密级状态： 绝密(　　) 秘密(　　) 内部资料(　　) 公开(　√)
Security Class: Top-Secret (　　) Secret (　　) Internal (　　) Public (√)

Rockchip_Wifidisplay_使用说明

Rockchip_Wifidisplay_Introduction

（技术部，第二系统产品部）

(Technical Department, R & D Dept. II）

文件状态： **Status:** [　] 草稿 [　] Draft [　] 正在修改 [　] Modifying [√] 正式发布 [√] Released	**文件标识：** **File No.:**	RK-SM-YF-250
	当前版本： **Current Version：**	V1.0
	作　　者： **Author:**	王剑辉 Wang Jianhui
	完成日期： **Finish Date:**	2020-04-03
	审　　核： **Auditor:**	邓训金、张崇松 Deng Xunjin/Zhang Chongsong
	审核日期： **Finish Date:**	2020-04-03

福州瑞芯微电子股份有限公司
Fuzhou Rockchip Electronics Co., Ltd.

福州瑞芯微电子股份有限公司
Fuzhou Rockchip Electronics Co., Ltd.

地址：	福建省福州市铜盘路软件园 A 区 18 号
网址：	www.rock-chips.com
客户服务电话：	+86-4007-700-590
客户服务传真：	+86-591-83951833
客户服务邮箱：	fae@rock-chips.com

Fuzhou Rockchip Electronics Co., Ltd.
Address: No. 18 Building, A District, No.89,software Boulevard Fuzhou,Fujian,PRC

Website:	www.rock-chips.com
Customer service tel.:	+86-4007-700-590
Customer service fax:	+86-591-83951833
Customer service e-mail:	fae@rock-chips.com

版 本 历 史 Revision History

版本号 Version no.	作者 Author	修改日期 Revision Date	修改说明 Revision description	备注 Remark
V1.0	王剑辉 Wang Jianhui	2020-04-03	发布初版 Initial version release	

目 录 Contents

前 言

概述

本文档主要介绍 Wifidisplay 原理，以及 Rockchip 平台 Android 版本无线投屏（Wifidisplay）问题排查、debug 手段，旨在帮助软件开发工程师对 Rockchip 平台 Android 版本的 wifidisplay 模块 debug、性能优化等。

读者对象

本文档主要适用于以下工程师：

技术支持工程师

软件开发工程师

1 Wifidisplay（Miracast）理论知识简介

1.1 Wifidisplay 功能框架图

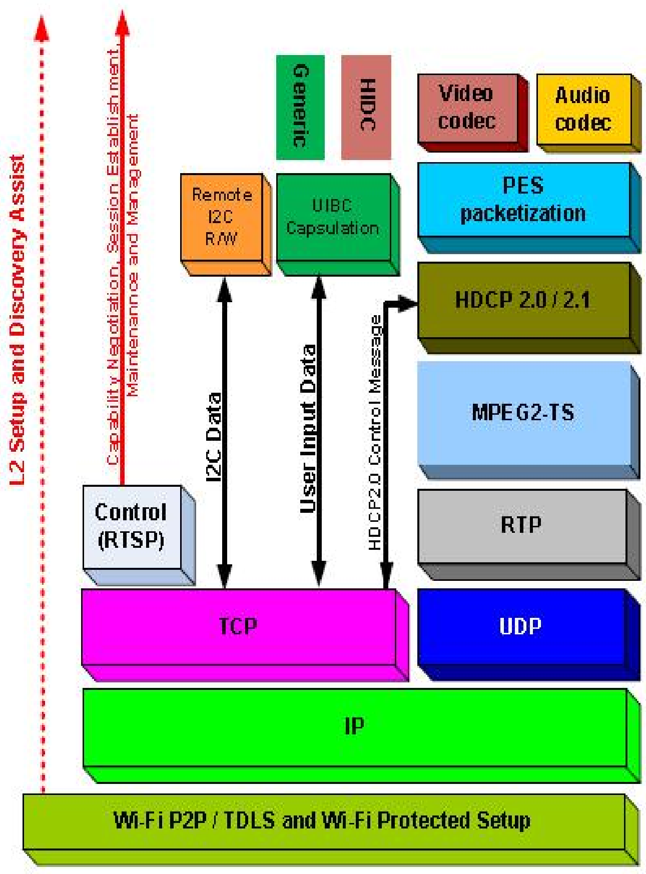


图 1 WFD 涉及的技术及协议框图

1.2 常见名称介绍

Miracast 是 Wi-Fi 联盟（Wi-Fi Alliance）对支持 Wi-Fi Display 功能的设备的认证名称。通过 Miracast 认证的设备将在最大程度内保持对 Wi-Fi Display 功能的支持和兼容。

Wifidisplay（WFD）是一种基于 Wi-Fi Direct 技术的设备互联并实时共享屏幕内容的解决方案，它拓宽了 Wi-Fi 技术的覆盖范围，通过设备之间的直接连接，给用户带来新的连接和共享体验。接收端设备可通过发现、配对、连接并呈现源设备的多媒体内容。

UIBC 是 User Input Back Channel，即反向控制功能，sink 端设备控制 source。

HDCP 是高带宽数字内容保护。

Source 是能提供多媒体内容传输的设备。

Sink 是能接受多媒体内容并将其呈现的设备。

Session 是一个 WFD 在传输和接收设备之间连接。

TDLS 是 Tunneled Direct Link Setup 一项 802.11 协议，在两个连接到同一个 Ap 的客户端设备之间建立的直接连接，Wifidisplay2.0 协议才支持这种连接方式，目前 RK 平台的 Wifidisplay 是 1.0 版本的协议，所以不支持 TDLS 传输方式，都是用 Wifi P2P 传输方式。

Wi-Fi Direct 是一种 P2P 的无线互联技术，它所建立的网络（piconet）是一种改进型的 ad hoc 网络，采用无线通讯模式。

1.3 Wifidisplay 协议流程

建立 WifiDisplay 主要步骤如下：

1. WFD Device Discovery（WFD 设备发现）
2. WFD Service Discovery (Optional)（WFD 服务发现（可选））
3. Device Selection（设备选择）
4. WFD Connection Setup（WFD 连接）
5. WFD Capability Negotiation（WFD 能力协商）
6. WFD Session Establishment（WFD 会话建立）
7. User Input Back Channel Setup (Optional)（UIBC 反向控制）
8. Link Content Protection Setup (Optional)（内容保护，即数据加密）
9. Payload Control（负载控制）
10. WFD Source and WFD Sink standby (Optional)
11. WFD Session Teardown（会话终止）

WFD 设备通过 wifiP2P 连接后，Sink 端与 Source 端建立 TCP 连接，Sink 端为 Client 而 Source 端为 Server。默认端口为 7236，执行的协议为 RTSP 协议。建立连接后进行 RTSP 协商。步骤 6，协商成功后建立会话；步骤 7，UIBC 通道建立，用于 Sink 端反向控制 Source 端，该步骤为可选实现；步骤 8，对传输的内容做加密保护（HDCP），步骤 9，开始音频及视频流的传输与控制，Payload Control：传输过程中，设备可根据无线信号的强弱，甚至设备的电量状况来动态调整传输数据和格式。可调整的内容包括压缩率，视音频格式，分辨率等内容。步骤 11，会话终止。

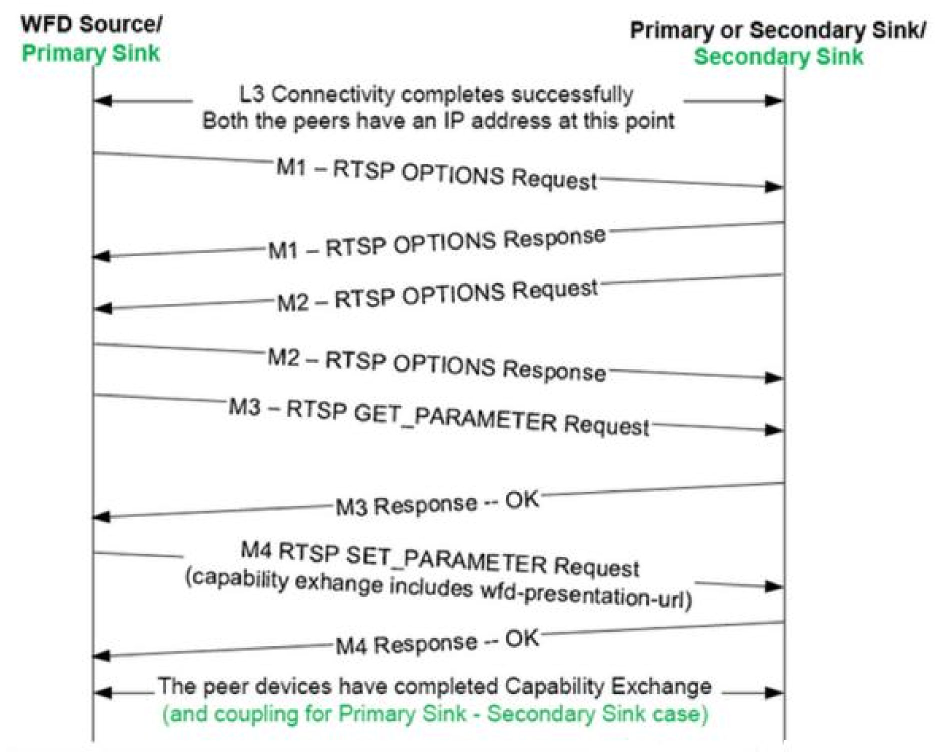

WFD Source/
Primary Sink
Primary or Secondary Sink/
Secondary Sink
L3 Connectivity completes successfully
Both the peers have an IP address at this point
M1 – RTSP OPTIONS Request
M1 – RTSP OPTIONS Response
M2 – RTSP OPTIONS Request
M2 – RTSP OPTIONS Response
M3 – RTSP GET_PARAMETER Request
M3 Response -- OK
M4 RTSP SET_PARAMETER Request
(capability exhange includes wfd-presentation-url)
M4 Response -- OK
The peer devices have completed Capability Exchange
(and coupling for Primary Sink - Secondary Sink case)

图 2 会话建立及协商过程图

RTSP M1 和 M2 主要协商 Source 和 Sink 都支持的 RTSP methods。RTSP M3 和 M4 主要协商 Source 和 Sink 在会话中使用的参数。

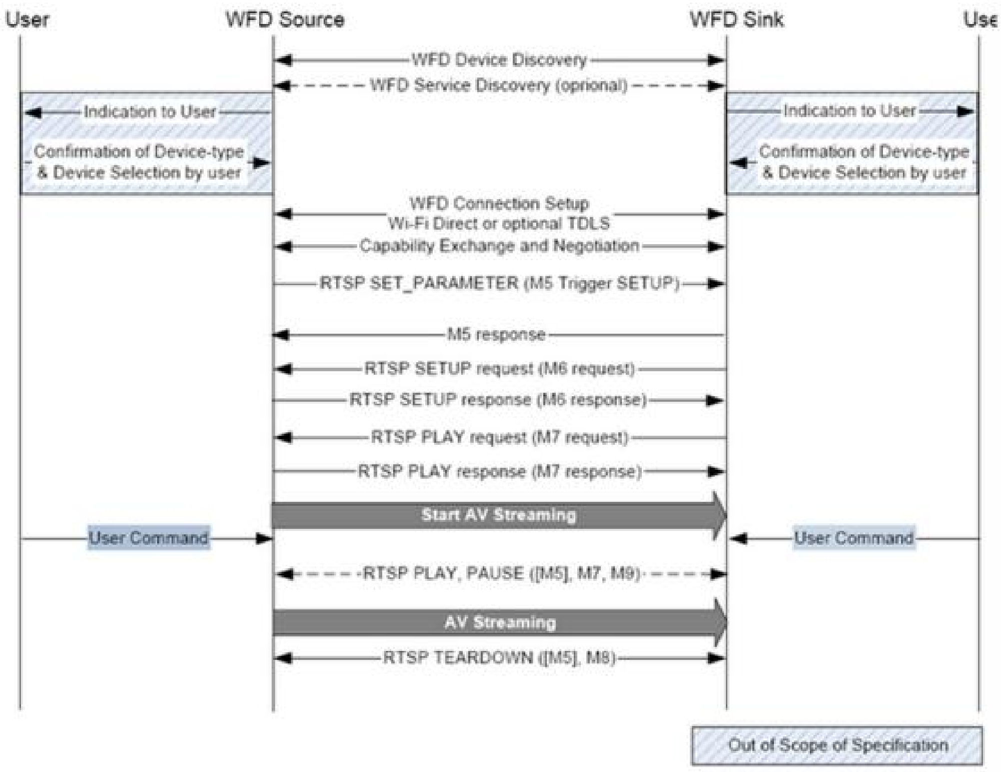

User
WFD Source
WFD Sink
Use
WFD Device Discovery
WFD Service Discovery (oprional)
Indication to User
Confirmation of Device-type & Device Selection by user
Indication to User
Confirmation of Device-type & Device Selection by user
WFD Connection Setup
Wi-Fi Direct or optional TDLS
Capability Exchange and Negotiation
RTSP SET_PARAMETER (M5 Trigger SETUP)
M5 response
RTSP SETUP request (M6 request)
RTSP SETUP response (M6 response)
RTSP PLAY request (M7 request)
RTSP PLAY response (M7 response)
Start AV Streaming
User Command
User Command
RTSP PLAY, PAUSE ([M5], M7, M9)
AV Streaming
RTSP TEARDOWN ([M5], M8)
Out of Scope of Specification

图 3 RTSP 协议控制图

RTSP 协议控制中主要有以下几种状态 SETUP、PLAY、PAUSE、TEARDOWN，当 Wifidisplay source 和 Wifidisplay sink 之间成功地完成了 RTSP M7 请求和响应消息的交换时，source 和 sink 就

建立了 WFD 会话。

对于 WifiDisplay 会话管理有以下模型可供参考，该结构大致分为四个层次，UI、Session Policy Management、协议实现层及基于 Wifi 的网络传输层。在协议实现层中主要分为几个模块 WFD Discovery、WFD Link Establishment、UIBC、Capability Negotiation、Session/Stream Control 等。

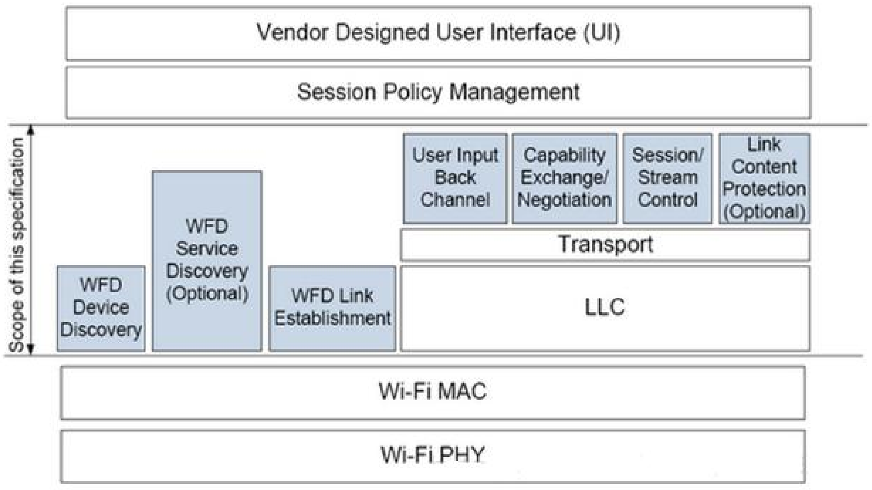

Vendor Designed User Interface (UI)
Session Policy Management
Scope of this specification
User Input Back Channel
Capability Exchange/ Negotiation
Session/ Stream Control
Link Content Protection (Optional)
Transport
WFD Service Discovery (Optional)
WFD Device Discovery
WFD Link Establishment
LLC
Wi-Fi MAC
Wi-Fi PHY

