

# 无脑仔的小明

----- 努力变得更好, 让我喜欢的人, 喜欢我.



昵称：无脑仔的小明  
园龄：5年4个月  
粉丝：229  
关注：17  
[+加关注](#)

<	2018年5月						>
日	一	二	三	四	五	六	
29	30	1	2	3	4	5	
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30	31	1	2	
3	4	5	6	7	8	9	

## 搜索

谷歌搜索

## 我的标签

Linux(42)  
C(37)  
网络编程(31)  
Socket(28)  
环境配置(22)  
物联网(22)  
服务器(21)  
年度月度总结(11)  
感悟人生(9)  
java(9)  
[更多](#)

## 随笔分类

Egret(1)  
Netty网络编程(4)  
React Native(1)  
Socket系列(30)  
Spring XXX

[首页](#) [新随笔](#) [联系](#) [管理](#)

随笔- 126 文章- 0 评论- 192

### udp穿透简单讲解和实现(Java)

在上一小节中了解到了通过浏览器自带的Webrtc功能来实现P2P视频聊天。在HTML5还没有普及和制定Webrtc标准的前提下，如果要在手机里进行视频实时对话等包括其他功能的话，还是要自己实现，还比较好扩展。所以本次要了解一下udp进行穿透(打洞)。

还是进入正题吧，了解P2P。

#### 1. 原理

关于原理网上随便就可以找到好多资料了。大部分都是讲解原理的，还配了图，还是不错的。这里不细说。



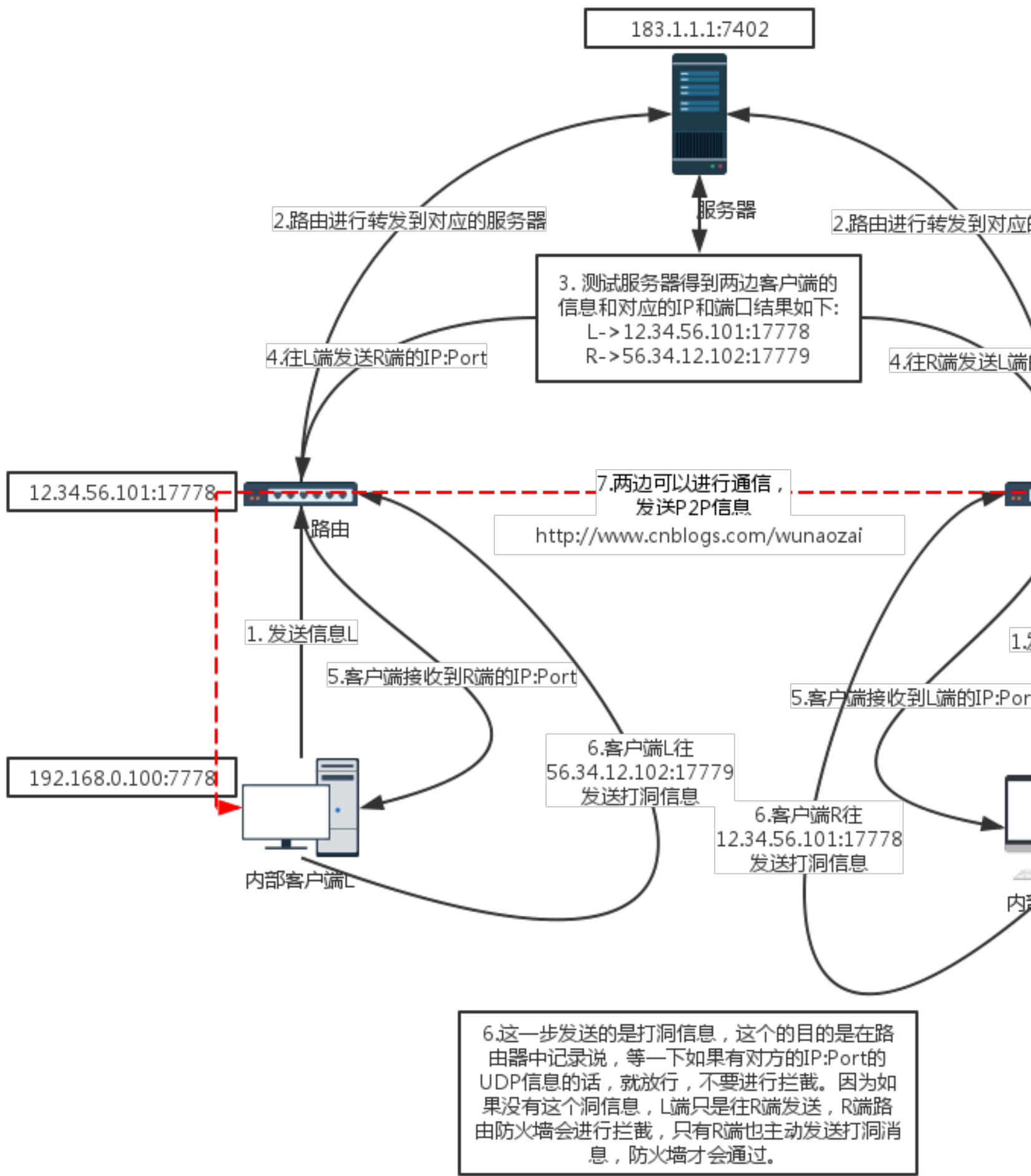
随笔档案

- 2018年4月 (1)
- 2018年3月 (3)
- 2018年2月 (5)
- 2018年1月 (9)
- 2017年12月 (6)
- 2017年10月 (1)
- 2017年8月 (1)
- 2017年6月 (1)
- 2017年3月 (1)
- 2017年2月 (1)
- 2016年11月 (2)
- 2016年6月 (1)
- 2016年5月 (1)
- 2016年3月 (3)
- 2015年12月 (4)
- 2015年11月 (2)
- 2015年6月 (4)
- 2015年5月 (5)
- 2015年1月 (2)
- 2014年12月 (1)
- 2014年11月 (1)
- 2014年10月 (1)
- 2014年9月 (7)
- 2014年8月 (21)
- 2014年7月 (12)
- 2014年6月 (1)
- 2014年5月 (5)
- 2014年4月 (7)
- 2014年3月 (6)
- 2014年2月 (1)
- 2014年1月 (1)
- 2013年10月 (1)
- 2013年7月 (2)
- 2013年6月 (1)
- 2013年5月 (1)
- 2013年4月 (1)
- 2013年1月 (3)

