

搭建AppRTC

搭建环境ubuntu 16.04server版本

腾讯课堂 零声学院 《WebRTC入门与提高》

<https://ke.qq.com/course/435382?quicklink=1>

咨询QQ群：782508536

WebRTC
入门几大坑

WebRTC入门与提高



看了那么多文章demo还是跑不起来

好不容易demo运行了画面确实黑的

为什么一到公网又黑屏？

HTTP的坑在哪里

终于看到画面了，但不懂原理啊

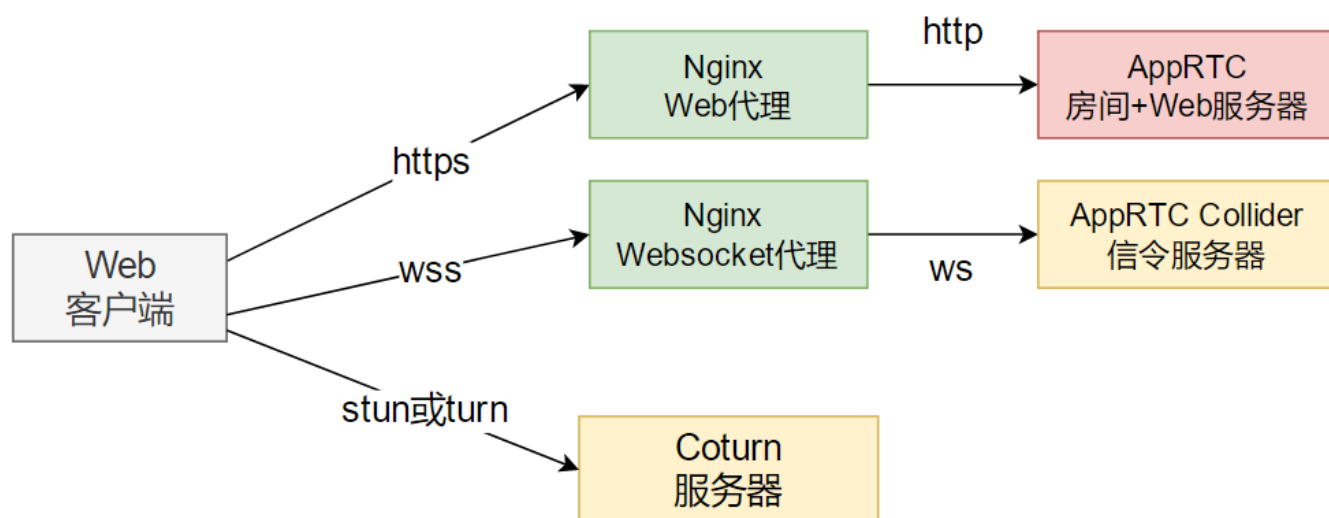
刚入门真要各种原理都深究吗？

为什么和Android通话不成功

1. 服务器组成

1. AppRTC 房间+Web服务器 <https://github.com/webrtc/apprtc>
2. Collider 信令服务器，在AppRTC源码里
3. CoTurn coturn打洞+中继服务器
4. Nginx 服务器，用于Web访问代理和Websocket代理。

