

# GStreamer RTP和RTSP

GUPAO TECH

我们的愿景

推动每一次人才升级

我们的使命

让每个人的职业生涯不留遗憾

请在此处  
输入文字

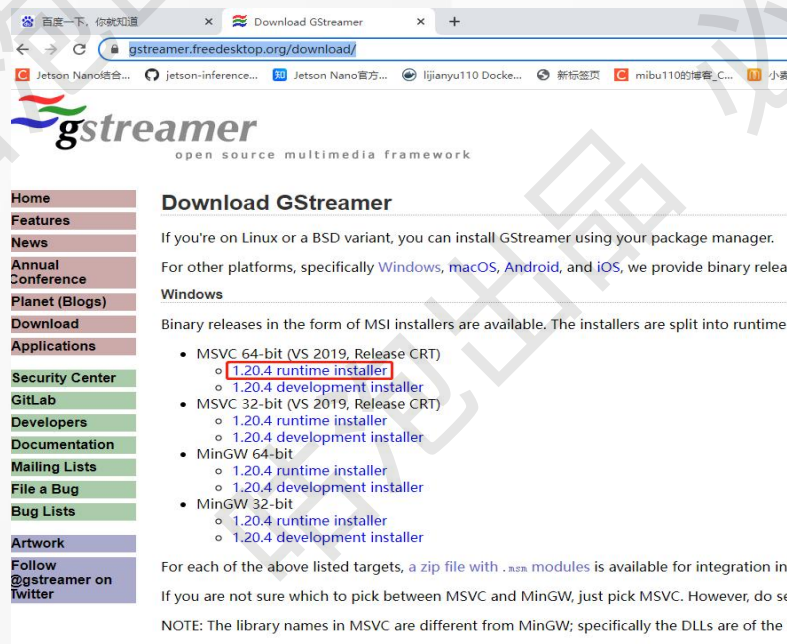
请在此处  
插入二维码

# RTP（传输层）

RTP全名是Real-time Transport Protocol（实时传输协议）。RTP用来为IP网上的语音、图像、传真等多种需要实时传输的多媒体数据提供端到端的实时传输服务。RTP为Internet上端到端的实时传输提供时间信息和流同步，但并不保证服务质量，服务质量由RTCP来提供。

# GStreamer Windows安装

<https://gstreamer.freedesktop.org/download/>



# GStreamer Windows安装

避坑：安装目录不要有中文和空格，选择完整安装

避坑：一定要选择运行版安装，不要选开发板

# RTP:Server UDP

jetson nano: 作为服务器端  
IP: 192.168.2.9

gst-launch-1.0 videotestsrc is-live=true ! jpegenc ! rtpjpegpay !  
udpsink **host=192.168.2.7** port=5004

host是客户端地址

# RTP:Client UDP

Window10 录屏台式机：客户端  
IP:192.168.2.7

```
gst-launch-1.0 udpsrc port=5004 ! application/x-rtp,encoding-  
name=JPEG ! rtpjpegdepay ! jpegdec ! autovideosink
```

# RTP:Server TCP

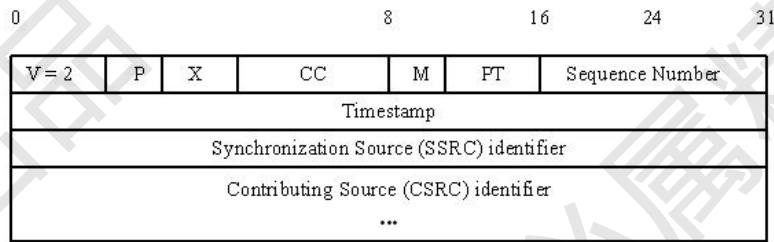
```
gst-launch-1.0 videotestsrc is-live=true ! jpegenc ! rtpjpegpay !  
rtpstreampay ! tcpserversink host=192.168.2.9 port=4953
```