积分与排名

积分 - 150474  
排名 - 1852

阅读排行榜



2. 代码讲解

本次使用Java语言。网络框架使用Netty4，其实这些都是次要的，原理看懂才是关键。

服务器代码EchoServer.java

```
1 package com.jieli.nat.echo;
2
3 import io.netty.bootstrap.Bootstrap;
4 import io.netty.channel.ChannelOption;
5 import io.netty.channel.EventLoopGroup;
6 import io.netty.channel.nio.NioEventLoopGroup;
7 import io.netty.channel.socket.nio.NioDatagramChannel;
8
9 public class EchoServer {
10
11     public static void main(String[] args) {
12         Bootstrap b = new Bootstrap();
13         EventLoopGroup group = new NioEventLoopGroup();
14         try {
15             b.group(group)
16               .channel(NioDatagramChannel.class)
17               .option(ChannelOption.SO_BROADCAST, true)
```



```
18         .handler(new EchoServerHandler());
19
20         b.bind(7402).sync().channel().closeFuture().await();
21     } catch (Exception e) {
22         e.printStackTrace();
23     } finally{
24         group.shutdownGracefully();
25     }
26
27 }
28 }
```



## 服务器代码EchoServerHandler.java



```
1 package com.jieli.nat.echo;
2
3 import java.net.InetAddress;
4 import java.net.InetSocketAddress;
5
6 import io.netty.buffer.ByteBuf;
7 import io.netty.buffer.Unpooled;
8 import io.netty.channel.ChannelHandlerContext;
9 import io.netty.channel.SimpleChannelInboundHandler;
10 import io.netty.channel.socket.DatagramPacket;
11
12 public class EchoServerHandler extends SimpleChannelInboundHandler<DatagramPacket>{
13
14     boolean flag = false;
15     InetSocketAddress addr1 = null;
16     InetSocketAddress addr2 = null;
17     /**
18      * channelRead0 是对每个发送过来的UDP包进行处理
19      */
20     @Override
21     protected void channelRead0(ChannelHandlerContext ctx, DatagramPacket packet)
22         throws Exception {
23         ByteBuf buf = (ByteBuf) packet.copy().content();
24         byte[] req = new byte[buf.readableBytes()];
25         buf.readBytes(req);
26         String str = new String(req, "UTF-8");
27         if(str.equalsIgnoreCase("L")){
28             //保存到addr1中 并发送addr2
29             addr1 = packet.sender();
30             System.out.println("L 命令, 保存到addr1中 ");
31         }else if(str.equalsIgnoreCase("R")){
32             //保存到addr2中 并发送addr1
33             addr2 = packet.sender();
34             System.out.println("R 命令, 保存到addr2中 ");
35         }else if(str.equalsIgnoreCase("M")){
36             //addr1 -> addr2
37             String remot = "A " + addr2.getAddress().toString().replace("/", " ")
38                 + " "+addr2.getPort();
39             ctx.writeAndFlush(new DatagramPacket(
40                 Unpooled.copiedBuffer(remot.getBytes()), addr1));
41             //addr2 -> addr1
42             remot = "A " + addr1.getAddress().toString().replace("/", " ")
43                 + " "+addr1.getPort();
44             ctx.writeAndFlush(new DatagramPacket(
45                 Unpooled.copiedBuffer(remot.getBytes()), addr2));
46             System.out.println("M 命令");
47         }
48
49     }
50
51     @Override
52     public void channelActive(ChannelHandlerContext ctx) throws Exception {
```

```
53         System.out.println("服务器启动...");
54
55         super.channelActive(ctx);
56     }
57 }
```



## 左边客户端EchoClient.java



```
1 package com.jieli.nat.echo;
2
3 import io.netty.bootstrap.Bootstrap;
4 import io.netty.channel.ChannelOption;
5 import io.netty.channel.EventLoopGroup;
6 import io.netty.channel.nio.NioEventLoopGroup;
7 import io.netty.channel.socket.nio.NioDatagramChannel;
8
9 /**
10  * 模拟P2P客户端
11  * @author
12  *
13  */
14 public class EchoClient{
15
16     public static void main(String[] args) {
17         int port = 7778;
18         if(args.length != 0){
19             port = Integer.parseInt(args[0]);
20         }
21         Bootstrap b = new Bootstrap();
22         EventLoopGroup group = new NioEventLoopGroup();
23         try {
24             b.group(group)
25                 .channel(NioDatagramChannel.class)
26                 .option(ChannelOption.SO_BROADCAST, true)
27                 .handler(new EchoClientHandler());
28
29             b.bind(port).sync().channel().closeFuture().await();
30         } catch (Exception e) {
31             e.printStackTrace();
32         } finally{
33             group.shutdownGracefully();
34         }
35     }
36 }
```



