情况,自动进入慢启动模式,ssthresh = max(cwnd/2,MIN\_SSTHRESH);dup\_ack = 0;cwnd = CWND\_INIT\_VAL

**Delay Ack/fast ack:**

接收数据时回复ACK有三种情况：

1. delay ack
2. Fast ack
3. 直接回复

接收端在每次收到数据时，开启delay ack定时器，定时时间为ato，在定时回调中进行ack回复。当接收到相同的数据时，取消delay ack ，开始fast ack。当接收的不连续的数据时，直接回复ack。如果连续接收到的新的包又连续了，进入delay ack模式。

**Rto计算公式**

//to see linux tcp tcp\_rtt\_estimator function source code

/\*

SRTT = SRTT+0.125(RTT-SRTT)

DevRTT = (1-0.25)\*DevRTT+ 0.25\*(|RTT-SRTT|)

RTO = SRTT + 4\*DevRTT

p2p\_tcp\_sock->srtt = SRTT\*8

p2p\_tcp\_sock->mdev = DevRTT\*4

p2p\_tcp\_sock->mdev\_max is max p2p\_tcp\_sock->mdev in cwnd

\*/

**Ato 计算公式**

Delta = 两次接收到的ack时间差

/\*to see linux source tcp\_event\_data\_recv \*/

if (delta <= DELAY\_ACK\_MIN / 2)

{

sk->ato = (sk->ato >> 1) + DELAY\_ACK\_MIN / 2;

}

else if (delta < sk->ato)

{

sk->ato = (sk->ato >> 1) + delta;

if (sk->ato > sk->rto)

sk->ato = sk->rto;

}

else if (delta > sk->ato)

{

//两次接收数据的时间差超过了ato时间，直接开始fast ack

p2p\_tcp\_enter\_fast\_ack(sk);

}