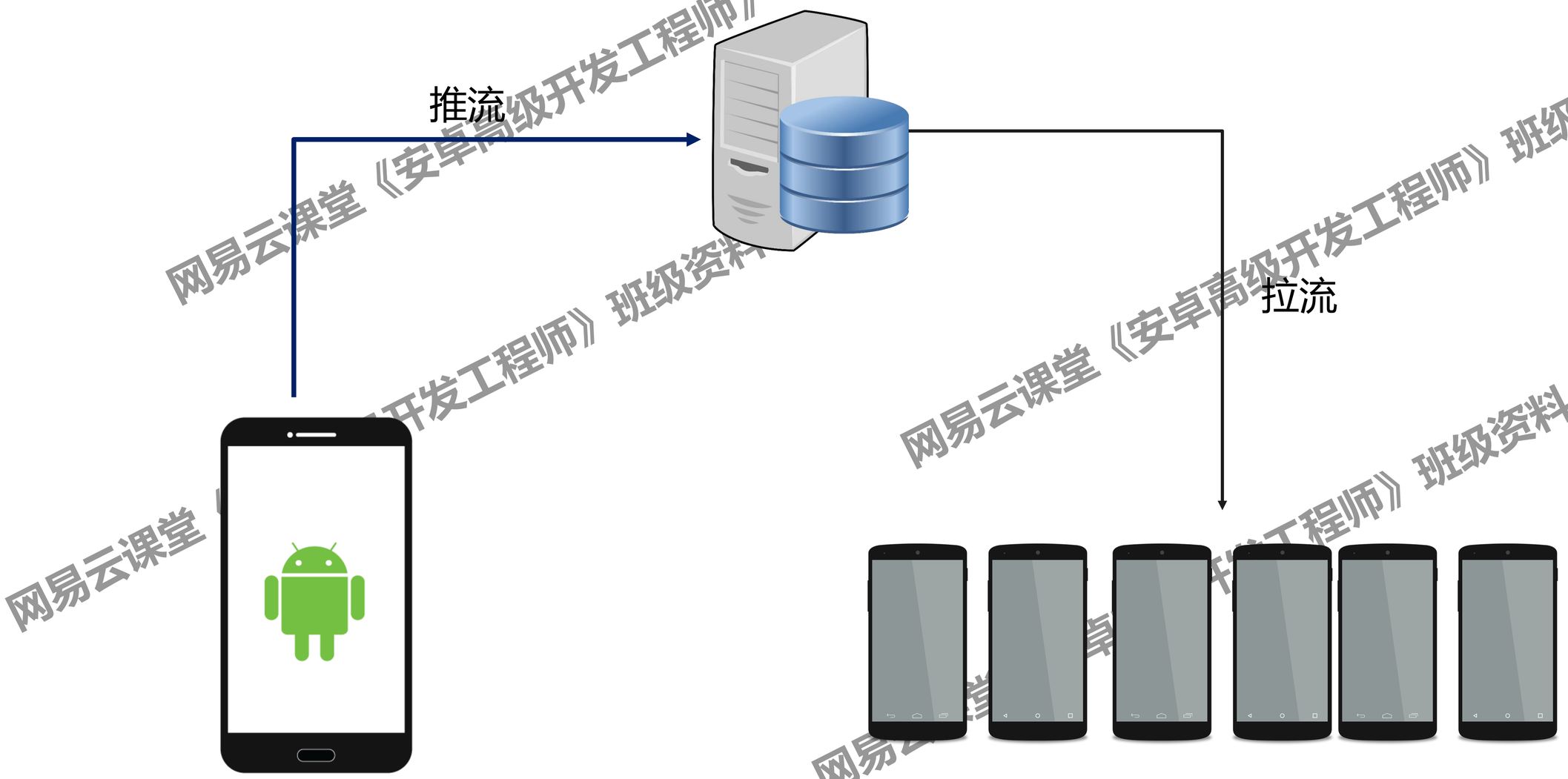