## 左边客户端EchoClientHandler.java



```
1 package com.jieli.nat.echo;
2
3 import java.net.InetSocketAddress;
4 import java.util.Vector;
5
6 import io.netty.buffer.ByteBuf;
7 import io.netty.buffer.Unpooled;
8 import io.netty.channel.ChannelHandlerContext;
9 import io.netty.channel.SimpleChannelInboundHandler;
10 import io.netty.channel.socket.DatagramPacket;
11
12 //L
```



```
13 public class EchoClientHandler extends SimpleChannelInboundHandler<DatagramPacket>{
14
15     @Override
16     protected void channelRead0(ChannelHandlerContext ctx, DatagramPacket packet)
17         throws Exception {
18         //服务器推送对方IP和PORT
19         ByteBuf buf = (ByteBuf) packet.content();
20         byte[] req = new byte[buf.readableBytes()];
21         buf.readBytes(req);
22         String str = new String(req, "UTF-8");
23         String[] list = str.split(" ");
24         //如果是A 则发送
25         if(list[0].equals("A")){
26             String ip = list[1];
27             String port = list[2];
28             ctx.writeAndFlush(new DatagramPacket(
29                 Unpooled.copiedBuffer("打洞信息".getBytes()), new
30 InetSocketAddress(ip, Integer.parseInt(port))));
31             Thread.sleep(1000);
32             ctx.writeAndFlush(new DatagramPacket(
33                 Unpooled.copiedBuffer("P2P info..".getBytes()), new
34 InetSocketAddress(ip, Integer.parseInt(port))));
35         }
36         System.out.println("接收到的信息:" + str);
37     }
38
39     @Override
40     public void channelActive(ChannelHandlerContext ctx) throws Exception {
41         System.out.println("客户端向服务器发送自己的IP和PORT");
42         ctx.writeAndFlush(new DatagramPacket(
43             Unpooled.copiedBuffer("L".getBytes()),
44             new InetSocketAddress("183.1.1.1", 7402)));
45         super.channelActive(ctx);
46     }
47 }
```



### 右边客户端EchoClient2.java



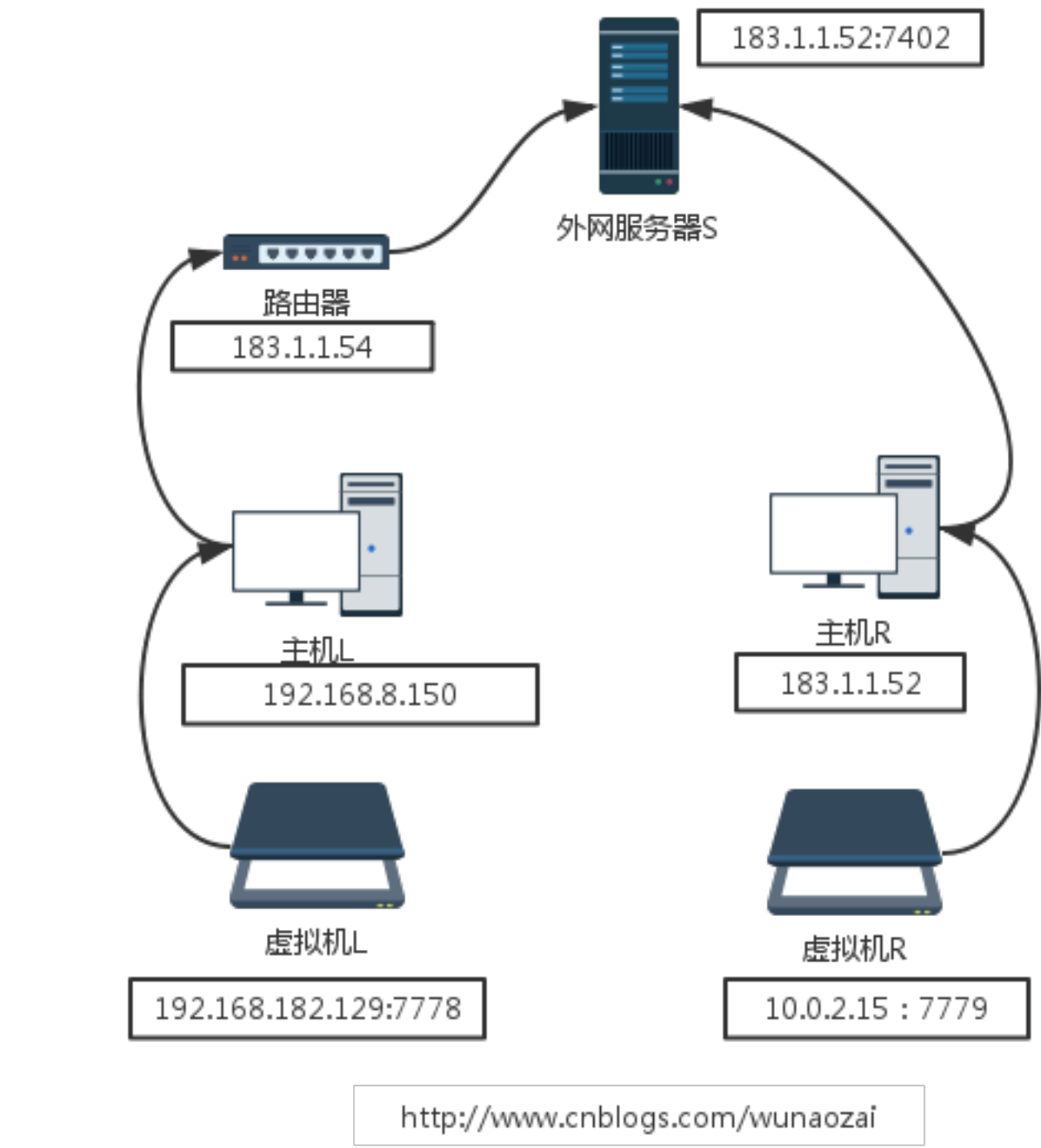
### 右边客户端EchoClientHandler2.java



## 3. 实验环境模拟

实验环境：1台本地主机L，里面安装虚拟机L，地址192.168.182.129. 通过路由器183.1.1.54上网。 1台服务器主机S，服务器地址183.1.1.52:7402, 同时服务器里安装虚拟机R，地址10.0.2.15 .由于外网地址只有两个，所以这能这样测试。通过虚拟机也是可以模拟出测试环境的。 图示如下：





