


360技术嘉年华—测试之美

视频流技术在手机远控中的应用

- 
- 01 手机远控中开源技术的痛点**
 - 02 视频流技术的应用框架与优势**
 - 03 实现方案与效果展示**



Part 1 手机远控中开源技术的痛点

手机远控 : <http://opentest.360.cn/>

360开测 手机远控 用户手册

可用时间: 00:08:35 + 结束租用

liangxiaojing-s@360.cn | 退出

输入文本

← 退格 发送

100% 9:35

会员助手

主题

Xiaohu

5

应用工具

浏览器

拨号

联系人

信息

相机

Chrome

清晰

流畅

手机基本信息

系统信息:

Android版本:

Android 8.1

SDK版本:

27

ABI:

arm64-v8a

内存大小:

5.6GB

是否ROOT:

未ROOT

硬件信息:

厂商:

HUAWEI

Model:

EML-AL00

CPU类型:

HISilicon Kirin 970

屏幕信息:

宽度:

1080

长度:

2244

安装

截图

logcat

shell

性能数据

文件管理

H5

远程调试

手机信息

脚本录制

屏幕显示

原方式：minicap

- 1、调用Android的私有Api
- 2、工具需适配多个Android版本

```
bin
├── arm64-v8a
│   ├── minicap
│   └── minicap-nopie
├── armeabi-v7a
│   ├── minicap
│   └── minicap-nopie
├── x86
│   ├── minicap
│   └── minicap-nopie
└── x86_64
    ├── minicap
    └── minicap-nopie
```

```
shared
├── android-22
│   ├── arm64-v8a
│   │   └── minicap.so
│   ├── armeabi-v7a
│   │   └── minicap.so
│   ├── x86
│   │   └── minicap.so
│   └── x86_64
│       └── minicap.so
├── android-9
│   └── armeabi-v7a
│       └── minicap.so
└── android-M
    ├── arm64-v8a
    │   └── minicap.so
    ├── armeabi-v7a
    │   └── minicap.so
    ├── x86
    │   └── minicap.so
    └── x86_64
        └── minicap.so
```