# RTP:Client TCP

```
gst-launch-1.0 tcpclientsrc host=192.168.2.9 port=4953 !  
application/x-rtp-stream,encoding-name=JPEG !  
rtpstreamdepay ! rtpjpegdepay ! jpegdec ! autovideosink
```



# RTP包封装



版本号 (V) : 2比特, 用来标志使用的RTP版本。

填充位 (P) : 1比特, 如果该位置位, 则该RTP包的尾部就包含附加的填充字节。

扩展位 (X) : 1比特, 如果该位置位的话, RTP固定头部后面就跟有一个扩展头部。

CSRC计数器 (CC) : 4比特, 含有固定头部后面跟着的CSRC的数目。

标记位 (M) : 1比特, 该位的解释由配置文档 (Profile) 来承担。

载荷类型 (PT) : 7比特, 标识了RTP载荷的类型。

序列号 (SN) : 16比特, 发送方在每发送完一个RTP包后就将该值增加1, 接收方可以由该值确定包的丢失及恢复包序列。序列号的初始值是随机的。

时间戳: 32比特, 记录了该包中数据的第一个字节的采样时刻。在一次会话开始时, 时间戳初始化成一个初始值。即使在没有信号发送时, 时间戳的数值也要随时间而不断地增加 (时间在流逝嘛)。时间戳是去除抖动和实现同步不可缺少的。

同步源标识符(SSRC): 32比特, 同步源就是指RTP包流的来源。在同一个RTP会话中不能有两个相同的SSRC值。该标识符是随机选取的 RFC1889推荐了MD5随机算法。

# matroskamux 混合器模式

Server :

```
gst-launch-1.0 videotestsrc is-live=true ! jpegenc !  
matroskamux ! tcpserver sink host=192.168.2.9 port=4953
```

# matroskamux 混合器模式

Client :

```
gst-launch-1.0 tcpclientsrc host=192.168.2.9 port=4953 !
```

```
matroskademux ! jpegdec ! autovideosink
```

# H264编码

服务器端

```
gst-launch-1.0 -v videotestsrc ! 'video/x-raw,width=(int)1280, height=(int)720, framerate=20/1' !  
videoscale ! videoconvert ! x264enc tune=zerolatency  
bitrate=2048 speed-preset=superfast ! rtph264pay  
config-interval=1 pt=96 ! udpsink host=192.168.2.7  
port=5004
```

## H264编码

客户端(windows 平台下是", Linux下是')

```
gst-launch-1.0 -v udpsrc port=5004 caps =  
"application/x-rtp, media=(string)video, clock-  
rate=(int)90000, encoding-name=(string)H264,  
payload=(int)96" ! rtph264depay ! decodebin !  
videoconvert ! autovideosink sync=false
```

## H264 硬编码

服务器端

```
gst-launch-1.0 -v videotestsrc ! 'video/x-raw,format=(string)NV12,width=(int)320,height=(int)180,framerate=20/1' ! nvvidconv ! nvv4l2h264enc ! h264parse ! rtph264pay config-interval=1 pt=96 ! udpsink host=192.168.2.7 port=5004
```

# 视频网站

上面都是服务器端产生图片或视频发送到客户端，而且不能暂停等控制功能

如何像 优酷 一样提供视频流服务呢？可以播放器播放，控制

# 实时流传输协议 (Real Time Streaming Protocol, RTSP)

是TCP/IP协议体系中的一个应用层协议，由哥伦比亚大学、网景和RealNetworks公司提交的IETF RFC标准。该协议定义了一对多应用程序如何有效地通过IP网络传送多媒体数据。RTSP在体系结构上位于RTP和RTCP之上，它使用TCP或UDP完成数据传输



## 启动deepstream python的docker

```
docker run -it --rm --net=host --runtime nvidia -e  
DISPLAY=$DISPLAY -v /tmp/argus_socket:/tmp/argus_socket -  
-device /dev/video0 -v  
~/GStreamer:/opt/nvidia/deepstream/deepstream-  
6.0/sources/deepstream_python_apps/apps/GStreamer  
715a93f1d755
```