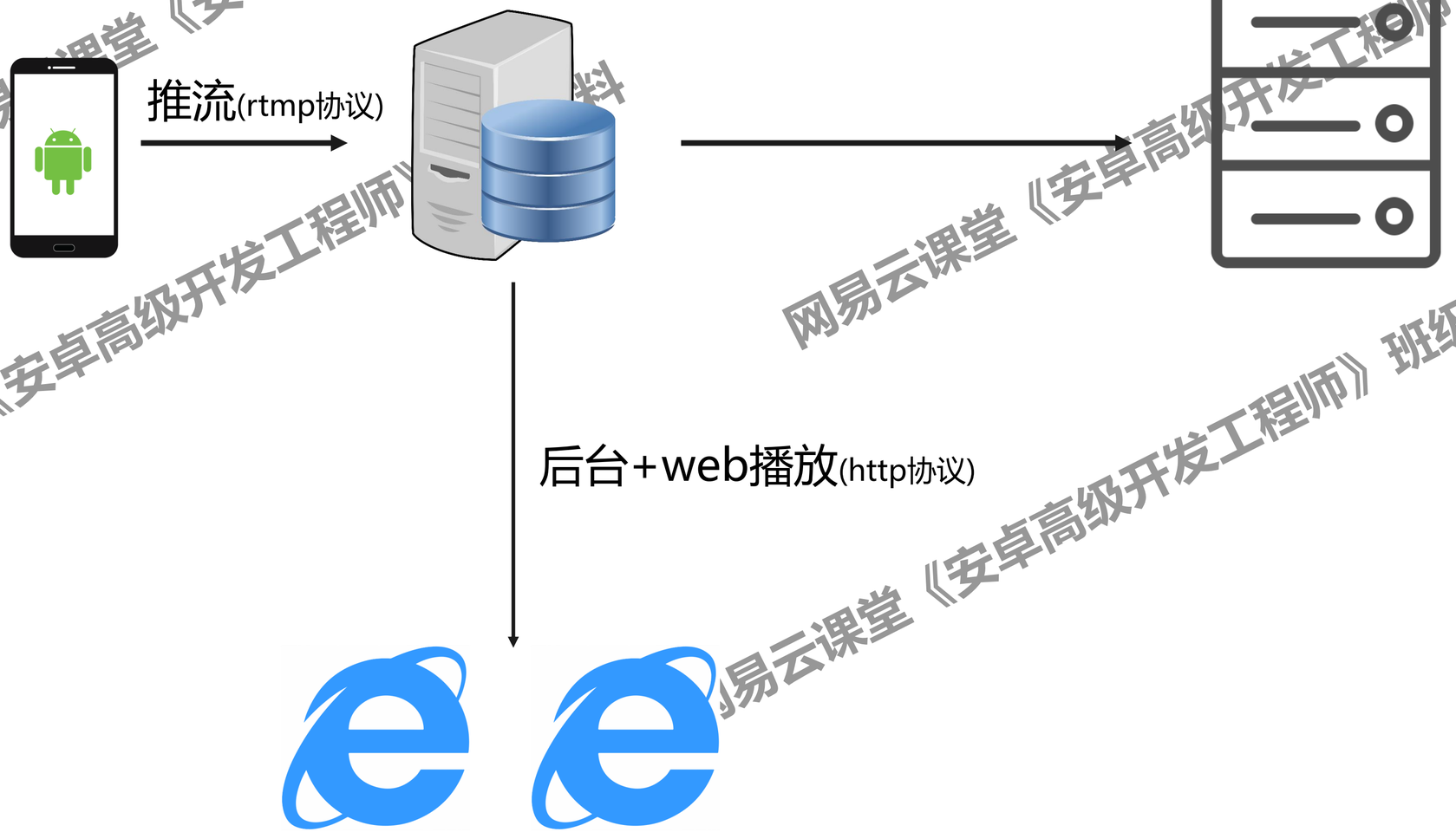
云信实战项目