minicap的痛点

依赖于stf
更新



不同Android
版本适配

不支持旋转



个别手机崩
溃不适用

新问题如何解决？

新问题的产生：Android P发布
如何解决：





Part 2 视频流技术的应用框架与优势

应用框架



优势



无需进行Android版本的适配测试



减少平台依赖



传输过程优于minicap

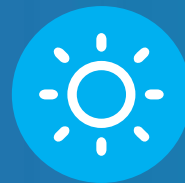


Part 3 实施方案与效果展示

实现方案



srcpy jar
传递屏幕的流信息



h264解析
将流字节分割为h264
数据



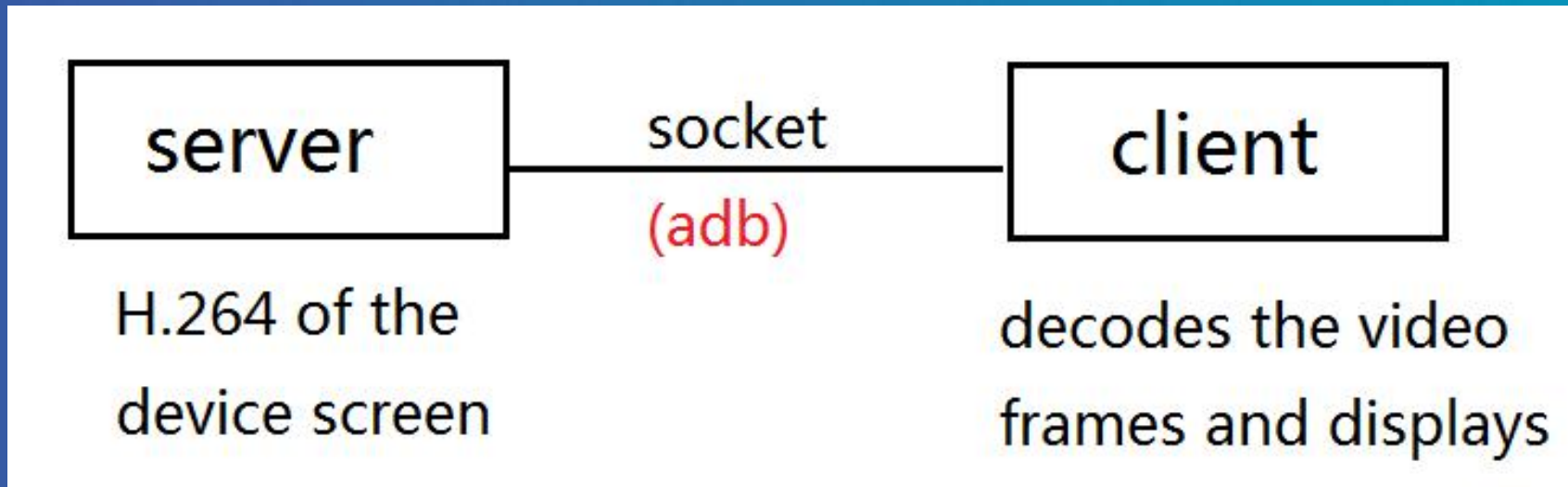
二进制格式传输
h264数据传输



Broadway
接收h264并显示

scrcpy

strcpy copies a string; **scrcpy** copies a screen.



视频数据显示

RTMP VS H264

H.264数据

NAL单元序列如下：



每个NAL单元包括一个原始字节序列负荷(RBSP, Raw Byte Sequence Payload)、一组对应于视频编码的NAL头信息。

NAL的单元有SPS、PPS、SEI、IDR_SLICE类型

二进制格式传输与Broadway

server

```
...  
self.write_message(data, binary=True)  
...
```



前端播放插件：<https://github.com/mbebenita/Broadway>

插件使用例子：<https://github.com/131/h264-live-player>

```
var player = new Player({  
  useWorker: true,  
  reuseMemory: true,  
});  
document.querySelector('.container').appendChild(player.canvas);  
  
var uri = "ws://127.0.0.1/chat";  
var h264_stream = new WebSocket(uri);  
h264_stream.binaryType = "arraybuffer";  
  
h264_stream.onmessage = function (evt) {  
  player.decode(evt.data);  
};
```