# RTSP 示例





```
1 import gi
2 gi.require_version('Gst', '1.0')
3 gi.require_version('GstRtspServer', '1.0')
4 from gi.repository import Gst, GObject, GstRtspServer
5
6 Gst.init(None)
7 mainloop = GObject.MainLoop()
8 server = GstRtspServer.RTSPServer()
9 factory = GstRtspServer.RTSPMediaFactory()
10 factory.set_launch(
11     ' ( videotestsrc is-live=1 '
12     '! video/x-raw,format=(string)NV12,width=(int)320,height=(int)180,framerate=20/1 '
13     '! nvvidconv ! nvv4l2h264enc ! h264parse ! rtph264pay name=pay0 bitrate=40000 pt=96 ) '
14 )
15 # allow multiple connections
16 factory.set_shared(True)
17 mounts = server.get_mount_points()
18 mounts.add_factory('/live', factory)
19 print('rtsp://192.168.2.9:8554/live')
20 server.attach(None)
21 mainloop.run()
22
```

# 客户端 EasyPlayer-RTSP

下载地址：

<https://github.com/tsingsee/EasyPlayer-RTSP>

<rtsp://192.168.2.9:8554/live>

名称	修改日期	类型
 Accelerated_GStreamer_User_Guide	2022/10/22 20:46	Chrome HT
 EasyPlayer-RTSP-Win-V3.0.19.0515	2022/10/18 22:46	WinRAR ZIP
 gstreamer-1.0-msvc-x86_64-1.20.4	2022/10/20 8:46	Windows In
 rtspVideoH264	2022/10/23 18:03	py