AppRTC组成图如下所示。



AppRTC 房间+Web服务器使用python+js语言

AppRTC Collider信令服务器采用go语言

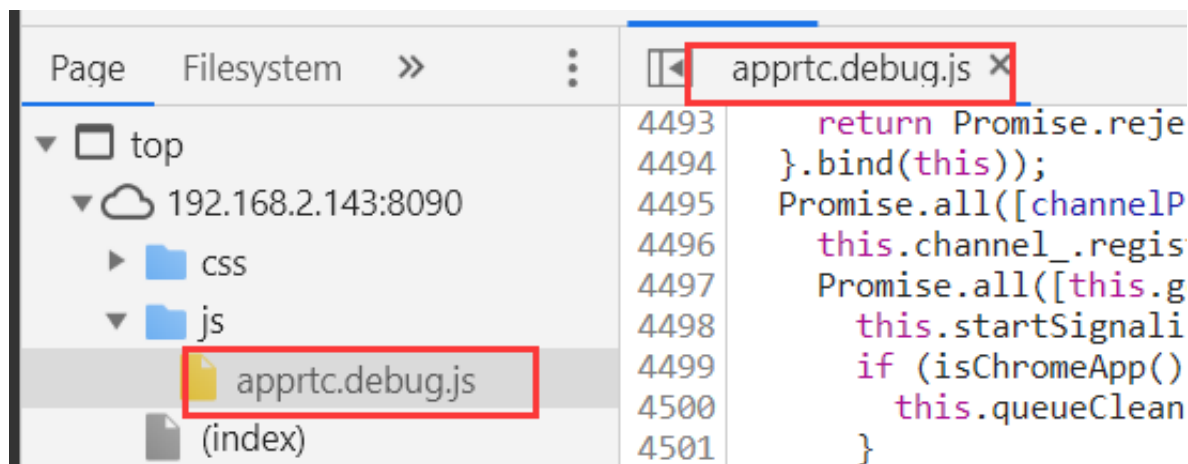
Coturn 采用C语言

在部署到公网时需要通过Nginx做Web和Websocket的代理连接

实际开发：把信令+房间管理 都是写到一个服务器

AppRTC的的价值：

- (1) js代码; `apprtc/out/chrome_app/js/apprtc.debug.js`



(2) Collider信令服务器原型。

2 准备工作

在一台全新的ubuntu 16.04 server版本安装AppRTC，前期准备工作。

1. 安装vim
2. 安装ssh
3. 安装ifconfig工具
4. 更新源
5. 安装git

2.1 安装vim

```
1 sudo apt-get install vim
```

2.2 安装ssh

```
1 sudo apt-get install openssh-server
```

输入 "sudo ps -e | grep ssh" --> 回车 --> 有 sshd, 说明 ssh 服务已经启动, 如果没有启动, 输入 "sudo service ssh start" --> 回车 --> ssh 服务就会启动。

2.3 安装ifconfig工具

```
1 sudo apt-get install net-tools
2 sudo apt-get install iputils-ping
```

2.4 更新源

将源更新为阿里源, 否则apt-get install安装软件较慢。

```
1 # 1 在修改source.list前, 最好先备份一份
2 sudo mv /etc/apt/sources.list /etc/apt/sources.list.old
3 # 2 执行命令打开source.list文件
4 sudo vim /etc/apt/sources.list
5 # 3 复制更新源
```

复制以下源到sources.list

```
1 # deb cdrom:[Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64
  (20160420.1)]/ xenial main restricted
2 deb-src http://archive.ubuntu.com/ubuntu xenial main restricted #Added
  by software-properties
3 deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted
4 deb-src http://mirrors.aliyun.com/ubuntu/ xenial main restricted
  multiverse universe #Added by software-properties
5 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
6 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main
  restricted multiverse universe #Added by software-properties
7 deb http://mirrors.aliyun.com/ubuntu/ xenial universe
8 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
```

```
9 deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse
10 deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse
11 deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted
    universe multiverse
12 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-backports main
    restricted universe multiverse #Added by software-properties
13 deb http://archive.canonical.com/ubuntu xenial partner
14 deb-src http://archive.canonical.com/ubuntu xenial partner
15 deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
16 deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main
    restricted multiverse universe #Added by software-properties
17 deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
18 deb http://mirrors.aliyun.com/ubuntu/ xenial-security multiverse
```

保存后更新源

```
1 # 4 update命令
2 sudo apt-get update
```

2.5 安装git

```
1 sudo apt-get install git
```

3 安装AppRTC必须的软件

3.0 创建目录

```
1 mkdir ~/webrtc
2 cd ~/webrtc
```

注意后续的目录，因为我的webrtc全路径为/home/lqf/webrtc，所以后续的目录都是采

用该目录。

安装需要的各种工具(除了apt之外还可以下载安装包或者源码自己编译安装):