图 4 WFD 设备会话管理的模型

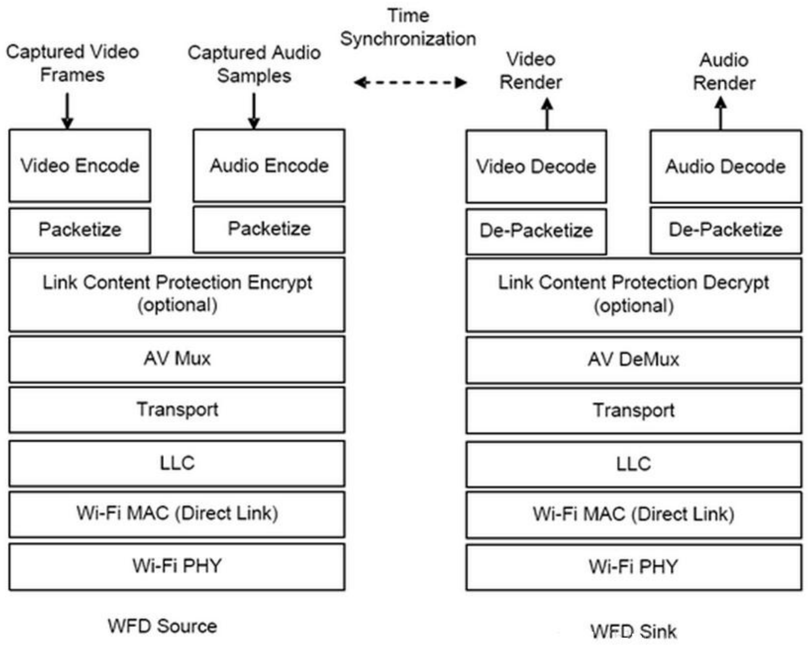


图 5 音频及视频流控制模型

实时流协议 RTSP 是一个应用层协议，用于控制具有实时特性的数据（例如多媒体流）的传送。RTSP 协议一般与 RTP/RTCP 和 RSVP 等底层协议一起协同工作，提供基于 Internet 的整套的流服务。它可以选择发送通道（例如：UDP、组播 UDP 和 TCP）和基于 RTP 的发送机制。它可以应用于组播和点播。RTP, RTCP, RSVP 定义如下：

1. 实时传输协议 RTP(Real-time Transport protocol)
2. 实时传输控制协议 RTCP(Real-time Transport Control protocol)
3. 实时流协议 RTSP(Real Time Streaming protocol)
4. 资源预留协议 RSVP(Resource Reserve Protocol)

客户端与服务器运行实时流控制协议 RTSP，以对该流进行各种 VCR 控制信号的交换，如播放（PLAY）、停止（PAUSE）、快进、快退等。当服务完毕，客户端提出拆线（TEARDOWN）请求。服务器使用 RTP/UDP 协议将媒体数据传输给客户端，一旦数据抵达客户端，客户端应用程序即可播放输出。在流式传输中，使用 RTP/RTCP/UDP 和 RTSP/TCP 两种不同的通信协议在客户端和服务器间建立联系。

LLC 层是由传输驱动程序实现的，主要功能为传输可靠性保障和控制，数据包的分段与重组，数据包的顺序传输。

1.4 Wifidisplay source 显示框架

为了实现 WifiDisplay，Google 在 Android 现有显示系统的基础上加入虚拟设备的支持，下图给出了 Android 显示系统的架构图。

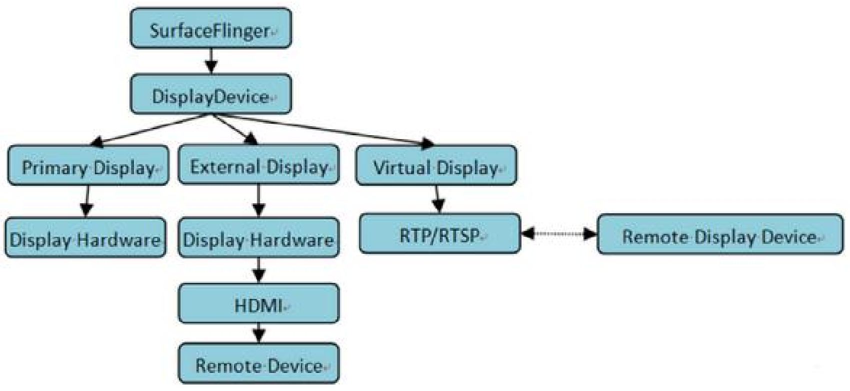


图 6 DisplayDevice 显示架构图

1.5 Wifidisplay Android 源码

涉及 Wifidisplay 模块的源码路径（以 Android10.0 为例）：

Display Manager

framework/base/services/core/java/com/android/server/display/

WifiDisplayController.java

WifiDisplayAdapter.java

PersistentDataStore.java

DisplayManagerService.java

RTSP

frameworks/av/media/libstagefright/rtsp

MediaRouter

frameworks/base/media/java/android/media/MediaRouter.java

Wi-Fi P2P

frameworks/base/wifi/java/android/net/wifi/p2p/

Wi-Fi Display source 端逻辑代码

frameworks/av/media/libstagefright/wifi-display

WifiDisplaySource.cpp--Source 端的协议交互

PlaybackSession.cpp--会话过程管理

MediaPuller.cpp--镜像媒体读取

Converter.cpp--编码

TSPacketizer.cpp--TS 打包

Sender.cpp--RTP 打包发送

Wi-Fi Display sink 端 app 代码（需要发邮件找 fae 窗口申请）

WifiDisplaySink.cpp--负责 SINK 端的协议

RTPSink.cpp--RTP 接收

TunnelRenderer.cpp--镜像数据的呈现

1.6 Android WifiDisplay 实现

1.6.1 Source 端实现

当用户点击了 optionMenu 中 enable wifi display 选项时，会触发相关的设备扫描及更新操作，在 WifiDisplaySettings 和 WifiDisplayController 都有注册 ContentObserver 来监控这个值的变化。触发设备扫描是在 WifiDisplayController 中通过 updateWfdEnableState() 进行的，最终通过 WifiP2pManager.requestPeers 来完成设备的扫描工作，获取扫描到的设备列表是在 WifiDisplaySettings 通过 update(int changes) 进行的。对于设备连接状态的管理主要通过 updateConnection()来进行。由于设备的连接过程是一个异步过程，所以在设备操作相关的过程中会反复调用 updateConnection()来判定设备状态及更新连接操作。

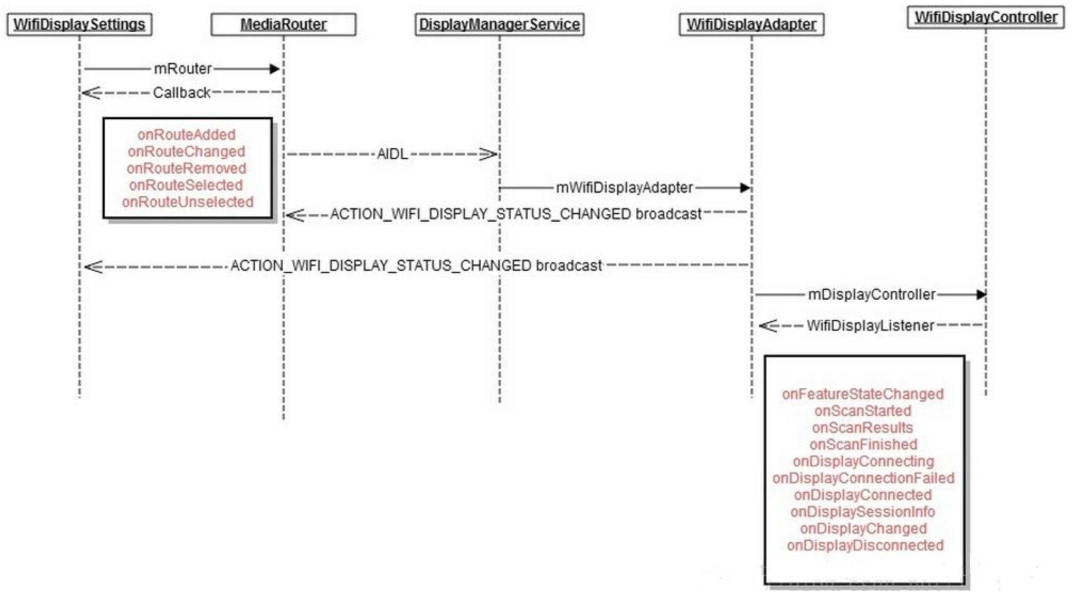


图 7 设备发现流程图

设备列表更新及管理通过下面函数 update 实现：

```
    private void update(int changes) {
        boolean invalidateOptions = false;
        // Update settings.
        if ((changes & CHANGE_SETTINGS) != 0) {
            mWifiDisplayOnSetting = Settings.Global.getInt(getContentResolver(),
                    Settings.Global.WIFI_DISPLAY_ON, 0) != 0;
            mWifiDisplayCertificationOn =
                Settings.Global.getInt(getContentResolver(),
                    Settings.Global.WIFI_DISPLAY_CERTIFICATION_ON, 0) != 0;
            mWpsConfig = Settings.Global.getInt(getContentResolver(),
                    Settings.Global.WIFI_DISPLAY_WPS_CONFIG, WpsInfo.INVALID);
            // The wifi display enabled setting may have changed.
            invalidateOptions = true;
        }
        // Update wifi display state.
        if ((changes & CHANGE_WIFI_DISPLAY_STATUS) != 0) {
            mWifiDisplayStatus = mDisplayManager.getWifiDisplayStatus();
            // The wifi display feature state may have changed.
            invalidateOptions = true;
        }
        // Rebuild the routes.
        final PreferenceScreen preferenceScreen = getPreferenceScreen();
        preferenceScreen.removeAll();
        // Add all known remote display routes.
        final int routeCount = mRouter.getRouteCount();
        for (int i = 0; i < routeCount; i++) {
            MediaRouter.RouteInfo route = mRouter.getRouteAt(i);
            if (route.matchesTypes(MediaRouter.ROUTE_TYPE_REMOTE_DISPLAY)) {
                preferenceScreen.addPreference(createRoutePreference(route));
            }
        }
        // Additional features for wifi display routes.
        if (mWifiDisplayStatus != null
                && mWifiDisplayStatus.getFeatureState() ==
        WifiDisplayStatus.FEATURE_STATE_ON) {
            // Add all unpaired wifi displays.
            for (WifiDisplay display : mWifiDisplayStatus.getDisplays()) {
                if (!display.isRemembered() && display.isAvailable()
                        && !display.equals(mWifiDisplayStatus.getActiveDisplay())) {
                            preferenceScreen.addPreference(
                    new UnpairedWifiDisplayPreference(getActivity(), display));
                }
            }
            // Add the certification menu if enabled in developer options.
            if (mWifiDisplayCertificationOn) {
                buildCertificationMenu(preferenceScreen);
            }
        }
        // Invalidate menu options if needed.
        if (invalidateOptions) {
            getActivity().invalidateOptionsMenu();
        }
    }
```

1.6.2 Sink 端实现

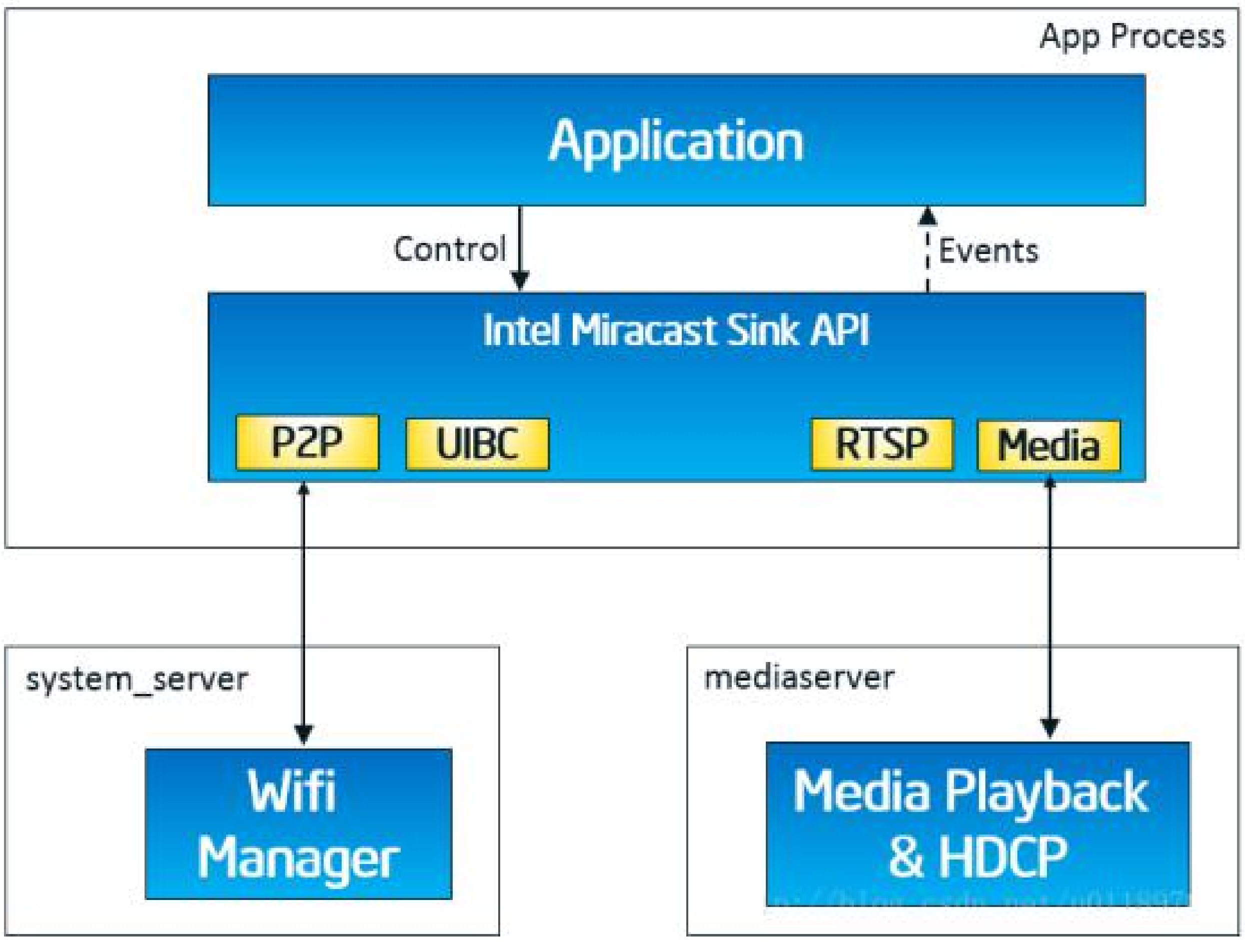


图 8 设备发现流程图

图 8 给出了 sink 端的实现框架图，从框架图可以看出 APP 主要和 Sink API 交互，Sink API 和框架服务中的 Wifi server 及 mediaserver 交互，APP 通过 Control interface 进行 WFD 相关的控制操