三台测试机ip如下

```
myuser@debian-srv: ~/workspace/iso
^C
--- 10.0.1.15 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1007ms

root@debian-srv:~/workspace/NATTest# java -jar echoserver.jar
服务器启动...
^Croot@debian-srv:~/workspace/NATTest# ifconfig
eth1      Link encap:Ethernet  HWaddr 82:00:00:08:00:05
          inet addr:183.1.1.54  Bcast:183.1.1.255  Mask:255.255.255.240
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14183137 errors:0 dropped:27 overruns:0 frame:0
          TX packets:20530785 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6631800998 (6.1 GiB)  TX bytes:27898358453 (25.9 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1043907 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1043907 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:26775646251 (24.9 GiB)  TX bytes:26775646251 (24.9 GiB)

root@debian-srv:~/workspace/NATTest#
```

```
root@localhost:~/workspace/NATTest
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:307491 errors:0 dropped:0 overruns:0 frame:0
TX packets:307491 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:174838088 (166.7 MiB)  TX bytes:174838088 (166.7 MiB)

root@localhost NATTest]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:D4:8B:67
          inet addr:192.168.182.129  Bcast:192.168.182.255  Mask:255.255.255.0
          inet6 addr: fe80::20e:29ff:fe4d:8b67/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:336186 errors:0 dropped:0 overruns:0 frame:0
          TX packets:273269 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:189628571 (180.8 MiB)  TX bytes:159603364 (152.2 MiB)
          Interrupt:19 Base address:0x2024

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:307491 errors:0 dropped:0 overruns:0 frame:0
          TX packets:307491 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:174838088 (166.7 MiB)  TX bytes:174838088 (166.7 MiB)

root@localhost NATTest]#
```

```
CentOS-01 [Running] - Oracle VM VirtualBox
Machine View Devices Help
RX packets:0 errors:0 dropped:0
TX packets:0 errors:0 dropped:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0

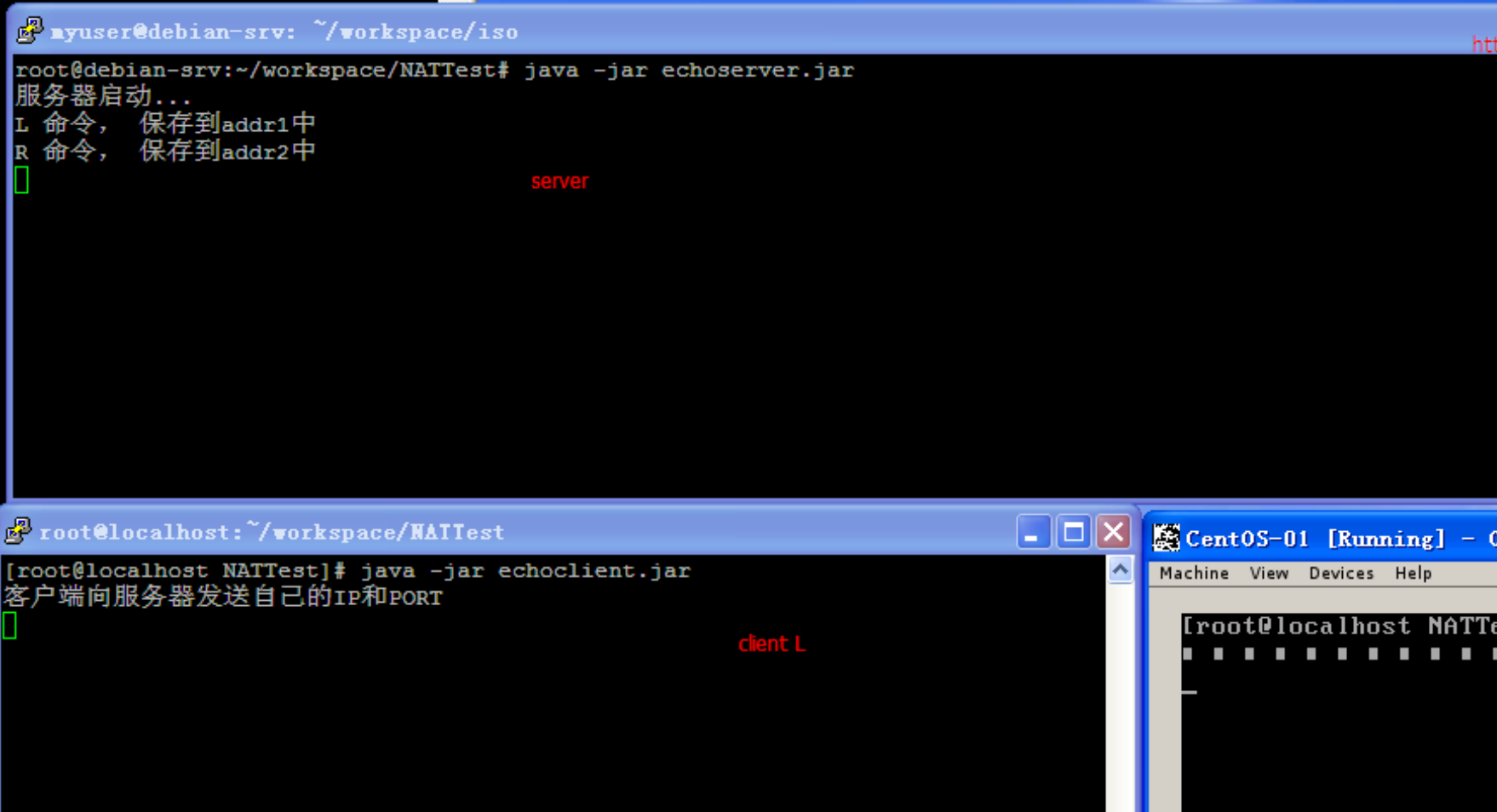
[root@localhost NATTest]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:FE:4D:8B
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4d:8b67/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:319672 errors:0 dropped:0 overruns:0 frame:0
          TX packets:154732 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:307957246 (293.6 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b) TX bytes:0

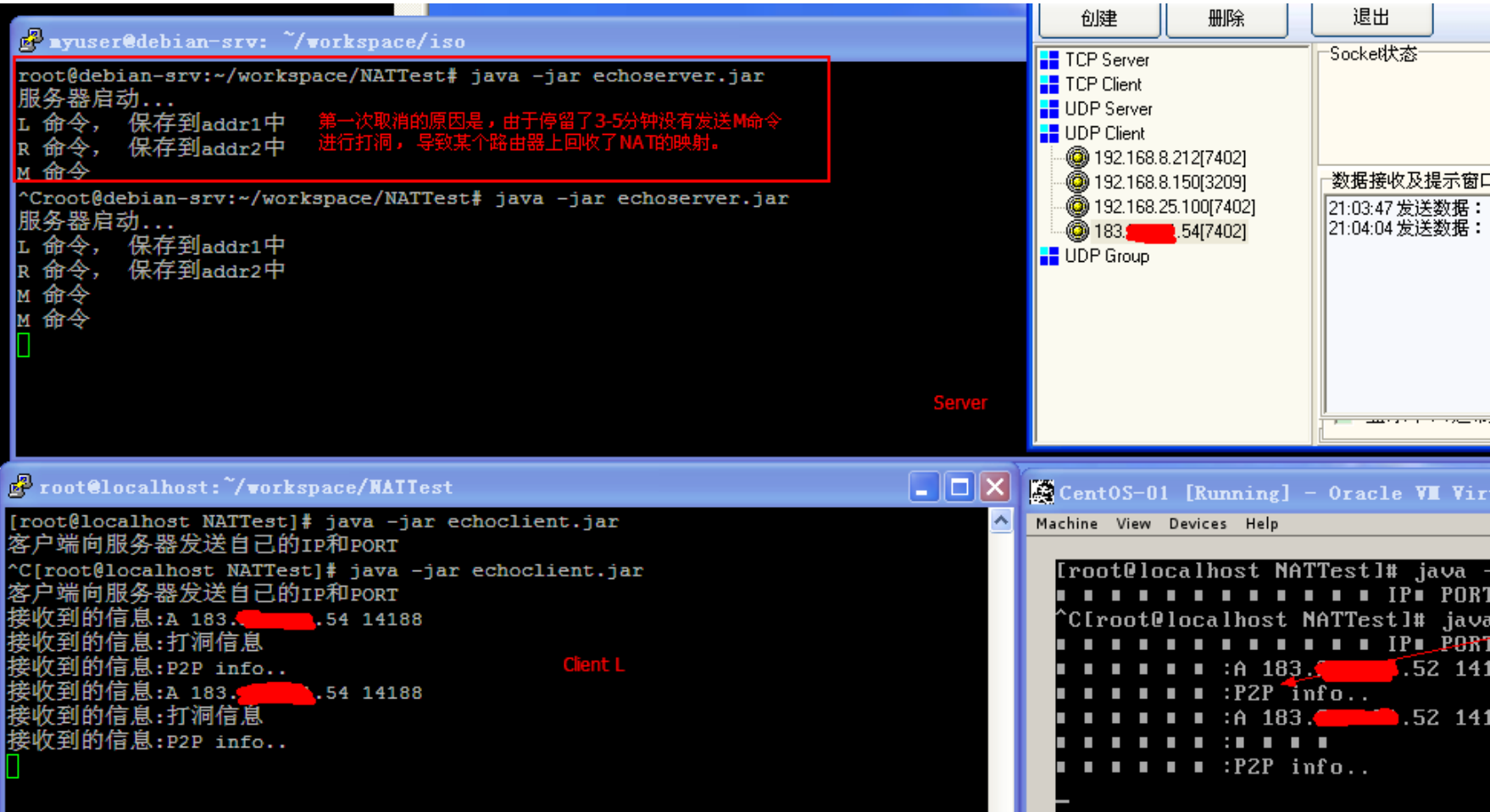
[root@localhost NATTest]#
```