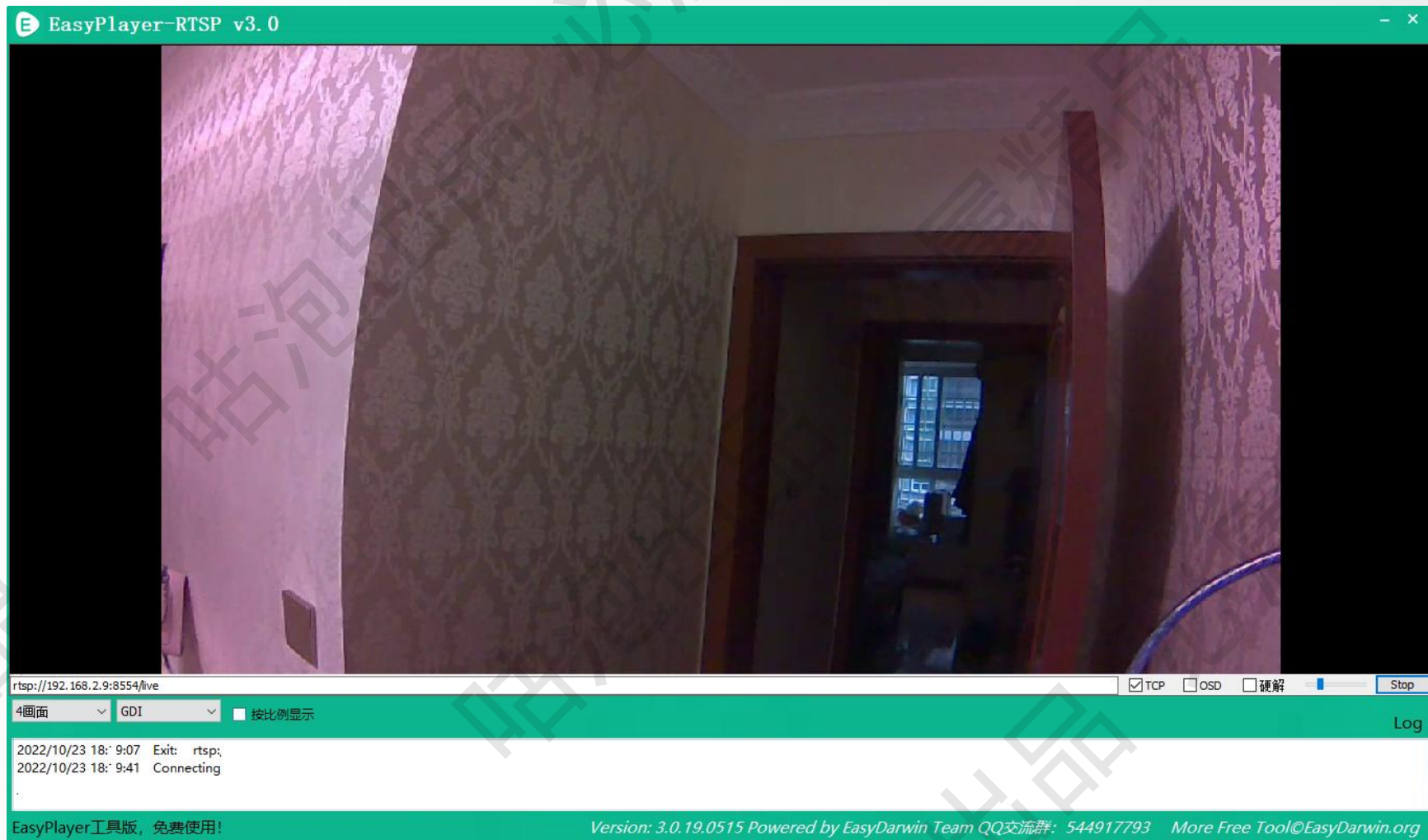
# 摄像头 RTSP代码

```
rtspCameraH264.py
1  import gi
2  gi.require_version('Gst','1.0')
3  gi.require_version('GstRtspServer','1.0')
4  from gi.repository import Gst, GObject, GstRtspServer
5
6  Gst.init(None)
7  mainloop = GObject.MainLoop()
8  server = GstRtspServer.RTSPServer()
9  factory = GstRtspServer.RTSPMediaFactory()
10 factory.set_launch(
11     ' ( nvarguscamerasrc '
12         '! video/x-raw(memory:NVMM),width=(int)1280,height=(int)720,format=(string)NV12,framerate=(frac
13         '! nvv4l2h264enc ! h264parse ! rtph264pay name=pay0 bitrate=40000 pt=96 ) '
14 )
15
16 # allow multiple connections
17 factory.set_shared(True)
18 mounts = server.get_mount_points()
19 mounts.add_factory('/live', factory)
20 print('rtsp://192.168.2.9:8554/live')
21 server.attach(None)
22 mainloop.run()
```

# 摄像头RTSP核心代码

```
factory.set_launch(  
    ' ( nvarguscamerasrc '  
    '! video/x-  
raw(memory:NVMM),width=(int)1280,height=(int)720,format=(string)NV12,framerate=(fraction)30/1 '  
    '! nvv4l2h264enc ! h264parse ! rtph264pay name=pay0  
    bitrate=40000 pt=96 ) '  
)
```





# 摄像头输出格式

```
zlx@nano128G:~$ v4l2-ctl --device=/dev/video0 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
    Index      : 0
    Type       : Video Capture
    Pixel Format: 'RG10'
    Name       : 10-bit Bayer RGRG/GBGB
                Size: Discrete 3264x2464
                    Interval: Discrete 0.048s (21.000 fps)
                Size: Discrete 3264x1848
                    Interval: Discrete 0.036s (28.000 fps)
                Size: Discrete 1920x1080
                    Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1640x1232
                    Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                    Interval: Discrete 0.017s (60.000 fps)
```

# 摄像头数据格式

摄像头的数据输出格式一般分为RAW RGB和非raw RGB格式。

Raw RGB是Sensor每个感光点感应到的RGB数值。

Raw RGB经过ISP的处理还原出三原色，输出YUV或者RGB格式。



# 摄像头数据格式 YUV

YUV 是一种颜色编码方法：与 RGB 编码方法类似

其中“Y”表示明亮度(Luminance 或 Luma)，也就是灰阶值，U 和 V 表示的则是色度(Chrominance 或 Chroma)，用来表示，作用是描述影像色彩及饱和度，用于指定像素的颜色