（直播视频推流）

推送服务器流程



服务器搭建



Nginx是什么?

Nginx是一个高性能的HTTP和反向代理web服务器，用来处理前端(Android IOS Web)过来的请求、以往在一台服务器上 需要部署多个服务 需要通过端口号指明访问具体服务。部署Nginx后不需要了。Nginx可以理解为 十字路口的 警察，主要用来导流与分发

常用链接

- Nginx下载地址 http://nginx.org/download/nginx-1.15.3.tar.gz
- 直播房间服务模块地址 https://codeload.github.com/arut/nginx-rtmp-module/tar.gz/v1.2.1

服务器搭建

```
下载nginx
        wget http://nginx.org/download/nginx-1.15.3.tar.gz
解压
        tar xvf nginx-1.15.3.tar.gz
下载nginx rtmp模块
        wget https://codeload.github.com/arut/nginx-rtmp-module/tar.gz/v1.2.1

解压
        tar xvf v1.2.1

进入nginx目录
        cd nginx-1.15.3

执行:
shell
#--add-module 指向rtmp模块目录
        ./configure --prefix=./bin --add-module=../nginx-rtmp-module-1.2.1

安装
        make  && make install
```

注意:

在这个过程中可能因为环境不同而出现不同错误，比如缺少pcre、openssl等，这时候就需要安装这些库。参考https://blog.csdn.net/z920954494/article/details/52132125

修改nginx.conf

cd bin/conf　　vim nginx.conf 修改为：

```
user root;
worker_processes  1;

error_log  logs/error.log debug;

events {
    worker_connections  1024;
}

rtmp {
    server {
        listen 1935;
        application myapp {
            live on;
            drop_idle_publisher 5s;
        }
    }
}
http {
    server {
        listen       8081;
        location /stat {
            rtmp_stat all;
            rtmp_stat_stylesheet stat.xsl;
        }
        location /stat.xsl {
            root /root/nginx/nginx-rtmp-module-1.2.1/;
        }
        location /control {
            rtmp_control all;
        }
        location /rtmp-publisher {
            root /root/nginx/nginx-rtmp-module-1.2.1/test;
        }

        location / {
            root /root/nginx/nginx-rtmp-module-1.2.1/test/www;
        }
    }
}
```

注意：

让rtmp协议的请求走向房间服务器

推流思路图

数据源

摄像头(Camera)

采集(onPreviewFrame)

VideoChannel

麦克风(AudioRecord)

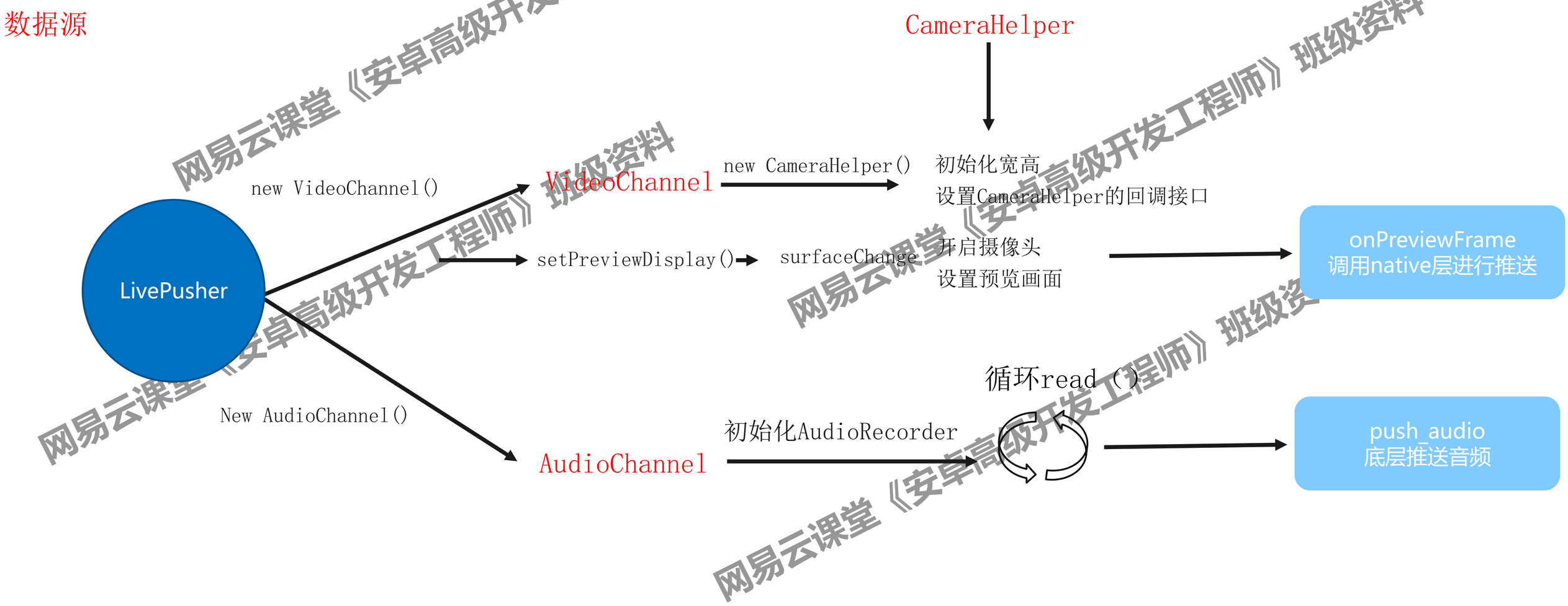
读取(read)