3.1 安装JDK

```
1 # 先安装add-apt-repository命令支持
2 sudo apt-get install python-software-properties
3 sudo apt-get install software-properties-common
4 # 第一步: 打开终端, 添加ppa源
5 sudo add-apt-repository ppa:openjdk-r/ppa
6 # 第二步: 更新源
7 sudo apt-get update
8 # 第三步: 安装openjdk 8
9 sudo apt-get install openjdk-8-jdk
10 # 第四步: 配置openjdk 8为默认java环境
11 sudo update-alternatives --config java
12 sudo update-alternatives --config javac
13 # 最后, 验证一下是否成功
14 java -version
15 # 返回
16 openjdk version "1.8.0_222"
17 OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-
18 1ubuntu1~16.04.1-b10)
19 OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

3.2 安装node.js

```
1 cd ~/webrtc
2 wget https://nodejs.org/dist/v10.16.0/node-v10.16.0-linux-x64.tar.xz
3 # 解压
4 tar -xvf node-v10.16.0-linux-x64.tar.xz
5 # 进入目录
6 cd node-v10.16.0-linux-x64/
7 # 查看当前的目录
```

```

8  pwd
9
10 # 确认一下nodejs下bin目录是否有node 和npm文件，如果有就可以执行软连接，比如
11 sudo ln -s /home/lqf/webRTC/node-v10.16.0-linux-x64/bin/npm
   /usr/local/bin/
12 sudo ln -s /home/lqf/webRTC/node-v10.16.0-linux-x64/bin/node
   /usr/local/bin/
13
14 # 看清楚，这个路径是你自己创建的路径，我的路径是/home/lqf/webRTC/node-
   v10.16.0-linux-x64
15
16 # 查看是否安装，安装正常则打印版本号
17 node -v
18 npm -v
19
20
21 # 有版本信息后安装 grunt-cli,先进到nodejs的bin目录，要和自己的目录匹配
22 cd /home/lqf/webRTC/node-v10.16.0-linux-x64/bin
23 sudo npm -g install grunt-cli
24 sudo ln -s /home/lqf/webRTC/node-v10.16.0-linux-x64/bin/grunt
   /usr/local/bin/
25
26 grunt --version
27 # 显示grunt-cli v1.3.2
28
29 #使用淘宝源替换npm，后续要执行npm时执行cnpm
30 sudo npm install -g cnpm --registry=https://registry.npm.taobao.org
31 sudo ln -s /home/lqf/webRTC/node-v10.16.0-linux-
   x64/lib/node_modules/cnpm/bin/cnpm /usr/local/bin/

```

3.3 安装Python和Python-webtest (python2.7)

ubuntu 一般都会自带python 2.7 输入 python -v 输出版本则已经有了

```

lqf@ubuntu:~$ python -V
Python 2.7.12

```

如果没有则安装

```

1 # 没有则输入下命令

```

```
2 sudo apt-get install python
3 # python 安装之后安装 Python-webtest
4 sudo apt-get install python-webtest
5
6
7 python -V
8 #Python 2.x
```

3.4 安装google_appengine

```
1 # 回到webrtc目录
2 cd ~/webrtc/
3 # 下载google_appengine
4 wget https://storage.googleapis.com/appengine-
  sdks/featured/google_appengine_1.9.40.zip
5 unzip google_appengine_1.9.40.zip
6
7 #配置环境变量: 在/etc/profile文件最后增加一行, 和自己路径保持一致
8 export PATH=$PATH:/home/lqf/webrtc/google_appengine
9 # 生效
10 source /etc/profile
```

3.5 安装go

安装go