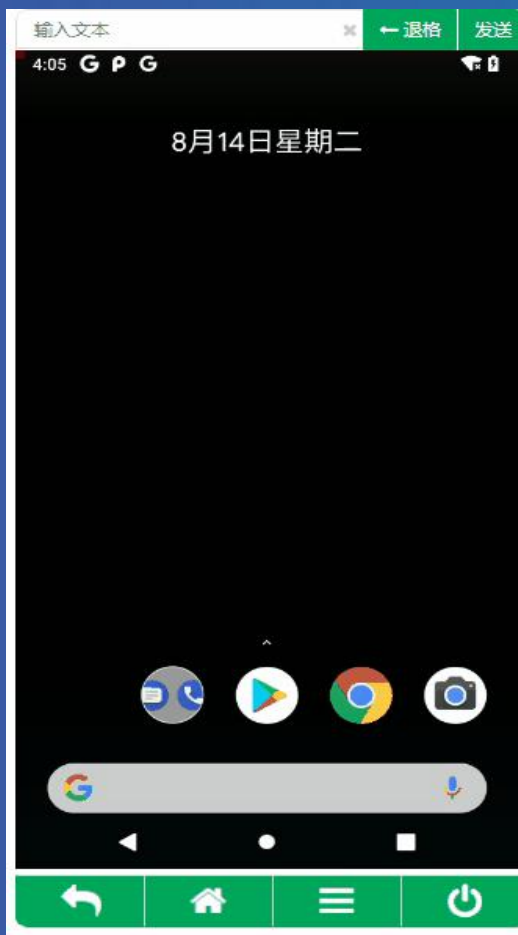
效果展示



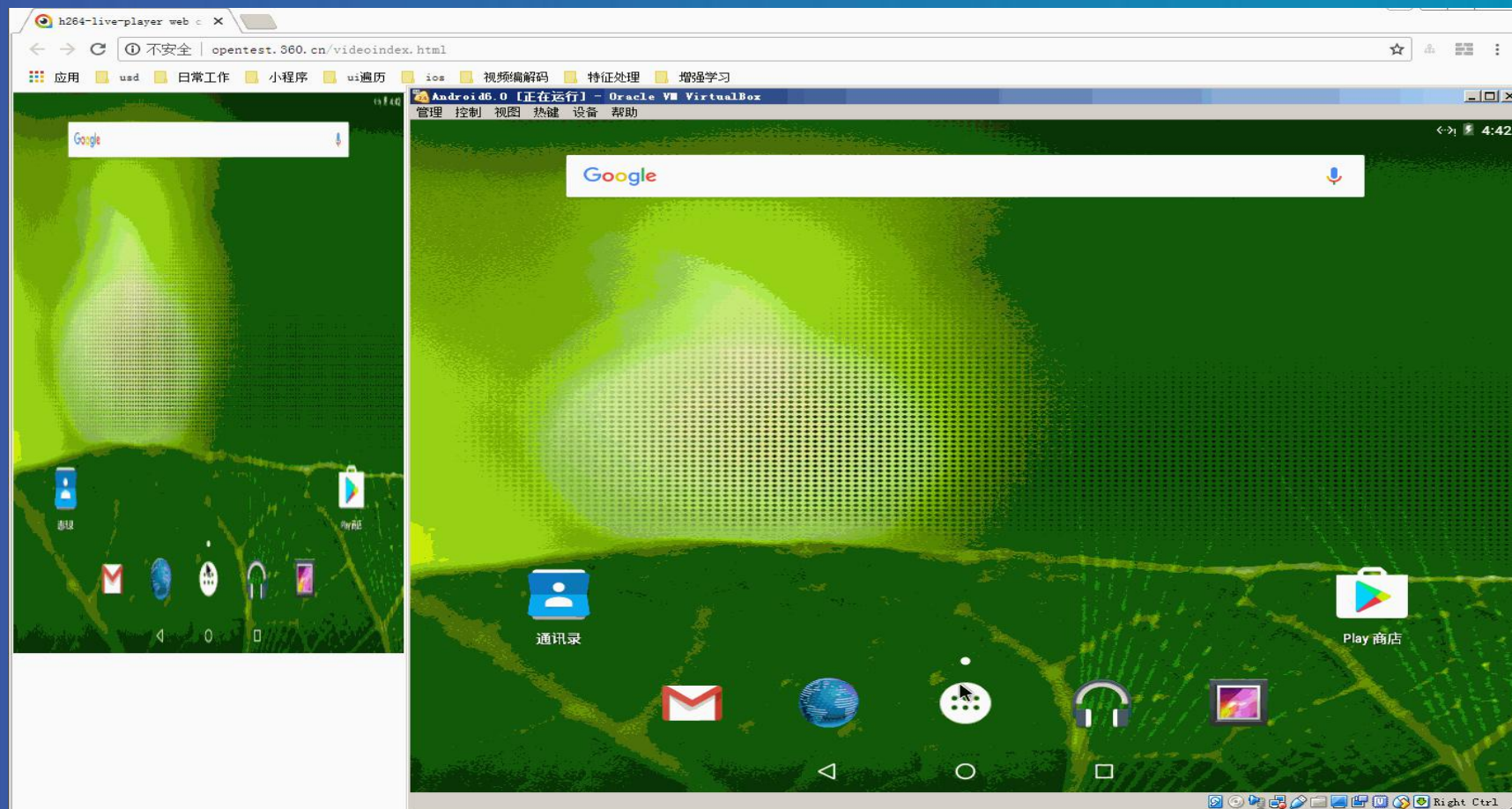
☐ 清晰 ☒ 流畅



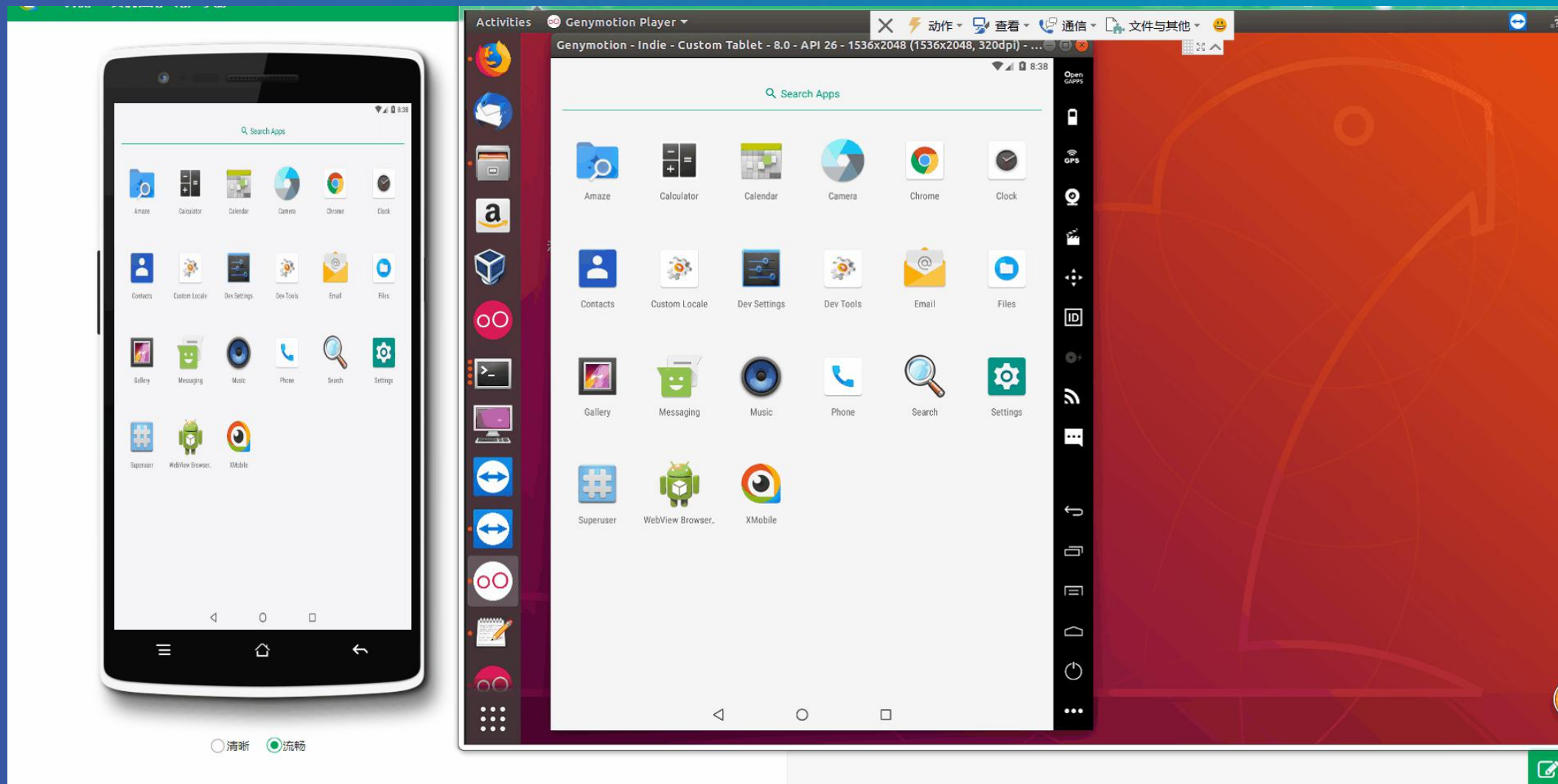
效果展示



效果展示



效果展示



优化

减小video分辨率，降低视频流数据传送

```
public void streamScreen(Device device, OutputStream outputStream) throws IOException {
    MediaFormat format = createFormat(bitRate, frameRate, iFrameInterval);
    device.setRotationListener(this);
    boolean alive;
    try {
        do {
            MediaCodec codec = createCodec();
            IBinder display = createDisplay();
            Rect contentRect = device.getScreenInfo().getContentRect();
            Rect videoRect = device.getScreenInfo().getVideoSize().toRect();
            Rect videoRect = new Rect(0, 0, __videoRect.width()/2, __videoRect.height()/2);
            setSize(format, videoRect.width(), videoRect.height());
            configure(codec, format);
            Surface surface = codec.createInputSurface();
            setDisplaySurface(display, surface, contentRect, videoRect);
        } while (alive);
    } catch (IOException e) {
        // ...
    }
}
```



谢谢！