AudioChannel

LivePusher

Native推流

推流思路图

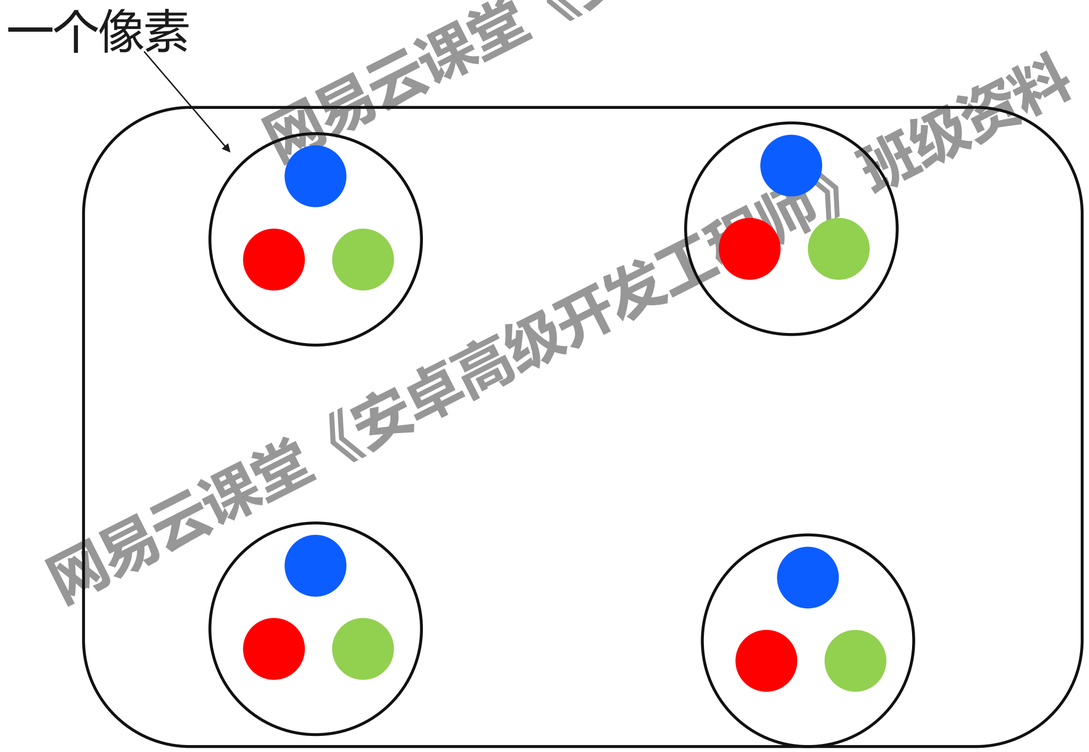


什么视频编码采用YUV420而不是rgb

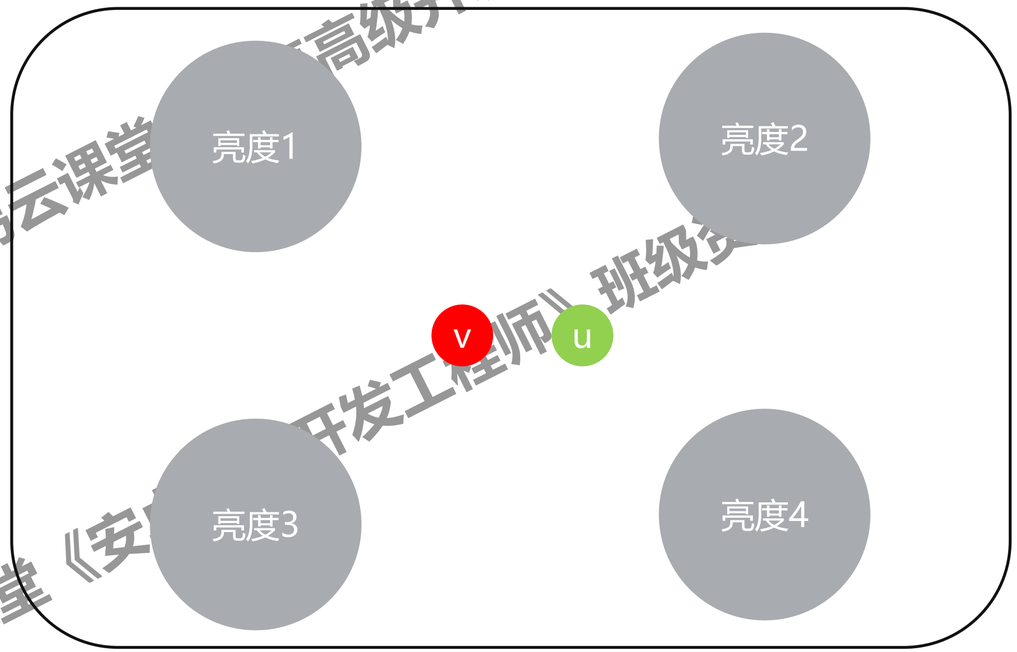
- Rgb原理： 定义RGB 是从颜色发光的原理来设计定的，由红、绿、蓝三盏灯，当它们的光相互叠合的时候，色彩相混，而亮度却等于两者亮度之总和（两盏灯的亮度嘛！），越混合亮度越高，即加法混合。RGB24 是指 R， G， B 三个分量各占 8 位
- Yuv原理： YUV 主要用于优化彩色视频信号的传输，与 RGB 视频信号传输相比，它最大的优点在于只需占用极少的频宽（ RGB 要求三个独立的视频信号同时传输）其中 “Y” 表示明亮度也就是灰阶值；而 “U” 和 “V” 表示的则是色度

人眼漫天过海

- RGB表示2*2个像素的区域



- 用yuv来表示



- 数据量 4*3字节
- 数据量 （4+1+1）字节
- 人眼对亮度是比较敏感的，对色度不敏感
- 内存计算公式 width*height*3/2 (2*2*3/2)

NV21编码

2、YUV420SP_NV21

1行：1(byte/pixel)*8pixel

4行

y	y	y	y	y	y	y	y
y	y	y	y	y	y	y	y
y	y	y	y	y	y	y	y
y	y	y	y	y	y	y	y

共有：1*8*4=32(byte)

1行：2byte/4pixel*16pixel

2行

v	u	v	u	v	u	v	u
v	u	v	u	v	u	v	u

共有：2/4*16*2=16(byte)