作，底层状态的接收则通过 Events interface，也就是一些相关的回调方法来处理。

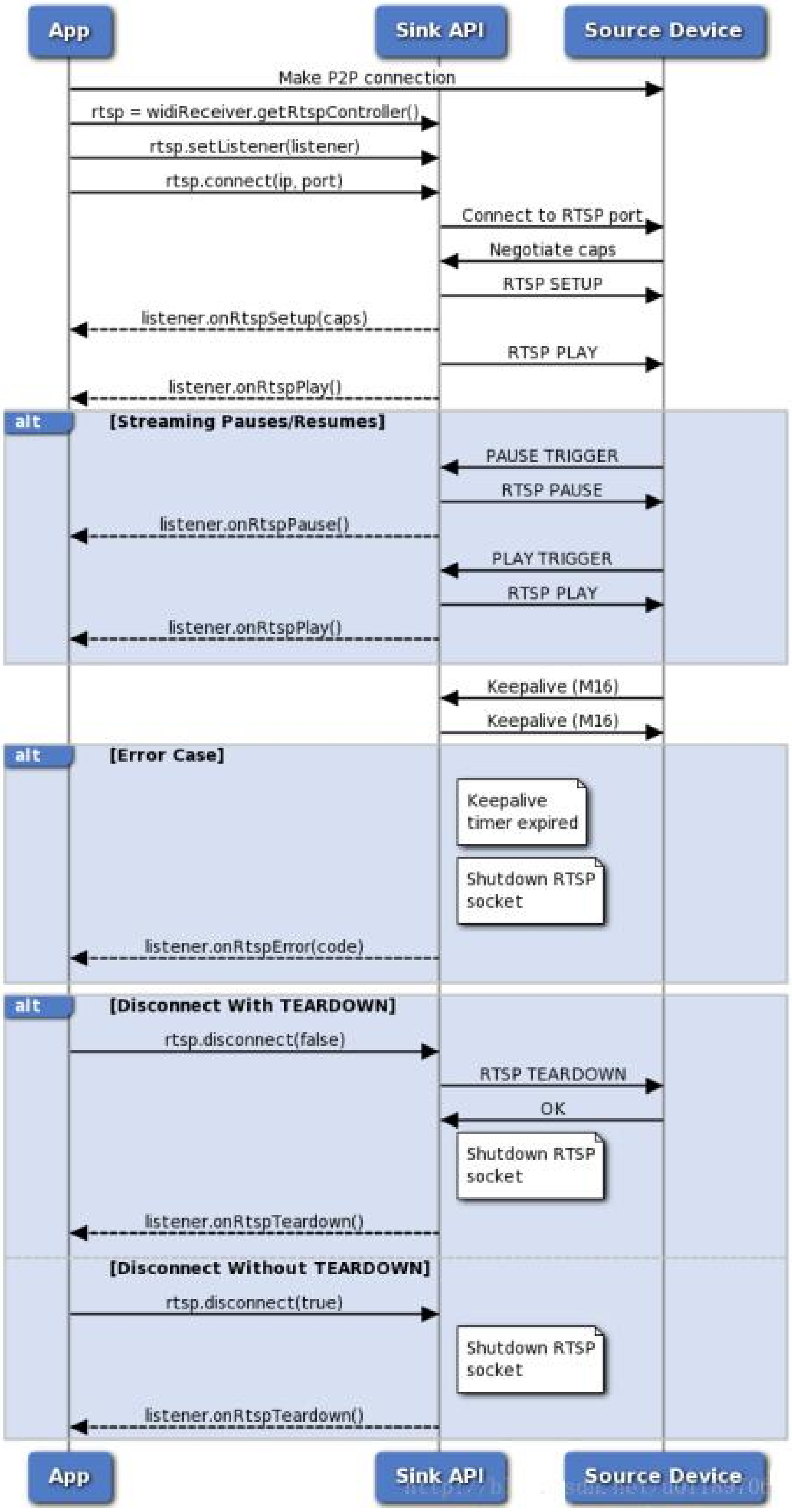


图 9 RTSP 会话流程图

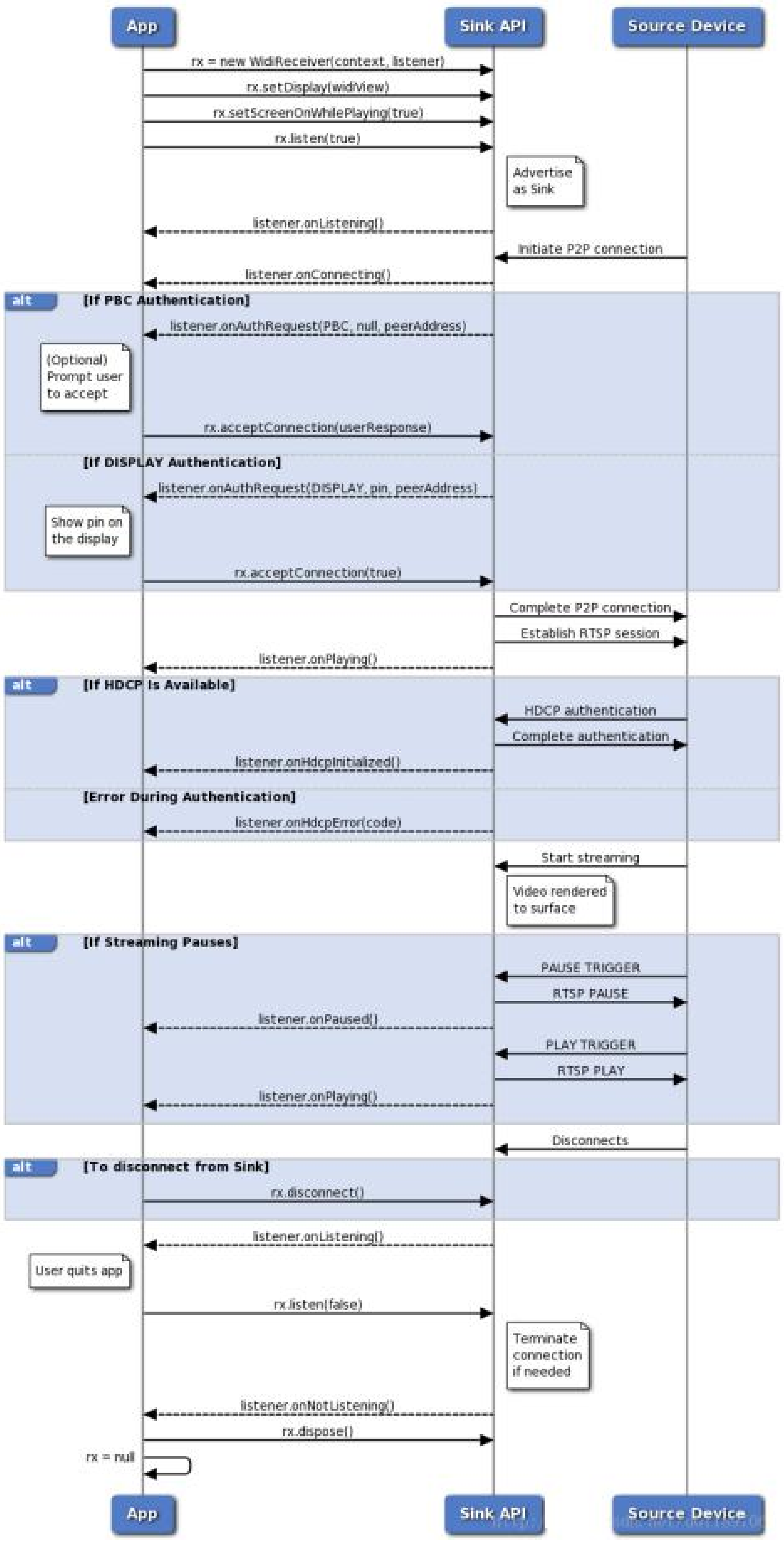

App
Sink API
Source Device
rx = new WidiReceiver(context, listener)
rx.setDisplay(widiView)
rx.setScreenOnWhilePlaying(true)
rx.listen(true)
Advertise as Sink
listener.onListening()
Initiate P2P connection
listener.onConnecting()
alt
[If PBC Authentication]
listener.onAuthRequest(PBC, null, peerAddress)
(Optional) Prompt user to accept
rx.acceptConnection(userResponse)
[If DISPLAY Authentication]
listener.onAuthRequest(DISPLAY, pin, peerAddress)
Show pin on the display
rx.acceptConnection(true)
Complete P2P connection
Establish RTSP session
listener.onPlaying()
alt
[If HDCP Is Available]
HDCP authentication
Complete authentication
listener.onHdcpInitialized()
[Error During Authentication]
listener.onHdcpError(code)
Start streaming
Video rendered to surface
alt
[If Streaming Pauses]
PAUSE TRIGGER
RTSP PAUSE
listener.onPaused()
PLAY TRIGGER
RTSP PLAY
listener.onPlaying()
Disconnects
alt
[To disconnect from Sink]
rx.disconnect()
listener.onListening()
User quits app
rx.listen(false)
Terminate connection if needed
listener.onNotListening()
rx.dispose()
rx = null
App
Sink API
Source Device

图 10 WFD 控制流程图

图 9 给出了 Intel 实现的 sink 端的 RTSP 会话管理流程图，RTSP 的协议实现主要通过 C++实现，对于协商后相关的状态反馈通过回调函数完成，如果想进一步了解相关的流程，请查看相关的代码。

图 10 给出了 WFD 会话管理的流程图，WFD 中除了 RTSP 的实现，还包括连接认证（Connection Auth）、视频流加密及解密(HDCP)、UIBC 实现等。

1.7 Wifidisplay 使用说明

使用说明分为 source 端和 sink 端。

wifidisplay source 端源码集成到系统 SDK 里面，目前所有的 RK 平台 Android SDK 源码都带有 wifidisplay source 功能。使用方法：首先准备一台带有 wifidisplay sink 功能的电视或者带有 sink 功能的 RK 平板，然后打开原生 Settings->设备->显示->投射（Android6.0 版本），打开原生 Settings -> Connected Devices -> Cast -> Enable wireless display（Android8.1 版本）。

wifidisplay sink 端是集成一个单独 APK，默认 SDK 上没有源码，需要源码的话可以在 ftp://www.rockchip.com 下载，然后放 SDK 源码 package/apps 目录下面编译，烧写固件，打开 WifiDisplay.apk。

2 Wifidisplay Debug 介绍

2.1 iperf 测试 P2P 吞吐率和丢包率

首先确认 RK 平台的设备是用于 sink（接收）端设备还是用于 source（发送）端设备。用于做 sink 端设备时，需要测试设备的下行吞吐率和丢包率；用于做 source 端设备时，需要测试设备的上行吞吐率和丢包率。下面分别说明设备吞吐率和丢包率测试方法和步骤：

RK 平台设备做 sink，其他平台手机做 source（如 nexus4）：

1）两端设备都安装好 iperf.apk

2）两端设备 wifi direct 连接

3）打开 iperf，在 RK 平台设备端（sink 端）输入 -u -s -i 1 -w 256K，然后选择“打开”按钮；在手机（source）端的 iperf 窗口中输入./iperf -u -c 10.255.255.251 -b 10M -i 1 -w 256K -t 23123

备注：10.255.255.251 是 RK 平台（sink）端的 p2p 直连状态下的 ip 地址，实际测试过程中，替换使用。

4）在手机（source）端点击“打开”按钮后，可在 RK 平台端看到一些 report，最后一项是丢包率。假如两个设备靠得很近，正常是不会出现丢包的（低概率出现 0.1%以下丢包属于正常），丢包严重的话，基本可以确认是 Wifi 模组硬件相关问题，需要排查下 Wifi 射频参数是否校准正确，Wifi 天线是否正常匹配，环境是否有视频干扰等。

RK 平台设备做 source，其他平台电视做 sink（如小米电视）：

1）两端设备都安装好 iperf.apk

2）两端设备 wifi direct 连接

3）打开 iperf，在其他平台电视端（sink 端）输入 -u -s -i 1 -w 256K，然后选择“打开”按钮；在 RK 设备（source）端的 iperf 窗口中输入./iperf -u -c 10.255.255.251 -b 10M -i 1 -w 256K -t 23123

备注：10.255.255.251 是其他平台电视（sink）端的 p2p 直连状态下的 ip 地址，实际测试过程中，替换使用。

4）在 RK 设备（source）端点击“打开”按钮后，可在其他平台电视端看到一些 report，最后一项是丢包率。假如两个设备靠得很近，正常是不会出现丢包的（低概率出现 0.1%以下丢包属于正常），丢包严重的话，基本可以确认是 Wifi 模组硬件相关问题，需要排查下 Wifi 射频参数是否校准正确，Wifi 天线是否正常匹配，环境是否有视频干扰等。

2.2 视频出现卡顿，利用 Wireshark 分析 pcap 包

2.2.1 TCPDUMP 抓包

RK 平台设备，adb shell 进入机器终端：

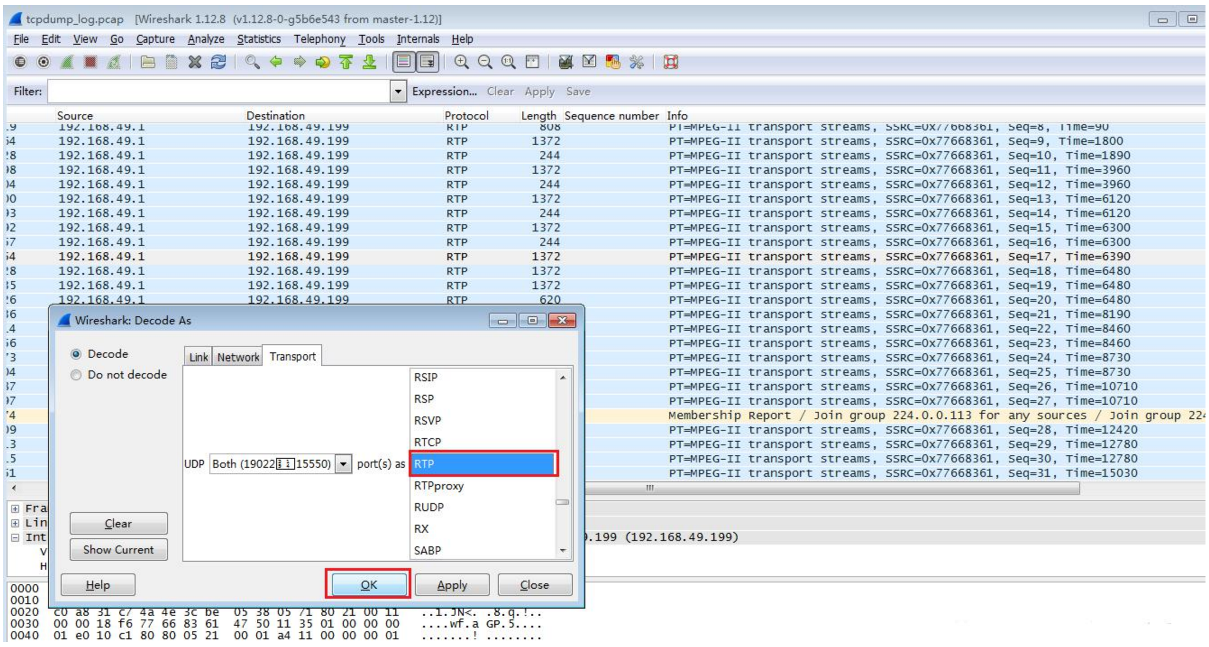
tcpdump -i any -s 0 -w /data/tcpdump_log.pcap

注意–i any 不能改成特定的 interface，因为 p2p interface 是动态生成的，有可能是 p2p-p2p0-0、p2p-p2p0-1、p2p-p2p0-2 等，抓取的 pcap 包尽量涉及从发送投屏请求到投屏图像显示一段时间，当分析投屏卡顿问题时，需要抓取到卡顿过程的 pcap 包。