```
1 sudo apt-get install golang-go
2 #检测go 版本
3 go version
4 #go version go1.6.2 linux/amd64
```

创建go工作目录

```
1 #创建go工作目录
2 mkdir -p ~/webrtc/goworkspace/src
```



```
3 #配置环境变量：在/etc/profile文件最后增加一行：
4 sudo vim /etc/profile
```

添加

```
1 export GOPATH=/home/lqf/webrtc/goworkspace
2 # 然后执行
3 source /etc/profile
```

3.6 安装apprtc

```
1 cd ~/webrtc/
2 # 先尝试使用github的下载连接
3 git clone https://github.com/webrtc/apprtc.git
4 # 如果github的连接下不了就用码云的链接
5 git clone https://gitee.com/sabergithub/apprtc.git
6 #将collider的源码软连接到go的工作目录下
7 ln -s /home/lqf/webrtc/apprtc/src/collider/collider $GOPATH/src
8 ln -s /home/lqf/webrtc/apprtc/src/collider/collidermain $GOPATH/src
9 ln -s /home/lqf/webrtc/apprtc/src/collider/collidertest $GOPATH/src
10
11 #下一步在编译 go get collidermain: 被墙
12 #报错: package golang.org/x/net/websocket: unrecognized import path
    "golang.org/x/net/websocket"
13 #所以先执行:
14 mkdir -p $GOPATH/src/golang.org/x/
15 cd $GOPATH/src/golang.org/x/
16 git clone https://github.com/golang/net.git net
17 go install net
18
19 #编译collidermain
20 cd ~/webrtc/goworkspace/src
21 go get collidermain
22 go install collidermain
23
```

3.7 安装coturn

```
1 sudo apt-get install libssl-dev
2 sudo apt-get install libevent-dev
3
4 # 返回webrtc目录
5 cd ~/webrtc
6 #git clone https://github.com/coturn/coturn
7 #cd coturn
8 # 提供另一种安装方式turnserver是coturn的升级版本
9 wget http://coturn.net/turnserver/v4.5.0.7/turnserver-4.5.0.7.tar.gz
10 tar xzf turnserver-4.5.0.7.tar.gz
11 cd turnserver-4.5.0.7
12
13 ./configure
14 make
15 sudo make install
```

3.8 安装Nginx

注意安装的时候要带ssl

```
1 sudo apt-get update
2 #安装依赖: gcc、g++依赖库
3 sudo apt-get install build-essential libtool
4 #安装 pcre依赖库 (http://www.pcre.org/)
5 sudo apt-get install libpcre3 libpcre3-dev
6 #安装 zlib依赖库 (http://www.zlib.net)
7 sudo apt-get install zlib1g-dev
8 #安装ssl依赖库
9 sudo apt-get install openssl
10
11
12 #下载nginx 1.15.8版本
13 wget http://nginx.org/download/nginx-1.15.8.tar.gz
14 tar xvzf nginx-1.15.8.tar.gz
```