三台测试机器分别启动





然后通过第三方工具发送一个M指定到服务器



一般路由器的缓存会保存一小段时间，具体跟路由器有关。

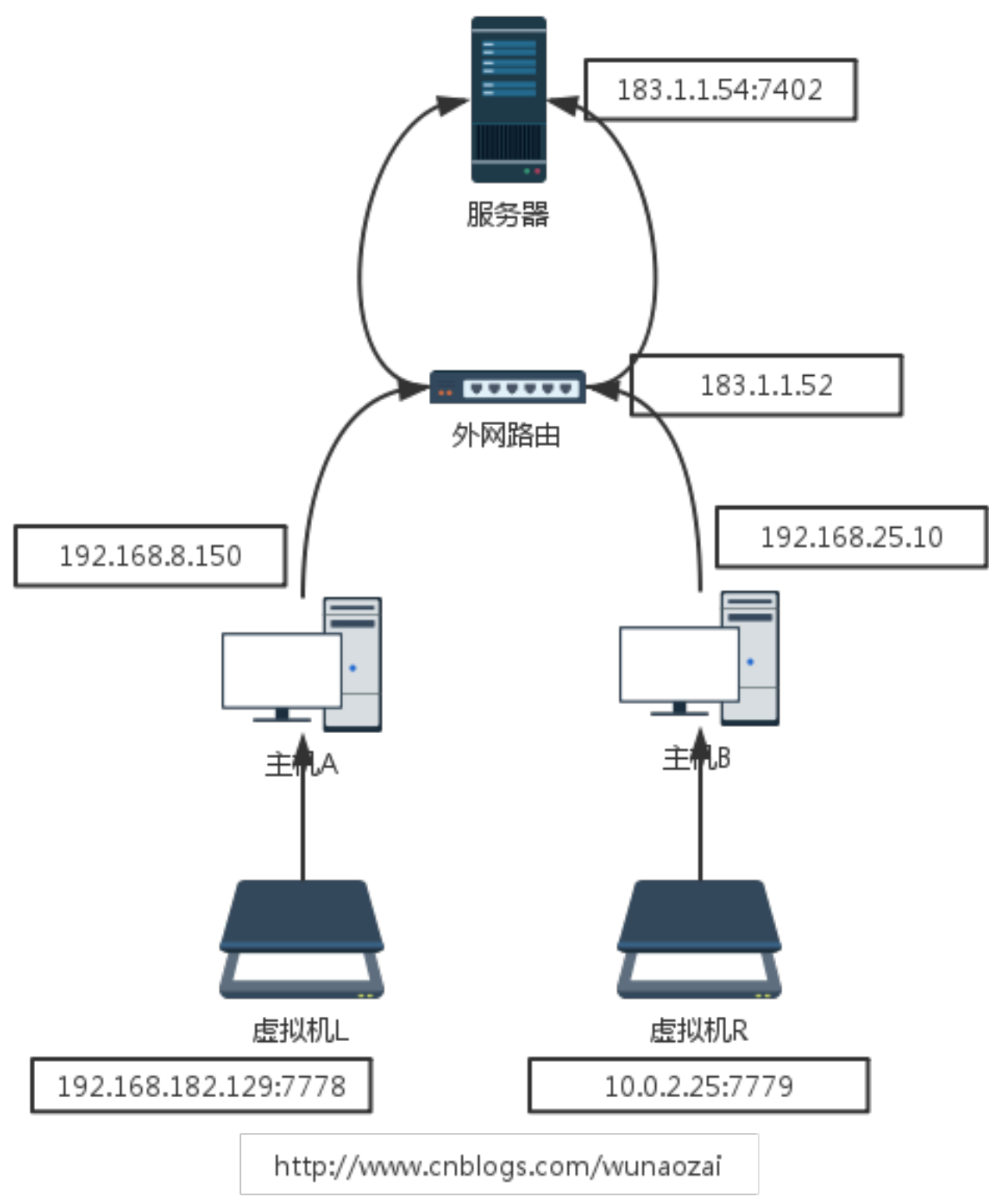
关于Client R会少接收到一个"打洞消息"这个信息。不是因为UDP的丢包，是Client L 发送的打洞命令。简单说一下。一开始ClientL发送一个UDP到Server，此时ClientL的路由器会保留这样的一条记录(ClientL:IP:Port->Server:IP:Port) 所以Server:IP:Port发送过来的信息，ClientL路由器没有进行拦截，所以可以接收得到。但是ClientR:IP:Port发送过来的消息在ClientL的路由器上是没有这一条记录的，所以会被拒绝。此时ClientL主动发送一条打洞消息(ClientL:IP:Port->ClientR:IP:Port)，使ClientL路由器保存一条记录。使ClientR可以通过指定的IP:Port发送信息过来。不过ClientL的这条打洞信息就不一定能准确的发送到ClientR。原因就是，同理，ClientR路由器上没有ClientL的记录。

由于ClientL ClientR路由器上都没有双方的IP:port，所以通过这样的打洞过程。

我觉得我这样描述还是比较难懂的。如果还不了解，请另外参考其他网上资料。

还有一个就是搭建这样的测试环境还是比较麻烦的。注意如果你只有一台电脑，然后搭建成下面这种测试环境，一般是不行的。因为ClientL和ClientR是通过183.1.1.52路由器进行数据的P2P传输，一般路由器会拒绝掉这种回路的UDP包。