## YUV格式好处

- 1.可以避免相互干扰(没有 UV 信息一样可以显示完整的图像，因而解决了彩色电视与黑白电视的兼容问题);
- 2.降低色度的采样率而不会对图像质量影响太大，降低了视屏是信号传输时对频宽(带宽)的要求。

# YUV数据格式有哪些

YU12、I420、YV12、NV12、NV21、YUV420P、YUV420SP、YUV422P、YUV444P

# nvarguscamerasrc 插件

```
Pad Templates:  
  SRC template: 'src'  
  Availability: Always  
  Capabilities:  
    video/x-raw(memory:NVMM)  
      width: [ 1, 2147483647 ]  
      height: [ 1, 2147483647 ]  
      format: { (string)NV12, (string)P010_10LE }  
      framerate: [ 0/1, 2147483647/1 ]
```

# 插件学习 nvv4l2h264enc

```
Pad Templates:
SRC template: 'src'
Availability: Always
Capabilities:
  video/x-h264
    stream-format: byte-stream
    alignment: { (string)au, (string)nal }

SINK template: 'sink'
Availability: Always
Capabilities:
  video/x-raw(memory:NVMM)
    width: [ 1, 2147483647 ]
    height: [ 1, 2147483647 ]
    format: { (string)I420, (string)NV12, (string)P010_10LE, (string)NV24 }
    framerate: [ 0/1, 2147483647/1 ]
```

# nvv4l2h264enc

## GPU 运算

(memory:NVMM)



# nvvidconv 插件

## 内存交换 格式转换

```
Pad Templates:
  SINK template: 'sink'
  Availability: Always
  Capabilities:
    video/x-raw(memory:NVMM)
      format: { (string)I420, (string)I420_10LE, (string)P010_10LE, (string)I4
20_12LE, (string)UYVY, (string)YUY2, (string)YVYU, (string)NV12, (string)NV16, (string)NV
24, (string)GRAY8, (string)BGRx, (string)RGBA, (string)Y42B }
      width: [ 1, 2147483647 ]
      height: [ 1, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
    video/x-raw
      format: { (string)I420, (string)UYVY, (string)YUY2, (string)YVYU, (string)NV12, (string)NV16, (string)NV24, (string)P010_10LE, (string)GRAY8, (string)BGRx, (string)RGBA, (string)Y42B }
      width: [ 1, 2147483647 ]
      height: [ 1, 2147483647 ]
      framerate: [ 0/1, 2147483647/1 ]
```

# nvvidconv 插件

## 内存交换 格式转换

```
SRC template: 'src'
Availability: Always
Capabilities:
  video/x-raw(memory:NVMM)
    format: { (string)I420, (string)I420_10LE, (string)P010_10LE, (string)UY
VY, (string)YUY2, (string)YVYU, (string)NV12, (string)NV16, (string)NV24, (string)GRAY8,
(string)BGRx, (string)RGBA, (string)Y42B }
    width: [ 1, 2147483647 ]
    height: [ 1, 2147483647 ]
    framerate: [ 0/1, 2147483647/1 ]
  video/x-raw
    format: { (string)I420, (string)UYVY, (string)YUY2, (string)YVYU, (string)
NV12, (string)NV16, (string)NV24, (string)GRAY8, (string)BGRx, (string)RGBA, (string)Y4
2B }
    width: [ 1, 2147483647 ]
    height: [ 1, 2147483647 ]
    framerate: [ 0/1, 2147483647/1 ]
```



# 谢谢观赏

GUPAO TECH



替换小标题文字，或简要说明

我们的愿景

推动每一次人才升级

我们的使命

让每个人的职业生涯不留遗憾

请在此处  
输入文字

请在此处  
插入二维码