```

15 cd nginx-1.15.8/
16
17
18 # 配置，一定要支持https
19 ./configure --with-http_ssl_module
20
21 # 编译
22 make
23
24 #安装
25 sudo make install

```

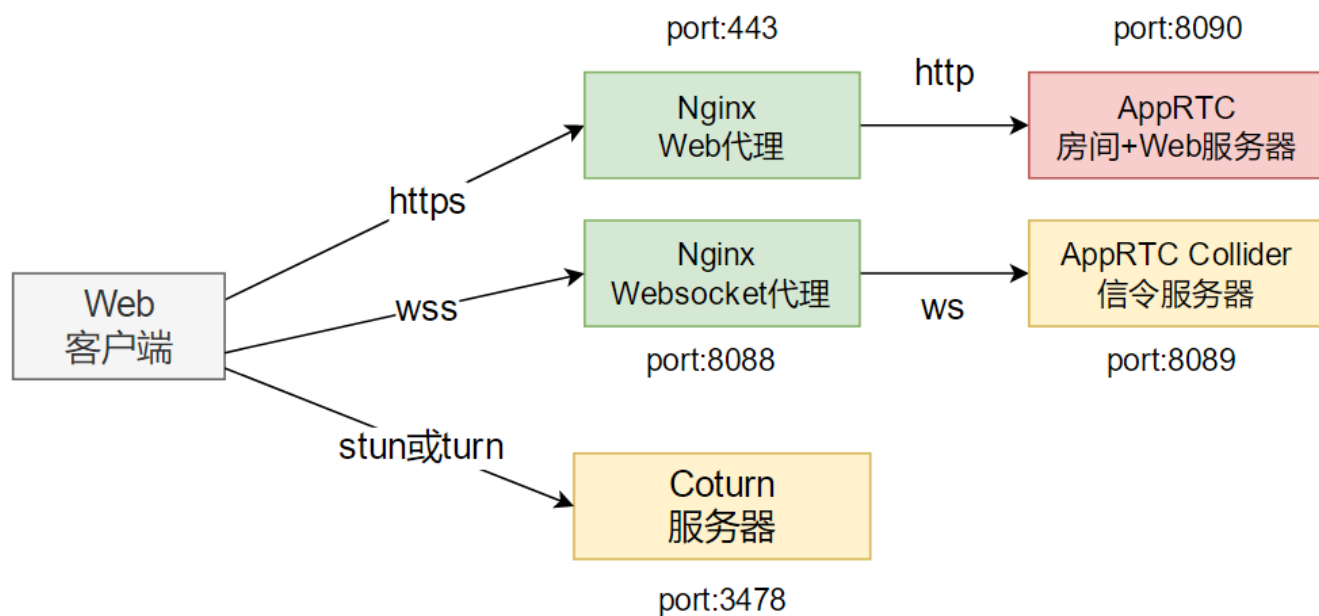
默认安装目录： **/usr/local/nginx**

启动： `sudo /usr/local/nginx/sbin/nginx`

停止： `sudo /usr/local/nginx/sbin/nginx -s stop`

重新加载配置文件： `sudo /usr/local/nginx/sbin/nginx -s reload`

4 配置与运行



4.1 coturn 打洞+中继服务器

配置防火墙，允许访问3478端口（含tcp和udp，此端口用于nat穿透）

可以前台执行：sudo turnserver -L 0.0.0.0 -a -u lqf:123456 -v -f -r nort.gov 然后退出，再到后台去执行

```
1 sudo nohup turnserver -L 0.0.0.0 -a -u lqf:123456 -v -f -r nort.gov &
2 #账号 lqf 密码:123456 这一步随便给，但是后面配置apprtc时需要用到
3 #命令后加 & ,执行起来后按 ctr+c,不会停止
```

```
1 #开启新窗口 执行
2 lsof -i:3478
3 #输出大致这样的成功
4 COMMAND      PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
5 turnserve 30299 root  16u  IPv4  61868      0t0  UDP *:3478
6 turnserve 30299 root  17u  IPv4  61869      0t0  UDP *:3478
7 turnserve 30299 root  32u  IPv4  61879      0t0  TCP *:3478 (LISTEN)
8 turnserve 30299 root  33u  IPv4  61880      0t0  TCP *:3478 (LISTEN)
```

4.2 collider 信令服务器

配置防火墙，允许访问8089端口（tcp，用于客户端和collider建立websocket信令通信）

```
1 # 执行collider 信令服务器
2 sudo nohup $GOPATH/bin/collidermain -port=8089 -tls=false -room-
  server="http://192.168.2.143:8090" &
```

-room-server="<http://192.168.2.143:8090>" 实际是连接房间服务器的地址