这个时候就要进行内网的穿透了。这个就要像我上一篇博客里面的Webrtc是如何通信一样的了，要通过信令来交换双方信息。

```
{
  "eventName": "ice candidate",
  "data": {
    "label": 1,
    "candidate": "candidate:713629939 2 tcp 1518214910 192.168.8.150 0 typ host tcptype active generation 0",
    "label": 1,
    "candidate": "candidate:1630307812 2 tcp 1518149374 192.168.25.11 0 typ host tcptype active generation 0",
    "label": 1,
    "candidate": "candidate:416293236 2 tcp 1518083838 192.168.182.1 0 typ host tcptype active generation 0",
    "label": 0,
    "candidate": "candidate:800028948 1 udp 2122129151 192.168.25.11 3006 typ host generation 0",
    "label": 0,
    "candidate": "candidate:1448336772 1 udp 2122063615 192.168.182.1 3007 typ host generation 0",
    "label": 0,
    "candidate": "candidate:376141290 2 udp 2122260222 192.168.229.1 3008 typ host generation 0",
    "label": 0,
    "candidate": "candidate:1678433283 2 udp 2122194686 192.168.8.150 3009 typ host generation 0",
    "label": 0,
    "candidate": "candidate:800028948 2 udp 2122129150 192.168.25.11 3010 typ host generation 0",
    "label": 0,
    "candidate": "candidate:1448336772 2 udp 2122063614 192.168.182.1 3011 typ host generation 0",
    "label": 2,
    "candidate": "candidate:376141290 1 udp 2122260223 192.168.229.1 3012 typ host generation 0",
    "label": 2,
    "candidate": "candidate:1678433283 1 udp 2122194687 192.168.8.150 3013 typ host generation 0",
    "label": 0,
    "candidate": "candidate:1491634458 1 tcp 1518280447 192.168.229.1 0 typ host tcptype active generation 0",
    "label": 0,
    "candidate": "candidate:713629939 1 tcp 1518214911 192.168.8.150 0 typ host tcptype active generation 0",
    "label": 0,
    "candidate": "candidate:1630307812 1 tcp 1518149375 192.168.25.11 0 typ host tcptype active generation 0",
    "label": 0,
    "candidate": "candidate:416293236 1 tcp 1518083839 192.168.182.1 0 typ host tcptype active generation 0",
    "label": 2,
    "candidate": "candidate:800028948 1 udp 2122129151 192.168.25.11 3014 typ host generation 0",
    "label": 2,
    "candidate": "candidate:1448336772 1 udp 2122063615 192.168.182.1 3015 typ host generation 0",
    "label": 1,
    "candidate": "candidate:376141290 1 udp 2122260223 192.168.229.1 3016 typ host generation 0",
    "label": 1,
    "candidate": "candidate:1678433283 1 udp 2122194687 192.168.8.150 3017 typ host generation 0",
    "label": 1,
    "candidate": "candidate:800028948 1 udp 2122129151 192.168.25.11 3018 typ host generation 0",
    "label": 1,
    "candidate": "candidate:1448336772 1 udp 2122063615 192.168.182.1 3019 typ host generation 0",
    "label": 1,
    "candidate": "candidate:376141290 2 udp 2122260222 192.168.229.1 3020 typ host generation 0",
    "label": 1,
    "candidate": "candidate:1678433283 2 udp 2122194686 192.168.8.150 3021 typ host generation 0",
    "label": 1,
    "candidate": "candidate:800028948 2 udp 2122129150 192.168.25.11 3022 typ host generation 0",
    "label": 1,
    "candidate": "candidate:1448336772 2 udp 2122063614 192.168.182.1 3023 typ host generation 0"
  }
}
```

就是发送包括自己内网的所有IP，支持TCPUDP等其他信息封装成信令发送到服务器然后转发到另一端的客户端。使客户端可以对多个IP:Port进行尝试性连接。这个具体的就不展开了。

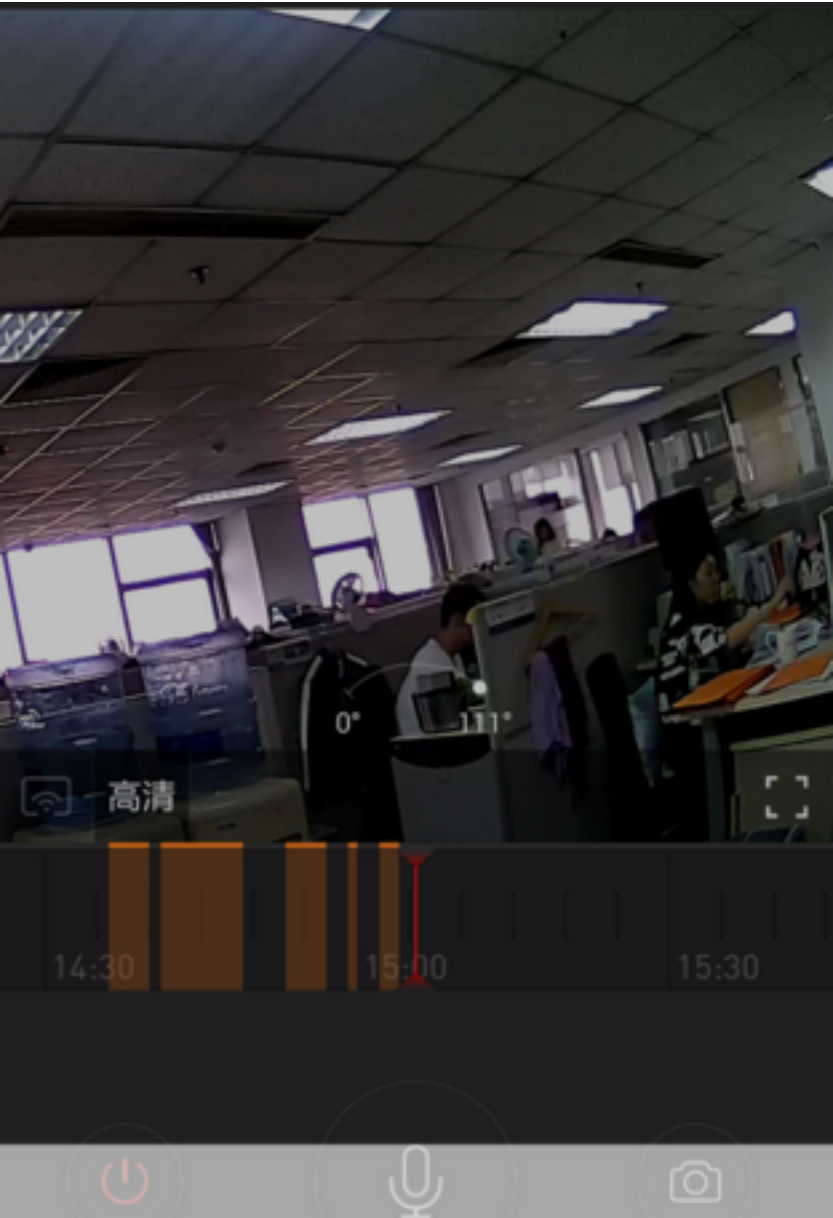
4.多说两句

最近智能家居比较火，其中有一类网络摄像头。也是我们公司准备做的一款产品。我简单做了一下前期的预研。目前的一些传统的行业所做的网络摄像头，大部分是基于局域网的网络摄像头，就是只能在自家的路由器上通过手机查看。这类产品，我觉得很难进入普通家庭，因为普通家庭也就那么不到100平方的房子，这种网络摄像头的就体现不是很好了。与普通的监控就是解决了布线的问题了。其他到没有什么提升。