2.2.2 利用 Wireshark 分析 pcap 包

下面分析一个正常的 pcap 包：

1）点击目标 UDP 包，右键选择 Decode As，将 UDP decode 成 RTP 包

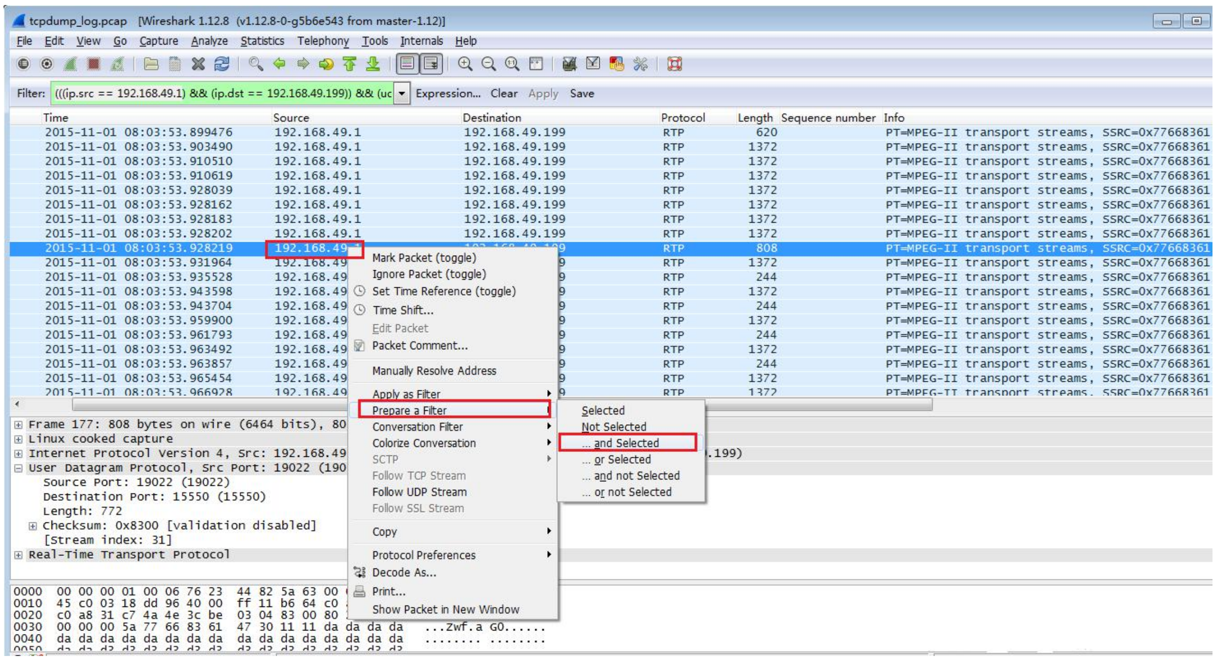

tcpdump_log.pcap [Wireshark 1.12.8 (v1.12.8-0-g5b6e543 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Wireshark: Decode As
Decode
Do not decode
Link Network Transport
RSIP
RSP
RSVP
RTCP
RTP
RTPproxy
RUDP
RX
SABP
Clear
Show Current
Help
OK
Apply
Close

```
    Source          Destination       Protocol Length Sequence number Info
.9  192.168.49.1    192.168.49.199    RTP       808                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=8, Time=90
54  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=9, Time=1800
28  192.168.49.1    192.168.49.199    RTP       244                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=10, Time=1890
98  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=11, Time=3960
04  192.168.49.1    192.168.49.199    RTP       244                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=12, Time=3960
00  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=13, Time=6120
03  192.168.49.1    192.168.49.199    RTP       244                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=14, Time=6120
02  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=15, Time=6300
57  192.168.49.1    192.168.49.199    RTP       244                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=16, Time=6300
54  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=17, Time=6390
28  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=18, Time=6480
35  192.168.49.1    192.168.49.199    RTP      1372                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=19, Time=6480
26  192.168.49.1    192.168.49.199    RTP       620                   PT=MPEG-II transport streams, SSRC=0x77668361, Seq=20, Time=6480
36                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=21, Time=8190
14                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=22, Time=8460
56                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=23, Time=8460
73                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=24, Time=8730
04                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=25, Time=8730
37                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=26, Time=10710
07                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=27, Time=10710
74                                                                    Membership Report / Join group 224.0.0.113 for any sources / Join group 224
09                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=28, Time=12420
13                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=29, Time=12780
15                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=30, Time=12780
51                                                                    PT=MPEG-II transport streams, SSRC=0x77668361, Seq=31, Time=15030

⊞ Fra
⊞ Lin
⊟ Int                                                    .199 (192.168.49.199)
    V
    H

0000
0010
0020  c0 a8 31 c7 4a 4e 3c be  05 38 05 71 80 21 00 11   ..1.JN<. .8.q.!..
0030  00 00 18 f6 77 66 83 61  47 50 11 35 01 00 00 00   ....wf.a GP.5....
0040  01 e0 10 c1 80 80 05 21  00 01 a4 11 00 00 00 01   .......! ........
```

UDP Both (19022↔15550) port(s) as RTP

2）过滤 RTP 包，过滤条件为 IP & PORT，通过 Prepare a Filter 选择

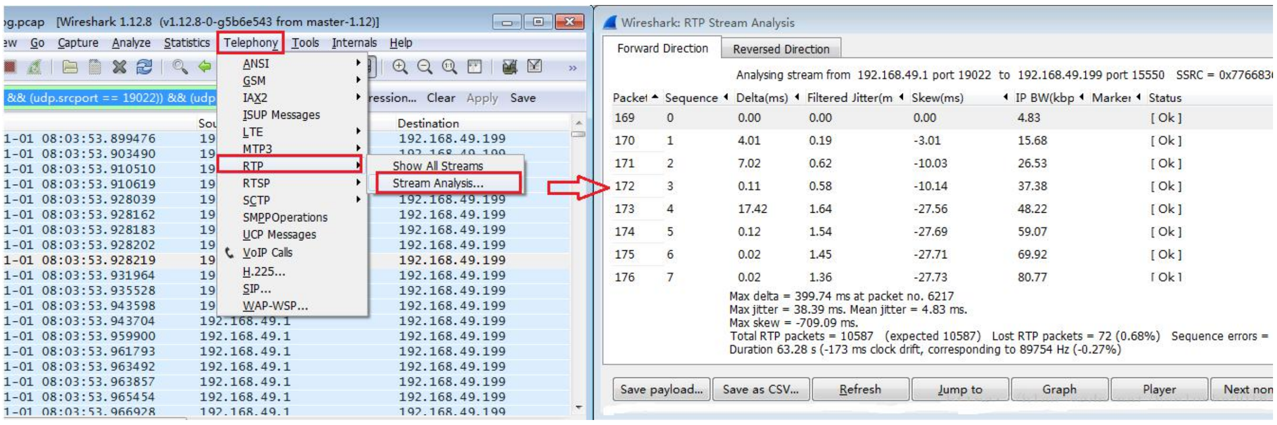
如这次的过滤条件为：(((ip.src == 192.168.49.1) && (ip.dst == 192.168.49.199)) && (udp.srcport == 19022)) && (udp.dstport == 15550)



```
Time                        Source          Destination      Protocol  Length  Sequence number  Info
2015-11-01 08:03:53.899476  192.168.49.1    192.168.49.199   RTP        620    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.903490  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.910510  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.910619  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.928039  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.928162  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.928183  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.928202  192.168.49.1    192.168.49.199   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.928219  192.168.49                       RTP        808    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.931964  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.935528  192.168.49                   9   RTP        244    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.943598  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.943704  192.168.49                   9   RTP        244    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.959900  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.961793  192.168.49                   9   RTP        244    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.963492  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.963857  192.168.49                   9   RTP        244    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.965454  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361
2015-11-01 08:03:53.966928  192.168.49                   9   RTP       1372    PT=MPEG-II transport streams, SSRC=0x77668361

⊞ Frame 177: 808 bytes on wire (6464 bits), 80
⊞ Linux cooked capture
⊞ Internet Protocol Version 4, Src: 192.168.49                .199)
⊟ User Datagram Protocol, Src Port: 19022 (190
    Source Port: 19022 (19022)
    Destination Port: 15550 (15550)
    Length: 772
  ⊞ Checksum: 0x8300 [validation disabled]
    [Stream index: 31]
⊞ Real-Time Transport Protocol

0000  00 00 00 01 00 06 76 23  44 82 5a 63 00
0010  45 c0 03 18 dd 96 40 00  ff 11 b6 64 c0
0020  c0 a8 31 c7 4a 4e 3c be  03 04 83 00 80
0030  00 00 00 5a 77 66 83 61  47 30 11 11 da da da da   ...Zwf.a G0......
0040  da da da da da da da da  da da da da da da da da   ........ ........
```

3）对过滤后的 RTP 包进行分析：Stream Analysis

可以从分析结果看到丢包率仅为 0.68%，乱序只有 3 次，播放应该是很流畅的



```
1-01 08:03:53.899476   19   192.168.49.199
1-01 08:03:53.903490   19   192.168.49.199
1-01 08:03:53.910510   19
1-01 08:03:53.910619   19
1-01 08:03:53.928039   19   192.168.49.199
1-01 08:03:53.928162   19   192.168.49.199
1-01 08:03:53.928183   19   192.168.49.199
1-01 08:03:53.928202   19   192.168.49.199
1-01 08:03:53.928219   19   192.168.49.199
1-01 08:03:53.931964   19   192.168.49.199
1-01 08:03:53.935528   19   192.168.49.199
1-01 08:03:53.943598   19   192.168.49.199
1-01 08:03:53.943704   192.168.49.1   192.168.49.199
1-01 08:03:53.959900   192.168.49.1   192.168.49.199
1-01 08:03:53.961793   192.168.49.1   192.168.49.199
1-01 08:03:53.963492   192.168.49.1   192.168.49.199
1-01 08:03:53.963857   192.168.49.1   192.168.49.199
1-01 08:03:53.965454   192.168.49.1   192.168.49.199
1-01 08:03:53.966928   192.168.49.1   192.168.49.199
```

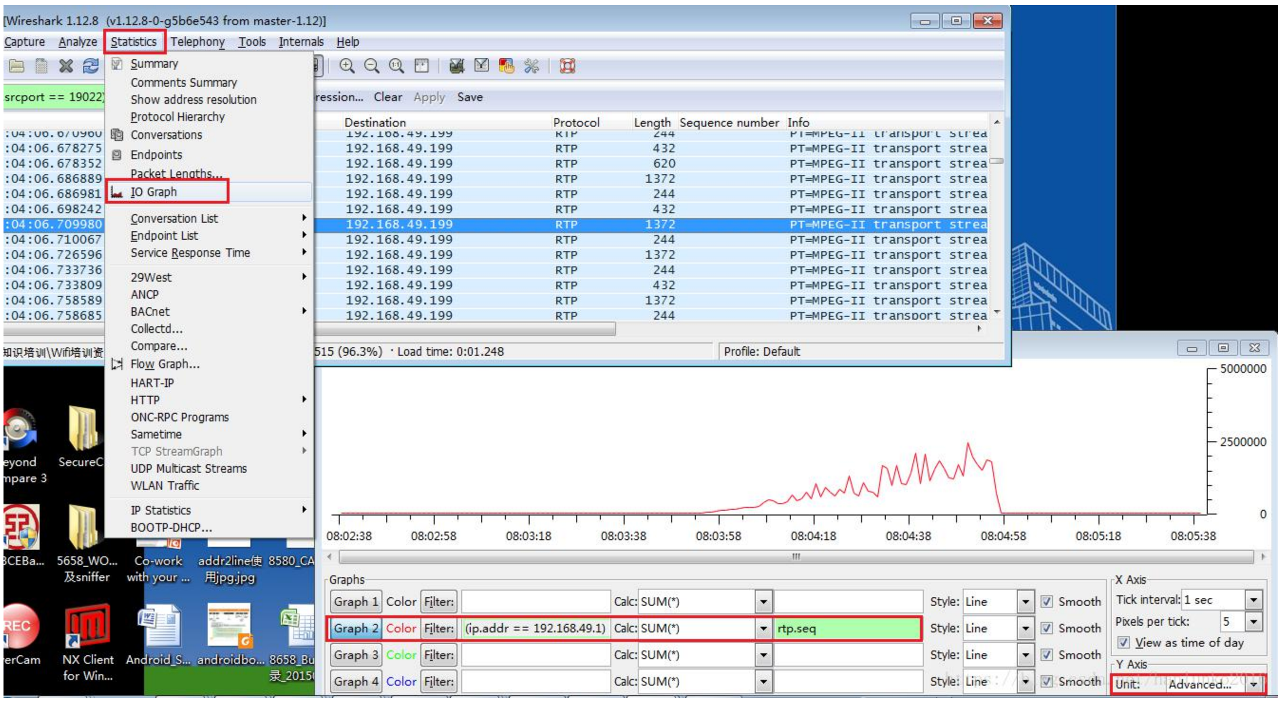
Analysing stream from 192.168.49.1 port 19022 to 192.168.49.199 port 15550 SSRC = 0x776683

Packet	Sequence	Delta(ms)	Filtered Jitter(m	Skew(ms)	IP BW(kbp	Marker	Status
169	0	0.00	0.00	0.00	4.83		[Ok]
170	1	4.01	0.19	-3.01	15.68		[Ok]
171	2	7.02	0.62	-10.03	26.53		[Ok]
172	3	0.11	0.58	-10.14	37.38		[Ok]
173	4	17.42	1.64	-27.56	48.22		[Ok]
174	5	0.12	1.54	-27.69	59.07		[Ok]
175	6	0.02	1.45	-27.71	69.92		[Ok]
176	7	0.02	1.36	-27.73	80.77		[Ok]

```
Max delta = 399.74 ms at packet no. 6217
Max jitter = 38.39 ms. Mean jitter = 4.83 ms.
Max skew = -709.09 ms.
Total RTP packets = 10587   (expected 10587)   Lost RTP packets = 72 (0.68%)   Sequence errors =
Duration 63.28 s (-173 ms clock drift, corresponding to 89754 Hz (-0.27%)
```

4）通过 IO Graph 分析数据包丢失和延迟

我们要使用函数功能时，须设置 Y Axis Unit 为 Advanced，过滤条件：(ip.addr == 192.168.49.1) and (ip.addr == 192.168.49.199)，函数：SUM(*)，统计字段：rtp.seq，从下图的波形看起来还是比较平滑，没有出现明显的峰值和下降。



```
Destination      Protocol  Length  Sequence number  Info
192.168.49.199   RTP       432                      PT=MPEG-II transport strea
192.168.49.199   RTP       620                      PT=MPEG-II transport strea
192.168.49.199   RTP       1372                     PT=MPEG-II transport strea
192.168.49.199   RTP       244                      PT=MPEG-II transport strea
192.168.49.199   RTP       432                      PT=MPEG-II transport strea
192.168.49.199   RTP       1372                     PT=MPEG-II transport strea
192.168.49.199   RTP       244                      PT=MPEG-II transport strea
192.168.49.199   RTP       1372                     PT=MPEG-II transport strea
192.168.49.199   RTP       244                      PT=MPEG-II transport strea
192.168.49.199   RTP       432                      PT=MPEG-II transport strea
192.168.49.199   RTP       1372                     PT=MPEG-II transport strea
192.168.49.199   RTP       244                      PT=MPEG-II transport strea
```

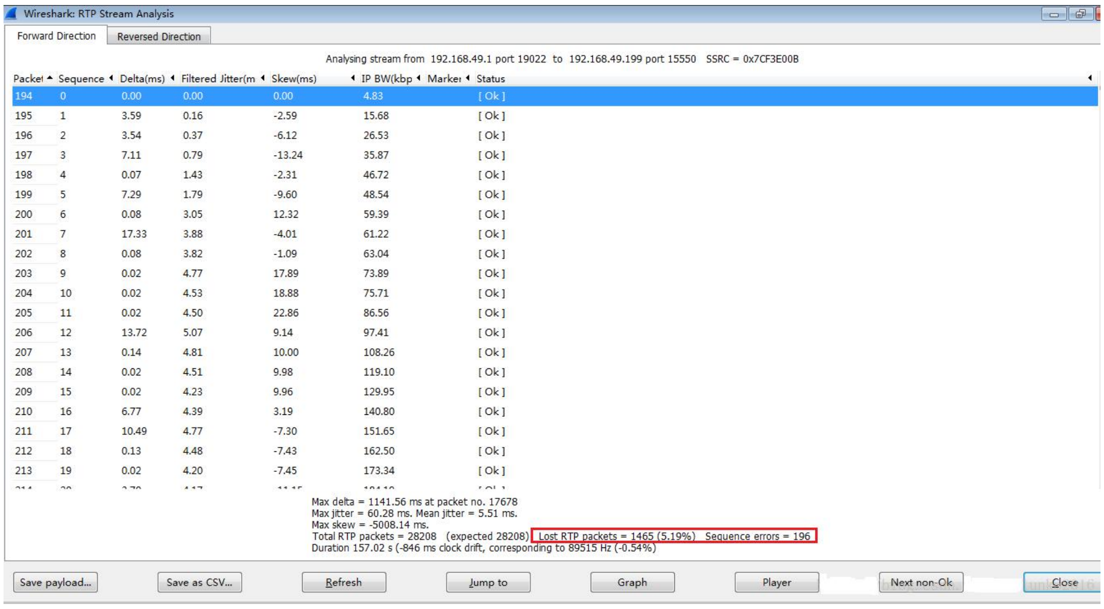
Graph 1	Color	Filter:		Calc: SUM(*)	Style: Line ✓ Smooth
Graph 2	Color	Filter: (ip.addr == 192.168.49.1)		Calc: SUM(*) rtp.seq	Style: Line ✓ Smooth
Graph 3	Color	Filter:		Calc: SUM(*)	Style: Line ✓ Smooth
Graph 4	Color	Filter:		Calc: SUM(*)	Style: Line ✓ Smooth

X Axis: Tick interval: 1 sec; Pixels per tick: 5; ✓ View as time of day
Y Axis: Unit: Advanced...