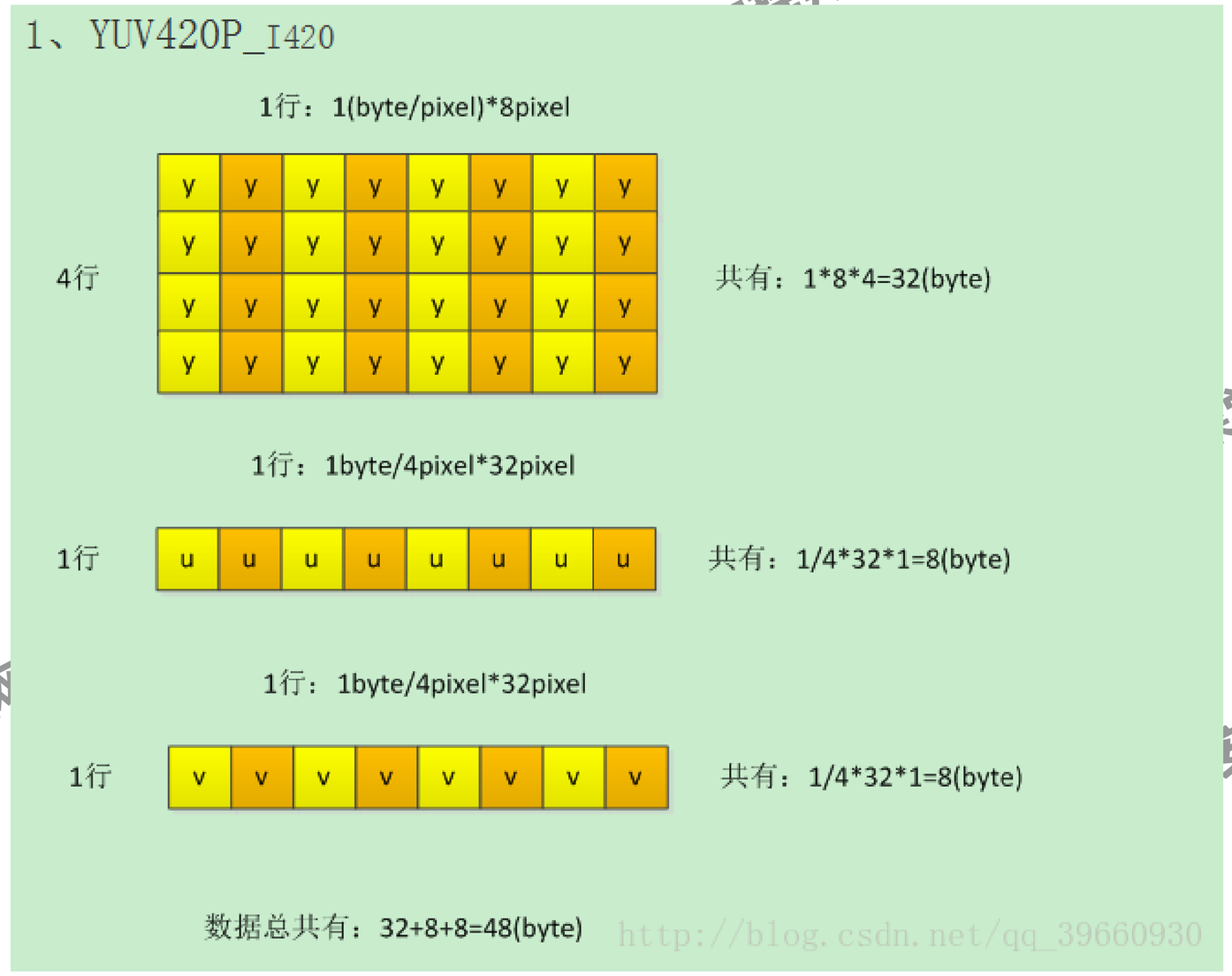
数据总共有：32+16=48(byte)