还有一类是互联网行业做的网络摄像头。小米、360、百度等都有做这类型的网络摄像头。这类型的公司靠自己强大的云存储能力来实现。对这几个产品做了简单的了解，它们是支持本地存储，同时复制一份到云盘上。然后移动端(手机)是通过访问云盘里面的视频来实现监控的。这样虽然有一小段时间的延时，但是这样的效果还是不错的。你想，在监控某个地方，摄像头区域一般画面是不会发生太大的变化的，一个小时里面也就那么几个画面是要看到的。



假使一段15分钟的视频，经过分析，得到下面这样。



然后拖动到高亮的滑动条，高亮的地方，表示视频画面变动较大。这样就比较有针对性，也方便了客户了。还有重要的一点放在网盘，随时随地可以查看。但是也有一点就是隐私问题比较麻烦。其他的还有很多就不展开说明了。

作为一个小厂，同时作为一名小兵，暂时还不知道公司要做哪种类型的，上级只是让我了解点对点穿透。我猜应该是在家里有个摄像头监控，数据保存在本地。网络部分是移动端发起连接到摄像头，实行点对点的实时监控和读取摄像头本地存储的视频回放，全程只是经过服务器进行命令控制。视频走P2P(走不通应该是转发，这个还不知道。会不会提示当前网络不支持这种提示啊？期待！！毕竟如果转发视频的话很麻烦，很占资源)，视频保存本地。我猜目前公司应该是做成这个样子的。(公司非互联网公司，没有那么好的\*aaS平台)

参考资料：

本文地址：<http://www.cnblogs.com/wunaozai/p/5545150.html>

作者：[无脑仔的小明](#)  
出处：<http://www.cnblogs.com/wunaozai/>  
本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。  
如果文中有什么错误，欢迎指出。以免更多的人被误导。

分类：[Netty网络编程](#)

标签：[Netty](#), [java](#), [UDP穿透](#), [NAT](#)

好文要顶

关注我

收藏该文

[无脑仔的小明](#)  
[关注 - 17](#)  
[粉丝 - 229](#)  
[+加关注](#)

5

推荐


0

反对

« 上一篇：[搭建WebRtc环境](#)  
» 下一篇：[Nginx 单机百万QPS环境搭建](#)



评论	
#1楼 2016-06-01 09:59   早起的菜鸟 	
“ 楼主厉害！！	支持(0) 反对(0)
#2楼 2016-06-01 11:49   我是思思哦 	
“ 看来楼主对p2p（udp透传）已经有比较好的理解了，关于网络设备中一种很特殊的场景就是对称性NAT，每次会话请求路由器都分配新的端口，如果两边的客户端都是接入在这个路由器下面的时候，透传会失败。这个时候必须要有中转服务器进行媒体流的转发了。就是webrtc里面提及的ICE框架。	支持(1) 反对(0)
#3楼 2016-06-01 22:27   ShareDuck 	
“ 非常好的文章，将UDP穿透的原理讲得很清楚，感谢博主的辛勤写作。	支持(0) 反对(0)
#4楼 2016-06-15 11:18   {-) 大傻逼 	
“ 师兄威武~~	支持(0) 反对(0)
#5楼 2016-11-07 00:37   grokker001 	
“ Demo写的很不错！	支持(0) 反对(0)
#6楼 2017-12-26 13:47   学数学的程序猿 	
“ 请问博主，文中的配图使用啥工具画的？非常漂亮啊！	支持(0) 反对(0)
#7楼[楼主 	
“ @ 学数学的程序猿 <a href="https://processon.com/">https://processon.com/</a> 这个网站，在线画图	支持(0) 反对(0)
#8楼 2018-03-03 22:23   我系一个程序员 	
“ 感谢万分！	支持(0) 反对(0)
刷新评论 刷新页面 返回顶部	

 注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】超50万VC++源码：大型组态工控、电力仿真CAD与GIS源码库！
- 【活动】2050 大会 - 博客园程序员团聚 （5.25 杭州·云栖小镇）
- 【推荐】0元免费体验华为云服务
- 【活动】腾讯云云服务器新购特惠，5折上云





## 云数据库MySQL仅12元/月

满足入门学习、小规模应用、测试场景

立即购买

### 最新IT新闻：

- 创维激进的代价
  - 小米上市估值或超百度，雷军如何穿越互联网黑暗森林？
  - VS Code 1.23.0发布，带来众多更新
  - Google发布开源容器运行时gVisor
  - 李小加：在按正常程序尽快处理小米上市申请
- » 更多新闻...



300+篇运维、数据库等  
实战资料免费下载

点击获取



300+篇运维、数据库等  
实战资料免费下载

点击获取

### 最新知识库文章：

- 如何成为优秀的程序员？
  - 菜鸟工程师的超神之路 -- 从校园到职场
  - 如何识别人的技术能力和水平？
  - 写给自学者的入门指南
  - 和程序员谈恋爱
- » 更多知识库文章...

Copyright ©2018 无脑仔的小明