```
1 #同样检查是否成功
```

```
2 sudo lsof -i:8089
3 #显示
4 COMMAND      PID USER  FD   TYPE DEVICE SIZE/OFF NODE NAME
5 colliderm 30354 root   3u   IPv6  62696      0t0  TCP *:8089 (LISTEN)
```

4.3 apprtc 房间服务器

先安装python的request模块

4.3.1 安装pip

下载setup-python工具

```
1 cd /home/lqf/webrtc
2 wget https://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11-py2.7.egg --no-check-certificate
3 chmod +x setuptools-0.6c11-py2.7.egg
4 sudo ./setuptools-0.6c11-py2.7.egg
5 wget https://pypi.python.org/packages/source/p/pip/pip-1.5.4.tar.gz
6 tar -xf pip-1.5.4.tar.gz
7 cd pip-1.5.4/
8 sudo python setup.py install
9 sudo pip install requests
```

4.3.2 修改配置文件

配置防火墙，允许访问8090端口（tcp，此端口用于web访问）

配置文件修改（主要是配置apprtc对应的conturn和collider相关参数）

```
1 vim /home/lqf/webrtc/apprtc/src/app_engine/constants.py
```

修改后（填的都是外网IP，为了适合更多数朋友测试，我这里用的是内网的环境，在公网部

署填入公网IP即可)

ICE_SERVER_OVERRIDE = None 注释掉

```
1  # Turn/Stun server override. This allows AppRTC to connect to turn
   servers
2  # directly rather than retrieving them from an ICE server provider.
3  # ICE_SERVER_OVERRIDE = None
4  # Enable by uncomment below and comment out above, then specify turn
   and stun
5  ICE_SERVER_OVERRIDE = [
6      {
7          "urls": [
8              "turn:192.168.2.143:3478?transport=udp",
9              "turn:192.168.2.143:3478?transport=tcp"
10         ],
11         "username": "lqf",
12         "credential": "123456"
13     },
14     {
15         "urls": [
16             "stun:192.168.2.143:3478"
17         ]
18     }
19 ]
20
21 ICE_SERVER_BASE_URL = 'https:192.168.2.143'
22 ICE_SERVER_URL_TEMPLATE = '%s/v1alpha/iceconfig?key=%s'
23 ICE_SERVER_API_KEY = os.environ.get('ICE_SERVER_API_KEY')
24
25 # Dictionary keys in the collider instance info constant.
26 WSS_INSTANCE_HOST_KEY = '192.168.2.143:8088'
27 WSS_INSTANCE_NAME_KEY = 'vm_name'
28 WSS_INSTANCE_ZONE_KEY = 'zone'
29 WSS_INSTANCES = [{
30     WSS_INSTANCE_HOST_KEY: '192.168.2.143:8088',
31     WSS_INSTANCE_NAME_KEY: 'wsserver-std',
32     WSS_INSTANCE_ZONE_KEY: 'us-central1-a'
33 }]
34
```

进到apprtc目录

```
1 #编译
2 cd /home/lqf/webrtc/apprtc
3 sudo cnpm install
4 sudo grunt build
```

启动:

```
sudo /home/lqf/webrtc/google_appengine/dev_appserver.py --host=0.0.0.0 --port=8090
/home/lqf/webrtc/apprtc/out/app_engine --skip_sdk_update_check
```

```
1 # 默认端口是8080， 可以自己指定端口，我们这里指定为8090
2 sudo nohup /home/lqf/webrtc/google_appengine/dev_appserver.py --
  host=0.0.0.0 --port=8090 /home/lqf/webrtc/apprtc/out/app_engine --
  skip_sdk_update_check &
3 #如果提示更新选择: n
```

此时可以通过谷歌浏览器访问测试:

<http://192.168.2.143:8090/>