如1920*1080的一帧图片，经过yuv的方式排列

0 ~ 1920*1080 全部是Y

后面是 u 和v 的交替变换

YUV I420编码



如1920*1080的一帧图片，经过yuv的方式排列

0 ~ 1920*1080 全部是Y

1920*1080 ~ 1920*1080+ 1920*1080/4全部是u

1920*1080+ 1920*1080/4 ~ 一帧末尾 全部是v

Android摄像头拍摄效果

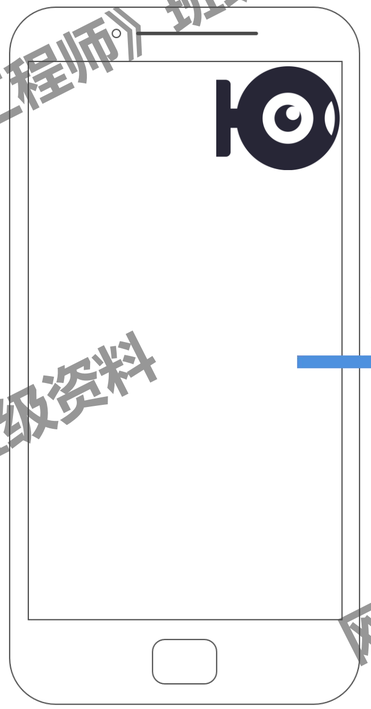
原始的画面

后置摄像头拍摄

输出的图像

Android摄像头拍摄效果

原始的画面



后置摄像头
正向角度

输出的图像

后置摄像头
正向角度

输出的图像

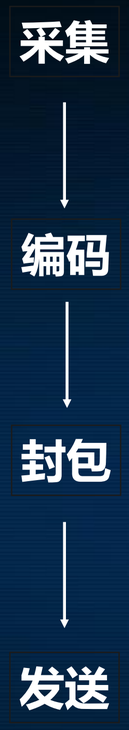
线程锁

线程锁

pthread_mutex_init ,
pthread_mutex_destory ,
pthread_mutex_lock ,
pthread_mutex_unlock
这几个函数以完成
锁的初始化，
锁的销毁，
上锁
释放锁操作。

```
#include <pthread.h>
#include <stdio.h>

pthread_mutex_t mutex ;
void *print_msg(void *arg){
        int i=0;
        pthread_mutex_lock(&mutex);
        for(i=0;i<15;i++){
                printf("output : %d\n",i);
                usleep(100);
        }
        pthread_mutex_unlock(&mutex);
}
int main(int argc,char** argv){
        pthread_t id1;
        pthread_t id2;
        pthread_mutex_init(&mutex,NULL);
        pthread_create(&id1,NULL,print_msg,NULL);
        pthread_create(&id2,NULL,print_msg,NULL);
        pthread_join(id1,NULL);
        pthread_join(id2,NULL);
        pthread_mutex_destroy(&mutex);
        return 1;
}
```



1、如何采集音频数据?

2、如何采集视频数据?

3、如何编码音视频数据?

4、如何组装RTMP数据?

5、如何发送音视频数据?

H264

视频编码格式:

代表软编码器------openh264、x264

16进制	类型	说明
0x67	SPS	序列参数集,profile、level、宽高与颜色空间等信息
0x68	PPS	图像参数集,通常情况下，PPS类似于SPS
0x65	I	关键帧,播放器需要从IDR开始播放,发送IDR前需要发送sps与pps
其他		P/B等需要参考帧进行解码的帧

Profile : baseline main high
Level : 限制了码率上限
Resolution : 分辨率
Bitrate : 码率,与数据大小成正比
Frame Rate : 帧率,每秒多少帧图像，影响流畅度
Frame Interval : 关键帧间隔

AAC

音频编码格式:

代表软编码器------faac、fdkaac等

AAC格式	说明
ADIF	Audio Data Interchange Format 音频数据交换格式。在明确定义的开始处进行解码。常用于保存文件
ADTS	Audio Data Transport Stream 音频数据传输流。可以在这个流中任何位置开始解码。常用于流数据的传输

Profile : lc main le

Sample Rate : 采样率

Channel : 声道数

Bitrate : 码率,与数据大小成正比

RTMP

Real Time Message Protocol(实时信息传输协议)

基于TCP的应用层协议

视频包
1、解码信息包(sps与pps)
2、数据包
音频包
1、解码信息包
2、数据包

视频包数据

关键帧	0x17	0x01	0x00	0x00	0x00	4字节数据长度	h264裸数据
非关键帧	0x27	0x01	0x00	0x00	0x00	4字节数据长度	h264裸数据
sps与pps	0x17	0x00	0x00	0x00	0x00	sps+pps数据	