下面分析一个卡顿严重的 pcap 包：

步骤 1）和 2）和上面分析的正常的视频步骤相同，请参考上面的分析步骤。

3）对过滤后的 RTP 包进行分析：Stream Analysis


Wireshark: RTP Stream Analysis

Forward Direction | Reversed Direction

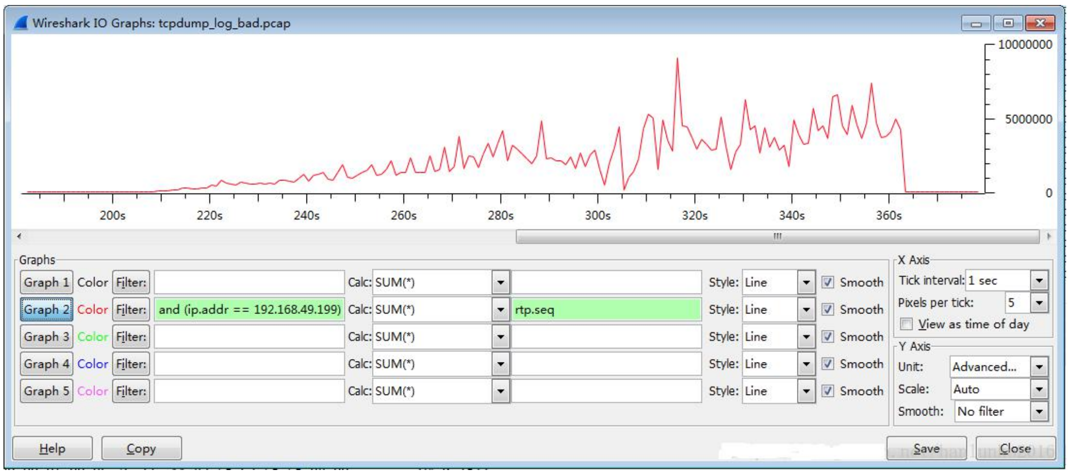
Analysing stream from 192.168.49.1 port 19022 to 192.168.49.199 port 15550 SSRC = 0x7CF3E00B

Packet	Sequence	Delta(ms)	Filtered Jitter(m	Skew(ms)	IP BW(kbp	Marker	Status
194	0	0.00	0.00	0.00	4.83		[Ok]
195	1	3.59	0.16	-2.59	15.68		[Ok]
196	2	3.54	0.37	-6.12	26.53		[Ok]
197	3	7.11	0.79	-13.24	35.87		[Ok]
198	4	0.07	1.43	-2.31	46.72		[Ok]
199	5	7.29	1.79	-9.60	48.54		[Ok]
200	6	0.08	3.05	12.32	59.39		[Ok]
201	7	17.33	3.88	-4.01	61.22		[Ok]
202	8	0.08	3.82	-1.09	63.04		[Ok]
203	9	0.02	4.77	17.89	73.89		[Ok]
204	10	0.02	4.53	18.88	75.71		[Ok]
205	11	0.02	4.50	22.86	86.56		[Ok]
206	12	13.72	5.07	9.14	97.41		[Ok]
207	13	0.14	4.81	10.00	108.26		[Ok]
208	14	0.02	4.51	9.98	119.10		[Ok]
209	15	0.02	4.23	9.96	129.95		[Ok]
210	16	6.77	4.39	3.19	140.80		[Ok]
211	17	10.49	4.77	-7.30	151.65		[Ok]
212	18	0.13	4.48	-7.43	162.50		[Ok]
213	19	0.02	4.20	-7.45	173.34		[Ok]

```
Max delta = 1141.56 ms at packet no. 17678
Max jitter = 60.28 ms. Mean jitter = 5.51 ms.
Max skew = -5008.14 ms.
Total RTP packets = 28208   (expected 28208)  Lost RTP packets = 1465 (5.19%)   Sequence errors = 196
Duration 157.02 s (-846 ms clock drift, corresponding to 89515 Hz (-0.54%)
```

4）通过 IO Graph 分析数据包丢失和延迟

我们要使用函数功能时，须设置 Y Axis Unit 为 Advanced，过滤条件：(ip.addr == 192.168.49.1) and (ip.addr == 192.168.49.199)，函数：SUM(*)，统计字段：rtp.seq，从下图看出现了若干明显的峰值和下降，说明丢包的情况比较严重。



Graphs

Graph 1 Color Filter: Calc: SUM(*) Style: Line ☑ Smooth
Graph 2 Color Filter: and (ip.addr == 192.168.49.199) Calc: SUM(*) rtp.seq Style: Line ☑ Smooth
Graph 3 Color Filter: Calc: SUM(*) Style: Line ☑ Smooth
Graph 4 Color Filter: Calc: SUM(*) Style: Line ☑ Smooth
Graph 5 Color Filter: Calc: SUM(*) Style: Line ☑ Smooth

X Axis
Tick interval: 1 sec
Pixels per tick: 5
☐ View as time of day

Y Axis
Unit: Advanced...
Scale: Auto
Smooth: No filter

2.2.3 Wifidisplay log 关键字

log 太多时，可以搜索以下关键字，从这句 log 开始 source 端已经打开 wifidsplay 功能：ActivityManager: START u0 {flg=0x8000 cmp=com.android.settings/.Settings$DisplaySettingsActivity (has extras)}

3 FAQ

3.1 Wifidisplay 无法投屏到 Win10 上

有些客户在开发过程中，发现 RK 平板无法投屏到 Windows10 系统上，可能是这两个原因导致，需要一个个去排查。

1. 客户的 Windows10 系统不支持 1280x720@p24 显示参数。
2. 客户的 Windows10 系统不支持 HDCP 加密。

```
diff --git a/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
b/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
index 0d2b7df..ded490d 100755
--- a/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
+++ b/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
//修改投屏分辨率
@ -78,11 +78,11 @ WifiDisplaySource::WifiDisplaySource(
mSupportedSourceVideoFormats.disableAll();
     mSupportedSourceVideoFormats.setNativeResolution(
-                VideoFormats::RESOLUTION_CEA, 15);    // 1280x720 p24
+                 VideoFormats::RESOLUTION_CEA, 5);    // 1280x720 p24
     // Enable all resolutions up to 1280x720p24
      mSupportedSourceVideoFormats.enableResolutionUpto(
-                VideoFormats::RESOLUTION_CEA, 15,
+                    VideoFormats::RESOLUTION_CEA, 5,
                     VideoFormats::PROFILE_CHP,    // Constrained High Profile
                     VideoFormats::LEVEL_32);       // Level 3.2
  }
//关掉 HDCP 加密
@ -591,7 +591,7 @ status_t WifiDisplaySource::sendM1(int32_t sessionID) {
      status_t WifiDisplaySource::sendM3(int32_t sessionID) {
       AString body =
-             "wfd_content_protection\r\n"
+              /*"wfd_content_protection\r\n"*/
              "wfd_video_formats\r\n"
              "wfd_audio_codecs\r\n"
```

3.2 投屏画面卡顿，经常出现花屏问题

经常有客户报投屏过程中出现卡顿和花屏现象，可以用 iperf 工具先测试下 Wifi 模组的 P2P 吞吐率和丢包率，测试方法参考 2.1 节，排查下 wifi 模组射频参数是否校准，天线匹配是否正常。也可以用 TCPDUMP 抓包，用 Wireshark 分析 pcap 包，debug 方法参考 2.2 节，确认 wifi 硬件是否正常，环境是否有干扰。假如没办法自己分析，可以把卡顿时的 pcap 包抓一份放 redmine 单号上，抓取命令：

```
tcpdump -i any -s 0 -w /data/tcpdump_log.pcap
```

排除 Wifi 硬件射频参数校准和环境干扰还会卡顿，可以通过以下三种方法来优化投屏卡顿：

1. 降低投屏分辨率和帧率，具体实现方法如 3.3 和 3.4 节。
2. Sink 端主动请求 I 帧，调整 Source 端 I 帧间隔时间，具体实现方法如 3.5 和 3.6 节。
3. 尽量使用 5G 频端 wifi，5G 带宽会比 2.4G 好。

3.3 Wifidisplay source 端设备设定默认分辨率

```
diff --git a/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
b/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
index 0d2b7df..ded490d 100755
--- a/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
+++ b/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
@ -78,11 +78,11 @ WifiDisplaySource::WifiDisplaySource(
mSupportedSourceVideoFormats.disableAll();
      mSupportedSourceVideoFormats.setNativeResolution(
-                   VideoFormats::RESOLUTION_CEA, 15);    // 1280x720 p24
+                    VideoFormats::RESOLUTION_CEA, 5);    // 1280x720 p24
      // Enable all resolutions up to 1280x720p24
```

```
        mSupportedSourceVideoFormats.enableResolutionUpto(
-                   VideoFormats::RESOLUTION_CEA, 15,
+                   VideoFormats::RESOLUTION_CEA, 5,
                    VideoFormats::PROFILE_CHP,    // Constrained High Profile
                    VideoFormats::LEVEL_32);      // Level 3.2
 }
```

Source 端所有支持的分辨率数组如下：

```
frameworks/av/media/libstagefright/wifi-display/VideoFormats.cpp
const VideoFormats::config_t VideoFormats::mResolutionTable[][32] = {
    {
        // CEA Resolutions
        { 640, 480, 60, false, 0, 0},
        { 720, 480, 60, false, 0, 0},
        { 720, 480, 60, true, 0, 0},
        { 720, 576, 50, false, 0, 0},
        { 720, 576, 50, true, 0, 0},
        { 1280, 720, 30, false, 0, 0},
        { 1280, 720, 60, false, 0, 0},
        { 1920, 1080, 30, false, 0, 0},
        { 1920, 1080, 60, false, 0, 0},
        { 1920, 1080, 60, true, 0, 0},
        { 1280, 720, 25, false, 0, 0},
        { 1280, 720, 50, false, 0, 0},
        { 1920, 1080, 25, false, 0, 0},
        { 1920, 1080, 50, false, 0, 0},
        { 1920, 1080, 50, true, 0, 0},
        { 1280, 720, 24, false, 0, 0},
        { 1920, 1080, 24, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
    },
    {
        // VESA Resolutions
        { 800, 600, 30, false, 0, 0},
        { 800, 600, 60, false, 0, 0},
        { 1024, 768, 30, false, 0, 0},
        { 1024, 768, 60, false, 0, 0},
        { 1152, 864, 30, false, 0, 0},
        { 1152, 864, 60, false, 0, 0},
        { 1280, 768, 30, false, 0, 0},
        { 1280, 768, 60, false, 0, 0},
        { 1280, 800, 30, false, 0, 0},
        { 1280, 800, 60, false, 0, 0},
        { 1360, 768, 30, false, 0, 0},
        { 1360, 768, 60, false, 0, 0},
        { 1366, 768, 30, false, 0, 0},
        { 1366, 768, 60, false, 0, 0},
        { 1280, 1024, 30, false, 0, 0},
        { 1280, 1024, 60, false, 0, 0},
        { 1400, 1050, 30, false, 0, 0},
        { 1400, 1050, 60, false, 0, 0},
        { 1440, 900, 30, false, 0, 0},
        { 1440, 900, 60, false, 0, 0},
        { 1600, 900, 30, false, 0, 0},
        { 1600, 900, 60, false, 0, 0},
        { 1600, 1200, 30, false, 0, 0},
        { 1600, 1200, 60, false, 0, 0},
        { 1680, 1024, 30, false, 0, 0},
        { 1680, 1024, 60, false, 0, 0},
        { 1680, 1050, 30, false, 0, 0},
        { 1680, 1050, 60, false, 0, 0},
        { 1920, 1200, 30, false, 0, 0},
        { 1920, 1200, 60, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
    },
    {
        // HH Resolutions
        { 800, 480, 30, false, 0, 0},
        { 800, 480, 60, false, 0, 0},
        { 854, 480, 30, false, 0, 0},
        { 854, 480, 60, false, 0, 0},
        { 864, 480, 30, false, 0, 0},
        { 864, 480, 60, false, 0, 0},
        { 640, 360, 30, false, 0, 0},
        { 640, 360, 60, false, 0, 0},
        { 960, 540, 30, false, 0, 0},
        { 960, 540, 60, false, 0, 0},
        { 848, 480, 30, false, 0, 0},
        { 848, 480, 60, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
        { 0, 0, 0, false, 0, 0},
    }
};
```

3.4 Wifidisplay sink 端设定支持的分辨率

```
frameworks/av/media/libstagefright/wifi-display/sink/WifiDisplaySink.cpp

char parameterByte[3]; // for save the transfer byte modify by lance 2013.07.31
memset(parameterByte, 0, 3);
resolutionToParameterByte(parameterByte, displayWidth, displayHeight, info.fps);
AString wfdVideoFormatsString(parameterByte);
wfdVideoFormatsString.append(" 00 02 08 00008423 00000055 00000555 00 0000 0000 00 none
none,01 08 00008423 00000055 00000555 00 0000 0000 00 none none");
```

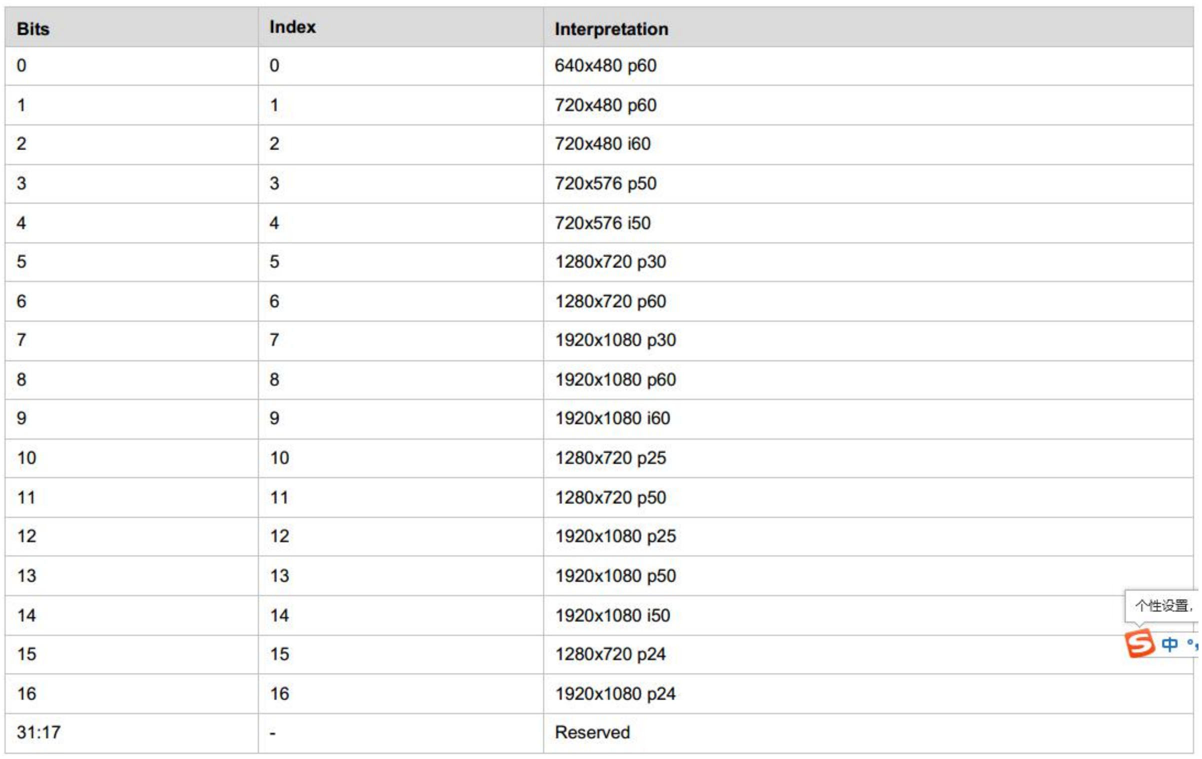
Android8.1 以前 wifi-display 的 sink 源码是在 framework/av/media/libstagefright 目录下面，从 Android8.1 开始，我们把 wifi-display 的 sink 代码移到 sink apk 里面了。从上面代码看出 Wifidisplay sink 支持的分辨率设置主要是 00008423 00000055 00000555。用 2 进制，从右到左，第 0bit 最左边。需要的就对应位置 1，不需要就置 0，然后转换成 16 进制。