```
1 #检查
2 sudo lsof -i:8090
3 #输出下列内容
4
```

4.4 nginx代理

4.4.1 生成证书

```
1 mkdir -p ~/cert
```

```
2 cd ~/cert
3 # CA私钥
4 openssl genrsa -out key.pem 2048
5 # 自签名证书
6 openssl req -new -x509 -key key.pem -out cert.pem -days 1095
```

4.4.2 Web代理

反向代理apprt看，使之支持https访问，如果http直接访问apprt看，则客户端无法启动视频音频采集（必须得用https访问）

配置web服务器

(1) 配置自己的证书

ssl_certificate /home/lqf/cert/cert.pem; // 注意证书所在的路径

ssl_certificate_key /home/lqf/cert/key.pem;

(2) 配置主机域名或者主机IP

server_name 192.168.2.143;

完整配置文件：/usr/local/nginx/conf/conf.d/apprt看-https-proxy.conf

```
1 upstream roomserver {
2     server 192.168.2.143:8090;
3 }
4 server {
5     listen 443 ssl;
6     ssl_certificate /home/lqf/cert/cert.pem;
7     ssl_certificate_key /home/lqf/cert/key.pem;
8     charset utf-8;
9     # ip地址或者域名
10    server_name 192.168.221.134;
11    location / {
12        # 转向代理的地址
13        proxy_pass http://roomserver$request_uri;
14        proxy_set_header Host $host;
15    }
16 }
```


编辑nginx.conf文件，在末尾}之前添加包含文件

```
1 include /usr/local/nginx/conf/conf.d/*.conf;
2 }
```

4.4.3 配置websocket代理

ws 不安全的连接 类似http

wss是安全的连接，类似https

完整配置文件：/usr/local/nginx/conf/conf.d/apprtc-websocket-proxy.conf

```
1 map $http_upgrade $connection_upgrade {
2     default upgrade;
3     '' close;
4 }
5 upstream websocket {
6     server 192.168.2.143:8089;
7 }
8
9 server {
10     listen 8088 ssl;
11     ssl_certificate /home/lqf/cert/cert.pem;
12     ssl_certificate_key /home/lqf/cert/key.pem;
13
14     server_name 192.168.2.143;
15     location /ws {
16         proxy_pass http://websocket;
17         proxy_http_version 1.1;
18         proxy_connect_timeout 4s; #配置点1
19         proxy_read_timeout 6000s; #配置点2，如果没效，可以考虑这个时间配
置长一点
20         proxy_send_timeout 6000s; #配置点3
21         proxy_set_header Upgrade $http_upgrade;
22         proxy_set_header Connection $connection_upgrade;
23     }
24 }
```

4.5 解决跨域问题

浏览器通话跨域问题 :pushState

Messages:Failed to start signaling: Failed to execute 'pushState' on 'History'

解决方法

修改文件apprtc/src/web_app/js/appcontroller.js

```
1 vim /home/lqf/webrtc/apprtc/src/web_app/js/appcontroller.js
2
3 #搜索 displaySharingInfo_ 大概是445行displaySharingInfo_函数第一行添加
4
5 roomLink=roomLink.replace("http","https");
```

最终结果 (大概446行的修改)

```
1 AppController.prototype.displaySharingInfo_ = function(roomId,
2   roomLink) {
3   roomLink=roomLink.replace("http","https");
4   this.roomLinkHref_.href = roomLink;
5   this.roomLinkHref_.text = roomLink;
6   this.roomLink_ = roomLink;
7   this.pushCallNavigation_(roomId, roomLink);
8   this.activate_(this.sharingDiv_);
9   };
10
```

然后重新build

```
1 cd ~/webrtc/apprtc
```

```
2 | sudo grunt build
```

在重新在前台运行

```
1 | sudo /home/lqf/webrtc/google_appengine/dev_appserver.py --  
host=0.0.0.0 --port=8090 /home/lqf/webrtc/apprtc/out/app_engine --  
skip_sdk_update_check
```

总结

- (1) 目录/home/lqf/webrtc
- (2) IP: 192.168.2.143
- (3) 前后台执行的问题
- (4) 防火墙开发端口的问题
- (5) 端口规划的问题