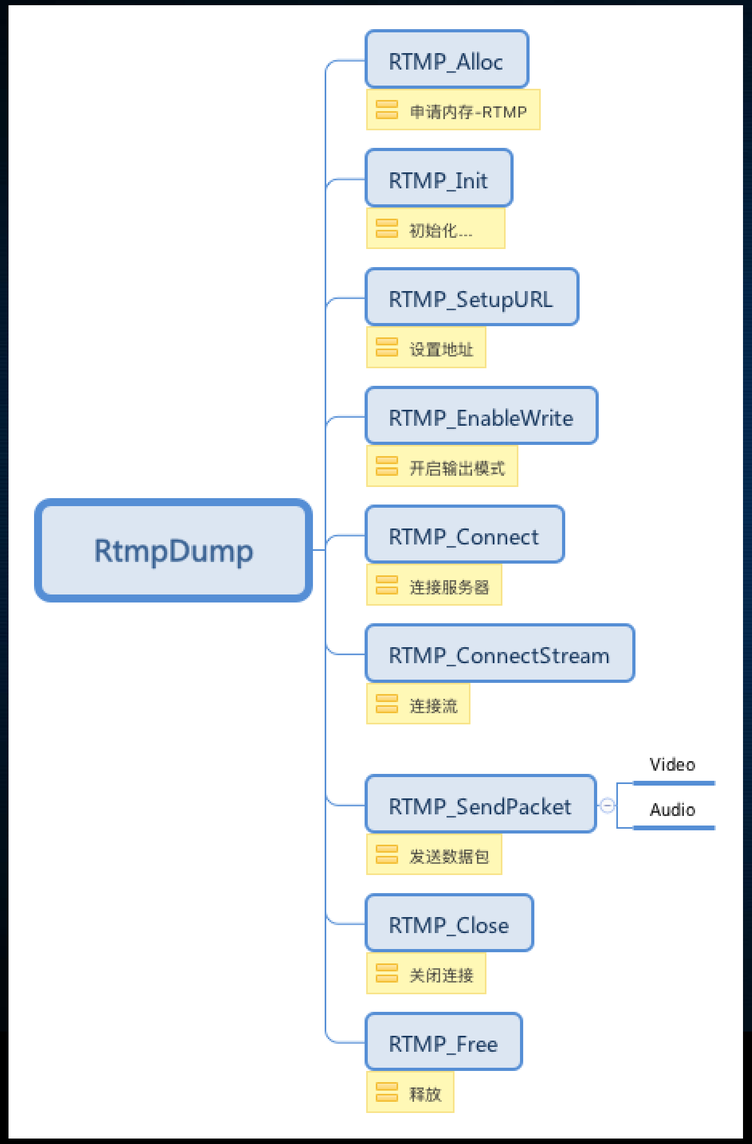
类型	长度	说明
configurationVersion	1	0x01 版本
avcProfileIndication	1	sps[1] Prifile
profile_compatibility	1	sps[2] 兼容性
profile_level	1	sps[3] Profile Level
lengthSizeMinusOne	1	0xff 包长数据所使用的字节数,通常为oxf
numOfSequenceParameterSets	1	0xe1 SPS个数,通常为0xe1
sequenceParameterSetLength	2	sps长度
sequenceParameterSetNALUnits	sequenceParameterSetLength	sps内容
numOfPictureParameterSets	1	0x01 pps个数
pictureParameterSetLength	2	pps长度
pictureParameterSetNALUnits	pictureParameterSetLength	pps内容

音频包数据

解码信息	0xAF	0x00	解码数据
数据	0xAF	0x01	音频数据

RTMPDump

http://rtmpdump.mplayerhq.hu/



根据Makefile编写CMakeLists.txt

CMakeLists.txt

```
1  #关闭ssl 不支持rtmps
2  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -DNO_CRYPTO"  )
3  #所有源文件放入 rtmp_source 变量
4  file(GLOB rtmp_source *.c)
5
6  #编译静态库
7  add_library(
8                rtmp
9                STATIC
10              ${rtmp_source} )
```

谢谢观看