00008423 是支持的 CEA 组分辨率。

00000055 是支持的 VESA 组分辨率。

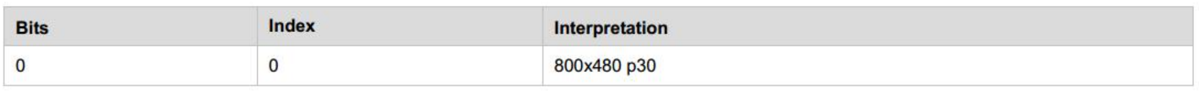
00000555 是支持的 HH 组分辨率。

Table 34. Supported CEA Resolution/Refresh Rates



Bits	Index	Interpretation
0	0	640x480 p60
1	1	720x480 p60
2	2	720x480 i60
3	3	720x576 p50
4	4	720x576 i50
5	5	1280x720 p30
6	6	1280x720 p60
7	7	1920x1080 p30
8	8	1920x1080 p60
9	9	1920x1080 i60
10	10	1280x720 p25
11	11	1280x720 p50
12	12	1920x1080 p25
13	13	1920x1080 p50
14	14	1920x1080 i50
15	15	1280x720 p24
16	16	1920x1080 p24
31:17	-	Reserved

Table 36. Supported HH Resolutions/Refresh Rates



Bits	Index	Interpretation
0	0	800x480 p30

Bits	Index	Interpretation
1	1	800x480 p60
2	2	854x480 p30
3	3	854x480 p60
4	4	864x480 p30
5	5	864x480 p60
6	6	640x360 p30
7	7	640x360 p60
8	8	960x540 p30
9	9	960x540 p60
10	10	848x480 p30
11	11	848x480 p60
31:12	-	Reserved

Table 35. Supported VESA Resolution/Refresh Rates

Bits	Index	Interpretation
0	0	800x600 p30
1	1	800x600 p60

Bits	Index	Interpretation
2	2	1024x768 p30
3	3	1024x768 p60
4	4	1152x864 p30
5	5	1152x864 p60
6	6	1280x768 p30
7	7	1280x768 p60
8	8	1280x800 p30
9	9	1280x800 p60
10	10	1360x768 p30
11	11	1360x768 p60
12	12	1366x768 p30
13	13	1366x768 p60
14	14	1280x1024 p30
15	15	1280x1024 p60
16	16	1400x1050 p30
17	17	1400x1050 p60
18	18	1440x900 p30
19	19	1440x900 p60
20	20	1600x900 p30
21	21	1600x900 p60
22	22	1600x1200 p30
23	23	1600x1200 p60
24	24	1680x1024 p30
25	25	1680x1024 p60
26	26	1680x1050 p30
27	27	1680x1050 p60
28	28	1920x1200 p30
31:29	-	Reserved

3.5 Wifidisplay Sink 端主动请求 I 帧

Android 平台 WifiDisplaySink.cpp 文件里面已经封装了 sendIDR 接口，sendIDR 是发送请求 I 帧的接口，当 wifidisplay sink 设备端出现画面卡顿时（码流数据丢包），可以发送请求 I 帧，快速恢复。sendIDR 接口实现如下：

```
void WifiDisplaySink::sendIDR(int32_t sessionID, const char *uri)
{
    AString request = AStringPrintf("SET_PARAMETER %s RTSP/1.0\r\n", uri);
    AppendCommonResponse(&request, mNextCSeq);
    request.append("Content-Type: text/parameters\r\n");
    request.append("Content-Length: 17\r\n");
    request.append("\r\n");
    request.append("wfd_idr_request\r\n");
    status_t err =
    mNetSession->sendRequest(sessionID, request.c_str(), request.size());

    ALOGI("%s\n",request.c_str());
    registerResponseHandler(
            sessionID, mNextCSeq, &WifiDisplaySink::onReceiveIdrResponse);

    if (err != OK) {
        return;
    }
    ++mNextCSeq;
}
```

Wifidisplay sink 端发送请求 I 帧补丁如下：

```
wangjianhui@ubuntu:~/3_rk3399_Android7.1_all/packages/apps/WifiDisplay$ git diff .
diff --git a/jni/TunnelRenderer.cpp b/jni/TunnelRenderer.cpp
index c918985..321b483 100755
--- a/jni/TunnelRenderer.cpp
+++ b/jni/TunnelRenderer.cpp
@@ -908,6 +908,7 @@ sp<ABuffer> TunnelRenderer::dequeueBuffer() {
         }
  #endif


+        property_set("persist.sys.wfd.droppingpacket", "1");
         ALOGI("dropping packet. extSeqNo %d didn't arrive in time but newSeqNo %d",
                 mLastDequeuedExtSeqNo + 1, extSeqNo);

```

```
diff --git a/jni/WifiDisplaySink.cpp b/jni/WifiDisplaySink.cpp
index 109b72e..f06a9dd 100755
--- a/jni/WifiDisplaySink.cpp
+++ b/jni/WifiDisplaySink.cpp
@@ -158,6 +158,8 @@ void *WifiDisplaySink::ThreadWrapper(void *)
                        fds[0].fd = -1;
                        goto wfd_sink_thread_end;
            }
+
+           property_set("persist.sys.wfd.droppingpacket", 0);
            while(!end_flag)
            {
                        int ret;
@@ -220,6 +222,11 @@ void *WifiDisplaySink::ThreadWrapper(void *)
                                    }

                        }
+
+                       if (property_get_int32("persist.sys.wfd.droppingpacket", 0) == 1) {
+                                   property_set("persist.sys.wfd.droppingpacket", 0);
+                                   sendIDR(mSessionID,mUrl->c_str());
+                       }
            }
 wfd_sink_thread_end:
           ALOGD("end of threadloop end_flag %d errno %d",end_flag,errno);
```

3.6 Wifidisplay Source 端设置 I 帧间隔时间

```
diff --git a/media/libstagefright/wifi-display/source/Converter.cpp
b/media/libstagefright/wifi-display/source/Converter.cpp
index 379d769..ed7a42c 100755
--- a/media/libstagefright/wifi-display/source/Converter.cpp
+++ b/media/libstagefright/wifi-display/source/Converter.cpp
@@ -171,8 +171,8 @@ status_t Converter::initEncoder() {
} else {
     mOutputFormat->setInt32("bitrate", videoBitrate);
     mOutputFormat->setInt32("bitrate-mode",OMX_Video_ControlRateConstant);
     mOutputFormat->setInt32("frame-rate", 30);
- mOutputFormat->setInt32("i-frame-interval", 15); // Iframes every 15 secs
```

```
+ mOutputFormat->setInt32("i-frame-interval", 3); // Iframes every 3 secs

// Configure encoder to use intra macroblock refresh mode
mOutputFormat->setInt32("intra-refresh-mode", OMX_VIDEO_IntraRefreshCyclic);
```

默认是 15s 发送一个 I 帧，修改后默认 3s 发送一个 I 帧。发送 I 帧间隔时间不能太小，否则会出现视频码流包太大，消耗网络带宽。

3.7 Wifidisplay 去除 HDCP 加密补丁

在实际使用过程中，有些客户使用的第三方接收端设备或者发送端设备不带 HDCP 加密功能，需要把 RK 的设备去除 HDCP 加密功能，才能实现投屏功能。可以使用下面补丁：

RK 设备做 source 功能

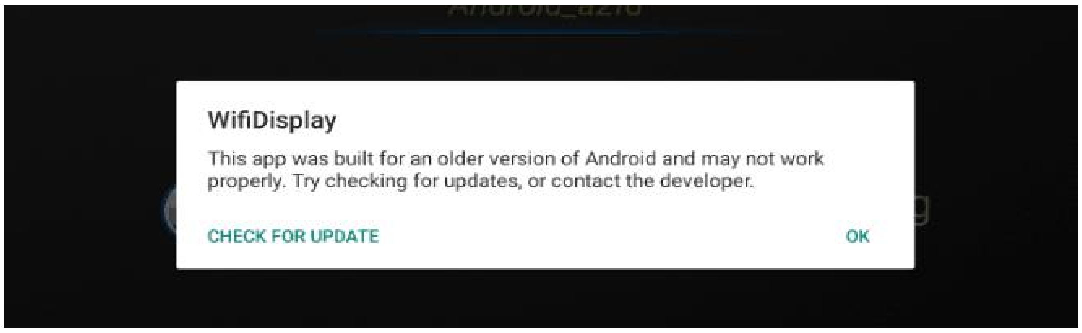
```
--- a/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
+++ b/media/libstagefright/wifi-display/source/WifiDisplaySource.cpp
@@ -591,7 +591,7 @@ status_t WifiDisplaySource::sendM1(int32_t sessionID) {

 status_t WifiDisplaySource::sendM3(int32_t sessionID) {
     AString body =
- "wfd_content_protection\r\n"
+ //"wfd_content_protection\r\n"
         "wfd_video_formats\r\n"
         "wfd_audio_codecs\r\n"
         "wfd_client_rtp_ports\r\n";
```

RK 设备做 sink 功能

```
--- a/media/libstagefright/wifi-display/sink/WifiDisplaySink.cpp
+++ b/media/libstagefright/wifi-display/sink/WifiDisplaySink.cpp
@@ -1038,6 +1038,7 @@ void WifiDisplaySink::onGetParameterRequest(
               }
 #else
         if (strstr(request_param, "wfd_content_protection")) {
+#if 0
 #ifdef WFD_HDCP_SUPPORT
              mUsingHDCP = true;
 #endif
@@ -1045,6 +1046,8 @@ void WifiDisplaySink::onGetParameterRequest(
               body.append(AStringPrintf("wfd_content_protection: HDCP2.1
port=%d\r\n",kHDCPDefaultPort));
              else
               body.append("wfd_content_protection: none\r\n");
+#endif
+ body.append("wfd_content_protection: none\r\n");
          }
 #endif
```

3.8 Android9.0 系统加 wifidisplay sink 功能弹提示框错误

Android9.0 系统，带上 wifidisplay sink 功能后，有弹如下图错误。



WifiDisplay

This app was built for an older version of Android and may not work properly. Try checking for updates, or contact the developer.

解决方案：

```
framewk/base/services/core/java/com/android/server/am/AppWarnings.java

onStartActivity(ActivityRecord r){
       showUnsupportedCompileSdkDialogIfNeeded(r);
       showUnsupportedDisplaySizeDialogIfNeeded(r);
       showDeprecatedTargetDialogIfNeeded(r);//这行代码加个过滤条件，即可解决
}
public void showDeprecatedTargetDialogIfNeeded(ActivityRecord r) {
       if (r.appInfo.targetSdkVersion < Build.VERSION.MIN_SUPPORTED_TARGET_SDK_INT) {
           mUiHandler.showDeprecatedTargetDialog(r);
       }
}
```

3.9 Wifidisplay sink 延时优化

客户使用过程中由于 wifi 干扰导致投屏延时严重问题，可以通过丢弃一些过时的数据包来优化延时效果。以 Android8.1 为例补丁如下（可以通过提 redmine 获取补丁）：

```
From 680dcc219fe6dc7a2efe63b43608b0f10ef27b99 Mon Sep 17 00:00:00 2001
From: WangJianhui <wjh@rock-chips.com>
Date: Thu, 25 Oct 2018 15:05:21 +0800
Subject: [PATCH] nuplayer: optimize wfd sink display rate.
```

```
Signed-off-by: WangJianhui <wjh@rock-chips.com>
---
 media/libmediaplayerservice/nuplayer/NuPlayer.cpp    |    4 +
 .../nuplayer/NuPlayerDecoder.cpp                      |    6 +
 .../nuplayer/NuPlayerRenderer.cpp                     | 130 ++++++++++++++++++++
 .../nuplayer/NuPlayerRenderer.h                       |   13 +++
 .../nuplayer/NuPlayerSource.h                         |    4 +
 .../nuplayer/StreamingSource.cpp                      |   23 +++-
 .../nuplayer/StreamingSource.h                        |    8 ++
 media/libstagefright/mpeg2ts/ATSParser.h              |    2 +
 8 files changed, 188 insertions(+), 2 deletions(-)

diff --git a/media/libmediaplayerservice/nuplayer/NuPlayer.cpp
b/media/libmediaplayerservice/nuplayer/NuPlayer.cpp
index df36046..2bf7cd3 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayer.cpp
+++ b/media/libmediaplayerservice/nuplayer/NuPlayer.cpp
@@ -1488,6 +1488,10 @@ void NuPlayer::onStart(int64_t startPositionUs, MediaPlayerSeekMode
mode) {
             flags |= Renderer::FLAG_REAL_TIME;
         }

+        if (mSource->isWFDStreaming()) {
+              flags |= Renderer::FLAG_WFD_STREAMING;
+        }
+
         bool hasAudio = (mSource->getFormat(true /* audio */) != NULL);
         bool hasVideo = (mSource->getFormat(false /* audio */) != NULL);
         if (!hasAudio && !hasVideo) {
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerDecoder.cpp
b/media/libmediaplayerservice/nuplayer/NuPlayerDecoder.cpp
index ac187cc..9bc1fb6 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerDecoder.cpp
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerDecoder.cpp
@@ -769,6 +769,12 @@ bool NuPlayer::Decoder::handleAnOutputBuffer(

         if (mRenderer != NULL) {
             // send the buffer to renderer.
+            if (mSource->isWFDStreaming()) {
+                  int64_t sys_start_time = mSource->getWFDStartSysTimeUs();
+                    int64_t audio_start_mediaTimeUs = mSource->getWFDStartMediaTimeUs();
+                    if ((sys_start_time != -1) && (audio_start_mediaTimeUs != -1))
+                         mRenderer->setWFDTimeUs(sys_start_time, audio_start_mediaTimeUs);
+               }
               mRenderer->queueBuffer(mIsAudio, buffer, reply);
               if (eos && !isDiscontinuityPending()) {
                    mRenderer->queueEOS(mIsAudio, ERROR_END_OF_STREAM);
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
index bd866cb..d63f89b 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
@@ -34,6 +34,8 @@

 #include <inttypes.h>

+#define DIRECT_RENDER_NO_AVSYNC
+
 namespace android {

 /*
@@ -86,6 +88,7 @@ const NuPlayer::Renderer::PcmInfo
NuPlayer::Renderer::AUDIO_PCMINFO_INITIALIZER

 // static
 const int64_t NuPlayer::Renderer::kMinPositionUpdateDelayUs = 100000ll;
+FILE *omx_rs_txt;

 NuPlayer::Renderer::Renderer(
               const sp<MediaPlayerBase::AudioSink> &sink,
@@ -104,6 +107,11 @@ NuPlayer::Renderer::Renderer(
           mVideoDrainGeneration(0),
           mAudioEOSGeneration(0),
           mPlaybackSettings(AUDIO_PLAYBACK_RATE_DEFAULT),
+          sys_start_time(-1),
+          audio_start_timeUs(-1),
+          last_adujst_time(-1),
+          last_timeUs(-1),
+          last_cont_timeUs(-1),
          mAudioFirstAnchorTimeMediaUs(-1),
            mAnchorTimeMediaUs(-1),
            mAnchorNumFramesWritten(-1),
@@ -1044,6 +1052,121 @@ bool NuPlayer::Renderer::onDrainAudioQueue() {
                    CHECK(entry->mBuffer->meta()->findInt64("timeUs", &mediaTimeUs));
                    ALOGV("onDrainAudioQueue: rendering audio at media time %.2f secs",
                                mediaTimeUs / 1E6);
+                    if (mFlags & FLAG_WFD_STREAMING) {
+                         int64_t sys_time = systemTime(SYSTEM_TIME_MONOTONIC) / 1000;
+                         CHECK_EQ(mAudioSink->getPosition(&numFramesPlayed), (status_t)OK);
+                         uint32_t numFramesPendingPlayout = mNumFramesWritten - numFramesPlayed;
+                         if(sys_start_time == 0)
+                           {
+                                        sys_start_time =sys_time;
+                                        ALOGD("sys_start_time == 0 %lld",(long long)sys_start_time);
+                           }
+                           if(audio_start_timeUs == 0)
+                           {
+                                        audio_start_timeUs = mediaTimeUs;
+                                        ALOGD("audio_start_timeUs == 0 %lld",(long
long)audio_start_timeUs);
+                           }
+                           if(last_adujst_time == 0)
+                                        last_adujst_time = sys_time;
+
+                           if (last_cont_timeUs > mediaTimeUs) {
+                                      ALOGD("###mediatimeus has been reset, %lld %lld", (long
long)last_cont_timeUs, (long long)mediaTimeUs);
+                                      last_timeUs = 0;
+                           }
+                           last_cont_timeUs = mediaTimeUs;
+
+                           int64_t pending_time =    numFramesPendingPlayout *
mAudioSink->msecsPerFrame() * 1000ll;
+
+                           if(sys_start_time + (mediaTimeUs - audio_start_timeUs)    - pending_time <
sys_time - 100000ll )
+                           {
+                                        if(last_timeUs <= mediaTimeUs )//loop tntil the real mediaTimeUs catch
up with the old setted one    , if there is no data,the old setted is also faster than the real mediaTimeUs.so
it's okay
```

```
+                                    {
+                                              if(sys_start_time + (mediaTimeUs - audio_start_timeUs) -
pending_time< sys_time - 300000ll || (sys_time - last_adujst_time > 20000000ll && sys_start_time +
(mediaTimeUs - audio_start_timeUs) - pending_time< sys_time - 100000ll))//recalcu late the
mediaTimeUs.
+                                              {
+                                                         int retrtptxt;
+                                                         if((retrtptxt = access("data/test/omx_rs_txt_file2",0))
== 0)
+                                                         {
+                                                                    if(omx_rs_txt == NULL)
+                                                                              omx_rs_txt =
fopen("data/test/omx_rs_txt2.txt","a");
+                                                                    if(omx_rs_txt != NULL)
+
+                                                                    {
+                                                                              if(sys_time - last_adujst_time >
20000000ll && sys_start_time + (mediaTimeUs- audio_start_timeUs) - pending_time < sys_time -
100000ll)
+
fprintf(omx_rs_txt,"NuPlayer::Renderer::onDrainAudioQueue adjust start %lld %lld sys %lld %lld
mediaTimeUs %lld last %lld delta %lld    %lld %lld\n",(long long)sys_start_time,(long
long)audio_start_timeUs,(long long)last_adujst_time,(long long)sys_time, (long
long)mediaTimeUs,(long long)last_timeUs,(long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs) + pending_time), (long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs)),(long long)(sys_time-last_adujst_time));
+                                                                              else
+
fprintf(omx_rs_txt,"NuPlayer::Renderer::onDrainAudioQueue before delay
300msstart             %lld    %lld sys %lld %lld mediaTimeUs %lld last %lld
delta %lld    %lld %lld\n",(long long)sys_start_time,(long long)audio_start_timeUs,(long
long)last_adujst_time,(long long)sys_time,(long long)mediaTimeUs,(long long)last_timeUs,(long
long)(sys_time-sys_start_time-(mediaTimeUs - audio_start_timeUs) + pending_time),(long
long)(sys_time-sys_start_time-(mediaTimeUs - audio_start_timeUs)),(long
long)(sys_time-last_adujst_time));
+                                                                              fflush(omx_rs_txt);
+
+                                                                    }
+                                                         }
+
ALOGD("catchup:%lld:%lld:%lld:%lld:%lld:%lld:%zu",(long long)sys_start_time,(long
long)audio_start_timeUs,(long long)mediaTimeUs,(long long)sys_time,(long long)(((sys_time -
sys_start_time - (mediaTimeUs - audio_start_timeUs))/11)*11),(long long)pending_time,
mAudioQueue.size());
+                                                    mediaTimeUs +=((sys_time - sys_start_time -
(mediaTimeUs - audio_start_timeUs) ) / 11) *11;
+                                                   last_adujst_time = sys_time;
+                                          }
+                                          else
+                                          {
+                                                   int retrtptxt;
+                                                   if((retrtptxt = access("data/test/omx_rs_txt_file2",0))
== 0)
+                                                   {
+                                                             if(omx_rs_txt == NULL)
+                                                                       omx_rs_txt =
fopen("data/test/omx_rs_txt2.txt","a");
+                                                             if(omx_rs_txt != NULL)
+                                                             {
+
fprintf(omx_rs_txt,"NuPlayer::Renderer::onDrainAudioQueue before dec delay 100-300 ms
start      %lld           %lld sys %lld %lld mediaTimeUs %lld last %lld delta %lld    %lld %lld\n",(long
long)sys_start_time,(long long)audio_start_timeUs,(long long)last_adujst_time,(long long)sys_time,(long
long)mediaTimeUs,(long long)last_timeUs,(long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs) + pending_time), (long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs)),(long long)(sys_time-last_adujst_time));
+                                                                       fflush(omx_rs_txt);
+
+                                                             }
+                                                   }
+                                          }
+                                 }
+                                 else
+                                 {
+                                                   int retrtptxt;
+                                                   if((retrtptxt = access("data/test/omx_rs_txt_file2",0))
== 0)
+                                                   {
+                                                             if(omx_rs_txt == NULL)
+                                                                       omx_rs_txt =
fopen("data/test/omx_rs_txt2.txt","a");
+                                                       if(omx_rs_txt != NULL)
+
+                                                       {
+
fprintf(omx_rs_txt,"NuPlayer::Renderer::onDrainAudioQueue before dec delay 100-300 ms
start     %lld          %lld sys %lld %lld mediaTimeUs %lld last %lld delta %lld    %lld %lld\n" ,(long
long)sys_start_time,(long long)audio_start_timeUs,(long long)last_adujst_time,(long long)sys_time,(long
long)mediaTimeUs,(long long)last_timeUs,(long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs) + pending_time), (long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs)),(long long)(sys_time-last_adujst_time));
+                                                               fflush(omx_rs_txt);
+
+                                                       }
+                                               }
+                                               {
+                                                       ALOGV("discarding the late
mediatimeUs %lld:%lld:%lld:%lld",(long long)mediaTimeUs,(long long)sys_start_time,(long
long)audio_start_timeUs, (long long)last_timeUs);
+                                                       entry->mNotifyConsumed->post();
+                                                       mAudioQueue.erase(mAudioQueue.begin());
+                                                       entry = NULL;
+                                               }
+                                               continue;
+                               }
+                       }
+                       else
+                       {
+                               int retrtptxt;
+                               if((retrtptxt = access("data/test/omx_rs_txt_file2",0)) == 0)
+                               {
+                                       if(omx_rs_txt == NULL)
+                                               omx_rs_txt = fopen("data/test/omx_rs_txt2.txt","a");
+                                       if(omx_rs_txt != NULL)
+
+                                       {
+
fprintf(omx_rs_txt,"NuPlayer::Renderer::onDrainAudioQueue before less than 100ms
start     %lld    %lld sys %lld %lld mediaTimeUs %lld last %lld delta %lld      %lld %lld\n",(long
long)sys_start_time,(long long)audio_start_timeUs,(long long)last_adujst_time,(long long)sys_time,(long
long)mediaTimeUs,(long long)last_timeUs ,(long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs) + pending_time),(long long)(sys_time-sys_start_time-(mediaTimeUs -
audio_start_timeUs)),(long long)(sys_time-last_adujst_time));
+                                                       fflush(omx_rs_txt);
+                                               }
+                                       }
+                               }
+
+                               if (last_timeUs > mediaTimeUs) {
+                                       ALOGD("error,last_timeUs=%lld, mediaTimeUs=%lld", (long
long)last_timeUs, (long long)mediaTimeUs);
+                               }
+
+                               last_timeUs = mediaTimeUs;
+                               if(sys_time - last_adujst_time > 20000000ll)
+                                       last_adujst_time = sys_time;
+                       }
                        onNewAudioMediaTime(mediaTimeUs);
                }

@@ -1285,6 +1408,13 @@ void NuPlayer::Renderer::postDrainVideoQueue() {
                // discontinuity. If we have not drained an audio buffer that was
                // received after this buffer, repost in 10 msec. Otherwise repost
                // in 500 msec.
+               if(mFlags & FLAG_WFD_STREAMING) {
+#ifdef DIRECT_RENDER_NO_AVSYNC
+                       msg->post();
+                       mDrainVideoQueuePending = true;
+                       return;
+#endif
+               }
                delayUs = realTimeUs - nowUs;
                int64_t postDelayUs = -1;
                if (delayUs > 500000) {
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
index f58b79c..ffc3b56 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
@@ -34,6 +34,7 @@ struct NuPlayer::Renderer : public AHandler {
        enum Flags {
            FLAG_REAL_TIME = 1,
            FLAG_OFFLOAD_AUDIO = 2,
+           FLAG_WFD_STREAMING = 8,
        };
        Renderer(const sp<MediaPlayerBase::AudioSink> &sink,
                 const sp<AMessage> &notify,
@@ -89,6 +90,11 @@ struct NuPlayer::Renderer : public AHandler {
                bool isStreaming,
                const sp<AMessage> &notify);

+       void setWFDTimeUs(int64_t sysTimeUs, int64_t mediaTimeUs) {
+           sys_start_time = sysTimeUs;
+           audio_start_timeUs = mediaTimeUs;
+       }
+
        enum {
            kWhatEOS                       = 'eos ',
            kWhatFlushComplete             = 'fluC',
@@ -172,6 +178,13 @@ private:
        AVSyncSettings mSyncSettings;
        float mVideoFpsHint;

+       //wifi display instance
+       int64_t sys_start_time;
+       int64_t audio_start_timeUs;
+       int64_t last_adujst_time;
+       int64_t last_timeUs;
+       int64_t last_cont_timeUs;
+
        int64_t mAudioFirstAnchorTimeMediaUs;
        int64_t mAnchorTimeMediaUs;
        int64_t mAnchorNumFramesWritten;
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerSource.h
b/media/libmediaplayerservice/nuplayer/NuPlayerSource.h
index 8ba9c0d..8a992a2 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerSource.h
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerSource.h
@@ -146,6 +146,10 @@ struct NuPlayer::Source : public AHandler {
            return INVALID_OPERATION;
        }

+       virtual bool isWFDStreaming() {return false;}
+       virtual int64_t getWFDStartSysTimeUs() {return -1;}
+       virtual int64_t getWFDStartMediaTimeUs() {return -1;}
+
 protected:
        virtual ~Source() {}

diff --git a/media/libmediaplayerservice/nuplayer/StreamingSource.cpp
b/media/libmediaplayerservice/nuplayer/StreamingSource.cpp
index fc0803b..04acdd0 100644
--- a/media/libmediaplayerservice/nuplayer/StreamingSource.cpp
+++ b/media/libmediaplayerservice/nuplayer/StreamingSource.cpp
@@ -33,7 +33,7 @@

 namespace android {

-const int32_t kNumListenerQueuePackets = 80;
+const int32_t kNumListenerQueuePackets = 200;

 NuPlayer::StreamingSource::StreamingSource(
                const sp<AMessage> &notify,
@@ -41,6 +41,9 @@ NuPlayer::StreamingSource::StreamingSource(
        : Source(notify),
          mSource(source),
          mFinalResult(OK),
+         mWFDFlag(false),
+         mWFDStartSysTimeUs(-1),
+         mWFDStartMediaTimeUs(-1),
          mBuffering(false) {
 }

@@ -91,6 +94,12 @@ void NuPlayer::StreamingSource::start() {
                parserFlags |= ATSParser::ALIGNED_VIDEO_DATA;
        }

+       if ((sourceFlags >> 16 & 0xFFFF) == 0x1234) {
+               mWFDFlag = true;
+               parserFlags |= ATSParser::WIFI_DISPLAY;
+               ALOGD("NuPlayer::StreamingSource::start sourceFlags %x",sourceFlags);
+       }
+
        mTSParser = new ATSParser(parserFlags);

        mStreamListener->start();
@@ -129,8 +138,18 @@ void NuPlayer::StreamingSource::onReadBuffer() {
                        type = mask;
                    }

-                   mTSParser->signalDiscontinuity(
+                   if (mWFDFlag)
+                   {
+                           int64_t sys_timeUs;
+                           int64_t mediaTimeUs;
+                           if (extra->findInt64("wifidisplay_sys_timeUs", &sys_timeUs) &&
extra->findInt64("timeUs", &mediaTimeUs)) {
+                                   mWFDStartSysTimeUs = sys_timeUs;
+                                   mWFDStartMediaTimeUs = mediaTimeUs;
+                           }
+                   } else {
+                           mTSParser->signalDiscontinuity(
                                    (ATSParser::DiscontinuityType)type, extra);
+                   }
                } else if (n < 0) {
                    break;
                } else {
diff --git a/media/libmediaplayerservice/nuplayer/StreamingSource.h
b/media/libmediaplayerservice/nuplayer/StreamingSource.h
index 2e1d2b3..f993580 100644
--- a/media/libmediaplayerservice/nuplayer/StreamingSource.h
+++ b/media/libmediaplayerservice/nuplayer/StreamingSource.h
@@ -45,6 +45,10 @@ struct NuPlayer::StreamingSource : public NuPlayer::Source {

        virtual bool isRealTime() const;

+       virtual bool isWFDStreaming() {return mWFDFlag != 0;}
+       virtual int64_t getWFDStartSysTimeUs() {return mWFDStartSysTimeUs;}
+       virtual int64_t getWFDStartMediaTimeUs() {return mWFDStartMediaTimeUs;}
+
 protected:
        virtual ~StreamingSource();

@@ -61,6 +65,10 @@ private:
        sp<NuPlayerStreamListener> mStreamListener;
        sp<ATSParser> mTSParser;

+       bool mWFDFlag;
+       int64_t mWFDStartSysTimeUs;
+       int64_t mWFDStartMediaTimeUs;
+
        bool mBuffering;
        Mutex mBufferingLock;
        sp<ALooper> mLooper;
diff --git a/media/libstagefright/mpeg2ts/ATSParser.h b/media/libstagefright/mpeg2ts/ATSParser.h
index 41c19cd..908b295 100644
--- a/media/libstagefright/mpeg2ts/ATSParser.h
+++ b/media/libstagefright/mpeg2ts/ATSParser.h
@@ -68,6 +68,8 @@ struct ATSParser : public RefBase {
                TS_TIMESTAMPS_ARE_ABSOLUTE = 1,
                // Video PES packets contain exactly one (aligned) access unit.
                ALIGNED_VIDEO_DATA          = 2,
+               // WiFi Display Flag
+               WIFI_DISPLAY                = 4,
        };

        enum SourceType {
```

3.10 Android9.0 和 Android10.0 Wifidisplay source 功能

Google 发布 Android9.0 及以上版本，已经把 Wifidsplay source 功能去掉了。2019 年 11 月份 RK 已经把 Wifidisplay source 功能代码移植进去了，需要 source 功能的客户可以把 SDK 代码更新到最新，或者提 redmine 获取补丁包。

3.11 Wifidisplay Sink 不能显示 Netflix 播放内容

检查 Wifdisplay Sink apk 代码是否有打开 HDCP 加密，打开 HDCP 加密需要在 apk 里面的 jni 目录下面的 Android.mk 文件修改下：LOCAL_CPPFLAGS += -DWFD_HDCP_SUPPORT，再加上

下面补丁。还有确认板子是否有烧写过 HDCP2.x key，同时确认 Source 端设备是否有支持 HDCP2.x 加密功能。

```
diff --git a/media/libmediaplayerservice/HDCP.cpp b/media/libmediaplayerservice/HDCP.cpp
index afe39367f..bd14ec379 100644
--- a/media/libmediaplayerservice/HDCP.cpp
+++ b/media/libmediaplayerservice/HDCP.cpp
@@ -164,7 +164,7 @@ void HDCP::ObserveWrapper(void *me, int msg, int ext1, int ext2) {
  }

  void HDCP::observe(int msg, int ext1, int ext2) {
-       Mutex::Autolock autoLock(mLock);
+       //Mutex::Autolock autoLock(mLock); //should be removed as it is a callback from hdcp library.

        if (mObserver != NULL) {
                mObserver->notify(msg, ext1, ext2, NULL /* obj */);
diff --git a/media/libmediaplayerservice/MediaPlayerService.cpp
b/media/libmediaplayerservice/MediaPlayerService.cpp
index 939307ba0..7bd493bbc 100755
--- a/media/libmediaplayerservice/MediaPlayerService.cpp
+++ b/media/libmediaplayerservice/MediaPlayerService.cpp
@@ -350,7 +350,14 @@ sp<IOMX> MediaPlayerService::getOMX() {
  }

  sp<IHDCP> MediaPlayerService::makeHDCP(bool createEncryptionModule) {
-       return new HDCP(createEncryptionModule);
+       ALOGI("MediaPlayerService::getHDCP");
+       Mutex::Autolock autoLock(mLock);
+
+       if (mHDCP.get() == NULL) {
+               mHDCP = new HDCP(createEncryptionModule);
+       }
+
+       return mHDCP;
  }

  sp<IRemoteDisplay> MediaPlayerService::listenForRemoteDisplay(
diff --git a/media/libmediaplayerservice/MediaPlayerService.h
b/media/libmediaplayerservice/MediaPlayerService.h
index 5e6e1c5c7..698f09e75 100755
--- a/media/libmediaplayerservice/MediaPlayerService.h
```

```
+++ b/media/libmediaplayerservice/MediaPlayerService.h
@@ -486,6 +486,7 @@ private:
                SortedVector< wp<MediaRecorderClient> > mMediaRecorderClients;
                int32_t                     mNextConnId;
                sp<IOMX>                      mOMX;
+               sp<IHDCP>                     mHDCP;
 };

 // ----------------------------------------------------------------------------
diff --git a/media/libstagefright/mpeg2ts/ATSParser.cpp b/media/libstagefright/mpeg2ts/ATSParser.cpp
index a256a4df1..cc57c5ce6 100644
--- a/media/libstagefright/mpeg2ts/ATSParser.cpp
+++ b/media/libstagefright/mpeg2ts/ATSParser.cpp
@@ -22,6 +22,9 @@
 #include "CasManager.h"
 #include "ESQueue.h"
 #include "include/avc_utils.h"
+#include <binder/IServiceManager.h>
+#include <media/IHDCP.h>
+#include <media/IMediaPlayerService.h>

 #include <android/hardware/cas/native/1.0/IDescrambler.h>
 #include <cutils/native_handle.h>
@@ -201,6 +204,7 @@ private:
     List<off64_t> mPesStartOffsets;

     ElementaryStreamQueue *mQueue;
+    sp<IHDCP> mHDCP;

     bool mScrambled;
     bool mSampleEncrypted;
@@ -1226,6 +1230,54 @@ status_t ATSParser::Stream::parsePES(ABitReader *br, SyncEvent *event) {
             optional_bytes_remaining -= 3;
         }

+        //HDCP Private Data
+        bool useHDCP = false;
+        uint32_t streamCTR = 0;
+        uint64_t outInputCTR = 0;
+        if(optional_bytes_remaining >= 17){
```

```
+                   uint8_t hdcp_private_flag = br->getBits(8);
+                   optional_bytes_remaining -= 1;
+                   if(hdcp_private_flag==0x8e){
+                       /* stream Counter */
+                         br->skipBits(13);
+                         streamCTR = br->getBits(2) << 30;
+                         br->getBits(1);
+                         streamCTR |= br->getBits(15) << 15;
+                         br->getBits(1);
+                         streamCTR |= br->getBits(15);
+                         br->getBits(1);
+
+                         /* input Counter */
+                         br->skipBits(11);
+                         outInputCTR = ((uint64_t)br->getBits(4)) << 60;
+                         br->getBits(1);
+                         outInputCTR |= ((uint64_t)br->getBits(15)) << 45;
+                         br->getBits(1);
+                         outInputCTR |= br->getBits(15) << 30;
+                         br->skipBits(1);
+                         outInputCTR |= br->getBits(15) << 15;
+                         br->getBits(1);
+                         outInputCTR |= br->getBits(15);
+                         br->getBits(1);
+
+                         useHDCP = true;
+                         //ALOGV("this packet has HDCP flag, CTR vlaue:%d:%lld", streamCTR,
outInputCTR);
+                         optional_bytes_remaining -= 16;
+                   }
+              }
+
+              if(useHDCP){
+                    if(mHDCP==NULL){
+                          sp<IServiceManager> sm = defaultServiceManager();
+                          sp<IBinder> binder = sm->getService(String16("media.player"));
+                          sp<IMediaPlayerService> service = interface_cast<IMediaPlayerService>(binder);
+                          CHECK(service != NULL);
+                          mHDCP = service->makeHDCP(false);
+                          if (mHDCP != NULL)
+                               ALOGD("successfully make HDCP decrypt object");
+                       }
+               }
+
                br->skipBits(optional_bytes_remaining * 8);

                // ES data follows.
@@ -1250,15 +1302,51 @@ status_t ATSParser::Stream::parsePES(ABitReader *br, SyncEvent *event)
{
                        ALOGV("There's %u bytes of payload, PES_packet_length=%u, offset=%d",
                                        dataLength, PES_packet_length, pesOffset);

-                       onPayloadData(
-                                       PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
-                                       br->data(), dataLength, pesOffset, event);
+                       if (mHDCP != NULL) {
+                               const uint8_t *pesData = br->data();
+                               size_t dataSize = br->numBitsLeft()/8;
+                               void *outData = malloc(dataSize);
+                               //ALOGD("packet length unkowned:%d",dataSize);
+                               status_t err =
mHDCP->decrypt(pesData,dataSize,streamCTR,outInputCTR,outData);
+                               if(err==OK){
+                                       onPayloadData(PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                                       (uint8_t*)outData, dataSize, pesOffset, event);
+                               }else{
+                                       ALOGE("decrypt pes failed");
+                                       onPayloadData(PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                                       (uint8_t*)pesData, dataSize, pesOffset, event);
+                               }
+                               free(outData);
+                               outData = NULL;
+                       }else {
+                               onPayloadData(
+                                               PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                               br->data(), dataLength, pesOffset, event);
+                       }

                        br->skipBits(dataLength * 8);
                } else {
```

```
-                   onPayloadData(
-                               PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
-                               br->data(), br->numBitsLeft() / 8, pesOffset, event);
+                   if (mHDCP != NULL) {
+                           const uint8_t *pesData = br->data();
+                           size_t dataSize = br->numBitsLeft()/8;
+                           void *outData = malloc(dataSize);
+                           //ALOGD("packet length unkowned:%d",dataSize);
+                           status_t err =
mHDCP->decrypt(pesData,dataSize,streamCTR,outInputCTR,outData);
+                           if(err==OK){
+                                   onPayloadData(PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                                   (uint8_t*)outData, dataSize, pesOffset, event);
+                           }else{
+                                   ALOGE("decrypt pes failed");
+                                   onPayloadData(PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                                   (uint8_t*)pesData, dataSize, pesOffset, event);
+                           }
+                           free(outData);
+                           outData = NULL;
+                   }else {
+                           onPayloadData(
+                                   PTS_DTS_flags, PTS, DTS, PES_scrambling_control,
+                                   br->data(), br->numBitsLeft() / 8, pesOffset, event);
+                   }

                    size_t payloadSizeBits = br->numBitsLeft();
                    if (payloadSizeBits % 8 != 0u) {
```

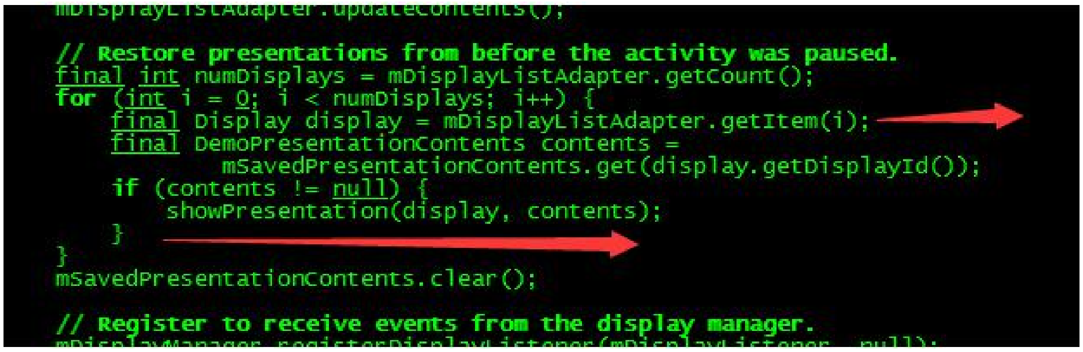
3.12 Wifidisplay HDCP2.x key 烧写

Wifidisplay 的 HDCP key 是用的 HDCP2.x，和 HDMI HDCP Transmit key 不是同一个，需要单独烧写，可以通过 FAE 窗口获取到烧写 HDCP key 的工具 RKDevInfoWriteTool_v1.2.3.zip。烧写过程在工具设置中把 RPMB 改成兼容模式，即可烧写。

3.13 Wifidisplay 投射主副屏选择

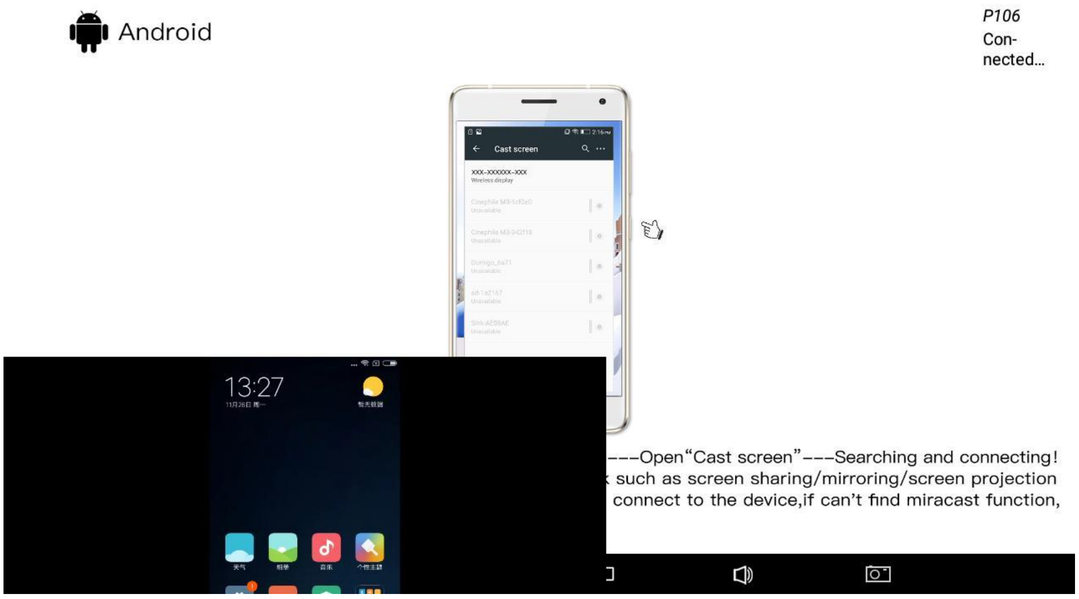
双屏异显的场景，默认投屏的是主屏，假如需要把副屏的内容通过 Wifidisplay 投射出去，可

以双屏异显的 app 把 presentation view 指定用 wifidisplay 的 id 绑定。在 PresentationActivity.java 文件。



```
mDisplayListAdapter.updateContents();

// Restore presentations from before the activity was paused.
final int numDisplays = mDisplayListAdapter.getCount();
for (int i = 0; i < numDisplays; i++) {
    final Display display = mDisplayListAdapter.getItem(i);
    final DemoPresentationContents contents =
            mSavedPresentationContents.get(display.getDisplayId());
    if (contents != null) {
        showPresentation(display, contents);
    }
}
mSavedPresentationContents.clear();

// Register to receive events from the display manager.
mDisplayManager.registerDisplayListener(mDisplayListener, null);
```

3.14 Wifidisplay 投屏显示问题


Android
P106
Con-
nected...
Cast screen
13:27
---Open“Cast screen”---Searching and connecting!
such as screen sharing/mirroring/screen projection
connect to the device,if can’t find miracast function,

没有全屏显示，只显示其中一部分。可以通过修改 setprop sys.display.oritation 1 来解决。

3.15 Wifidisplay 连续投屏 12.4h，画面出卡顿

有些客户发现长时间投屏后，出现画面卡顿问题。在 framework/av 目录下面打上下面修改：

```
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
index 0b5e159..b0e8561 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.cpp
@@ -868,7 +868,7 @@ size_t NuPlayer::Renderer::fillAudioBuffer(void *buffer, size_t size) {
                postEOSDelayUs = getPendingAudioPlayoutDurationUs(ALooper::GetNowUs());
```

```
                    }
                    ALOGV("fillAudioBuffer: notifyEOS "
-                               "mNumFramesWritten:%u    finalResult:%d    postEOSDelay:%lld",
+                               "mNumFramesWritten:%lld    finalResult:%d    postEOSDelay:%lld",
                                mNumFramesWritten, entry->mFinalResult, (long long)postEOSDelayUs);
                    notifyEOS(true /* audio */, entry->mFinalResult, postEOSDelayUs);
                }
@@ -1179,7 +1179,7 @@ bool NuPlayer::Renderer::onDrainAudioQueue() {
        return reschedule;
 }

-int64_t NuPlayer::Renderer::getDurationUsIfPlayedAtSampleRate(uint32_t numFrames) {
+int64_t NuPlayer::Renderer::getDurationUsIfPlayedAtSampleRate(uint64_t numFrames) {
        int32_t sampleRate = offloadingAudio() ?
                    mCurrentOffloadInfo.sample_rate : mCurrentPcmInfo.mSampleRate;
        if (sampleRate == 0) {
@@ -1187,7 +1187,7 @@ int64_t NuPlayer::Renderer::getDurationUsIfPlayedAtSampleRate(uint32_t
numFrames
            return 0;
        }
        // TODO: remove the (int32_t) casting below as it may overflow at 12.4 hours.
-       return (int64_t)((int32_t)numFrames * 1000000LL / sampleRate);
+       return (int64_t)((int64_t)numFrames * 1000000LL / sampleRate);
 }

 // Calculate duration of pending samples if played at normal rate (i.e., 1.0).
diff --git a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
index 2e5f9b2..f6472aa 100644
--- a/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
+++ b/media/libmediaplayerservice/nuplayer/NuPlayerRenderer.h
@@ -147,7 +147,7 @@ private:
        uint32_t mFlags;
        List<QueueEntry> mAudioQueue;
        List<QueueEntry> mVideoQueue;
-       uint32_t mNumFramesWritten;
+       uint64_t mNumFramesWritten;
        sp<VideoFrameScheduler> mVideoScheduler;

        bool mDrainAudioQueuePending;
```

```
@@ -289,7 +289,7 @@ private:
         void startAudioOffloadPauseTimeout();
         void cancelAudioOffloadPauseTimeout();

-        int64_t getDurationUsIfPlayedAtSampleRate(uint32_t numFrames);
+        int64_t getDurationUsIfPlayedAtSampleRate(uint64_t numFrames);

         DISALLOW_EVIL_CONSTRUCTORS(Renderer);
  };
```