<http://www.einarsundgren.se/gstreamer-basic-real-time-streaming-tutorial/>

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

pi没有搞远程桌面那样的图形界面会导致命令执行的失败，而且性能是个问题，我把系统可能搞坏也是个问题，所以就在自己kali上面练手。但是如果你的电脑只有核显的话，虚拟机使用不来va-api加速的。

gstreamer是个C/C++框架也是可以让bash直接执行的命令套件

套件名字：gst-\*

gst-inspect-1.0:

直接使用会显示当前机子上所有的plugins，可以使用grep过滤

如果后面加上pluginsname会列出关于它的详细信息

pi@raspberrypi:~ $ gst-inspect-1.0 | grep udpsink

udp: udpsink: UDP packet sender

udp: multiudpsink: UDP packet sender

udp: dynudpsink: UDP packet sender

https://gstreamer.freedesktop.org/documentation/plugins.html 插件列表

gst-launch-1.0：

官方给的示例：

gst-launch-1.0 -v fakesrc silent=false num-buffers=3 ! fakesink silent=false

gst-launch-1.0 videotestsrc ! videoconvert ! autovideosink

第一条是进行配置，第二条是设置pipeline，用!分开，读取-》转码-》播放。

-v verbose Create a test video stream

成功的话，会显示一个老式电视机搜不到台的花窗口。

videotestsrc 是安装gstreamer自带的测试输入文件。

xxxxsink就是输出 视频播放使用的是解压以后的裸流，sink了以后似乎就不能再加上一般的管道了。

A pipeline consists elements and links. Elements can be put into bins

of different sorts. Elements, links and bins can be specified in a

pipeline description in any order.

filesrc location=xxx就是属性 （对应是filesink location=xxx）

links估计就是指那些有输入也有输出用于连接的插件

raspivid -t 0 -h 720 -w 1080 -fps 25 -hf -b 2000000 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! gdppay ! tcpserversink host=YOUR\_RPI\_IP\_ADDRESS port=5000

fdsrc是从文件描述符中获取数据 大概默认了后面是fd=0吧

h264parse就是解析是不是h264

如果想主动往服务器发送数据，可以通过tcpclientsink插件进行传输

gdppay Payloads GStreamer Data Protocol buffers这样子装了的话之后到了还需要使用gdbdepay。

gst-rtsp-server对应使用这个指令：./test-launch "( tcpserversrc host=192.168.1.30 port=2223 ! gdpdepay ! rtph264pay name=pay0 pt=96 )" 有一个问题就是这个不能使用vlc客户端来打开，使用vlc来打开就会只显示一下然后出现花屏问题并且花屏了以后中断连接之后也不能再打开rtsp流或者可能直接说不能打开，我使用了其他播放器（可能是使用gstreamer的rtsp播放库写的）才行。

需要注意，因为rtsp服务器是需要数据的时候才接受来自tcpserversrc的数据，所以必须在手机或者客户端连接上rtsp的服务器端的时候，打开data-client才有效。打开顺序为 rtsp-server->rtsp-client->tcpclientsink

如果使用的是udpsink那么不管你收不收，datacline都会耗流量在那里发送，而与tcpclinet只有当有rtsp-server由使用者请求的时候也就是需要数据了才会接受数据。

filesrc location=xxxx

gst-launch videotestsrc ! video/x-raw-rgb, framerate=25/1, width=1280, height=720 ! fbdevsink device=/dev/fb0

中间设置video参数。

与树莓派之间的nc连接，延时高而且不稳定会导致卡：

raspivid -fps 25 -o - | nc -lp 2222

nc nc 182.168.1.30 2222 | gst-launch-1.0 fdsrc ! decodebin ! videoconvert ! autovideosink

decodebin Autoplug and decode to raw media

gst-launch-1.0 -v filesrc location='test.h264' ! decodebin ! autovideoconvert ! autovideosink

You asked for a h264 decoder and mention a few modules. Here are som comments.

1) h264parser is not a decoder, but rather a module to parse a h264 stream. It may be able to convert a h264 stream from NAL units to AU units and perhaps vice versa, but that's it. You use it in a stream to ensure that the following modules will receive the right caps information and perhaps for the unit conversion.

2) rtpXXXpay and rtpXXXdepay are modules used to encapsulate and decapsulate a stream for RTP streaming. In your case you may only need the modules, if you need to decapsulate an RTP stream.

3) Never used vdpauh264dec, so search Google for examples.

Now for decoding h264, you can among other things also do

a) use ffdec\_h264

b) use decodebin and decodebin2

c) use playbin and playbin2

Note that solution a) requires you to do proper decapsulating and/or proper demuxing before you can decode.

在做gstreamer项目的时候某些时候需要主动发送设备中采集的数据到服务端，这样就可以利用tcpclientsink和udpsink插件，主动发送数据到指定的服务器。

stream H.264 with RTSP on Raspberry Pi?

gst-launch-1.0 filesrc location=test.h264 ! h264parse ! rtph264pay ! rtph264depay ! decodebin ! autovideoconvert ! autovideosink

上面中间少一个插件都不行

RTSP的端口是tcp8554

rtph264depay ! gdbpay ! tcpclientsink host=xxx port =8554

encode and send H264 video via RTSP:最后还有发送

video test source:

videotestsrc ! imxvpuenc\_h264 bitrate=1000 ! rtph264pay name=pay0 pt=96"

or alternatively live captured video:

imxv4l2videosrc device=/dev/video0 ! imxipuvideotransform ! imxvpuenc\_h264DSD bitrate=1000 ! rtph264pay name=pay0 pt=96

gst-rpicamsrc is a GStreamer wrapper around the raspivid/raspistill functionality of the RaspberryPi, providing a GStreamer

source element capturing from the Rpi camera. https://github.com/thaytan/gst-rpicamsrc gst-launch-1.0 rpicamsrc bitrate=1000000 ! filesink location=test.h264

rtsp流转发摄像头：

1.获取树莓派的摄像头的视频源并且输出到 gstreamer中，并将数据传输到tcpserversink中，设定端口为5000

raspivid -t 0 -w 800 -h 600 -fps 25 -g 5 -b 4000000 -vf -n -o - | gst-launch-1.0 fdsrc ! h264parse ! gdppay ! tcpserversink host=127.0.0.1 port=5000clear

2.在gst-rtsp-server的example中test-launch，从本机的5000端口获取数据流作为服务器的源数据，建立rtsp的服务器。

./test-launch "( tcpclientsrc host=127.0.0.1 port=5000 ! gdpdepay ! rtph264pay name=pay0 pt=96 )"

3.在ubuntu14.2 中安装gstreamer ，执行以下代码来获取摄像头的图像

gst-launch-1.0 -v rtspsrc location=rtsp://192.168.11.22:8554/test ! gdpdepay ! rtph264depay ! avdec\_h264 ! videoconvert ! autovideosink

但是表示很卡，第三步我是在windows上使用vlc打开网络串流实现的。网速稳定在700KB/S但是还是卡。

如此看来之前./test-readme、./test-video等导致vlc无法播放，估计是gst-launch与gst-rtsp-server的联动问题，真是xxxx

raspivid -t 0 -w 800 -h 600 -fps 25 -g 5 -b 4000000 -vf -n -o - | ./test-launch "( fdsrc ! h264parse ! rtph264pay name=pay0 pt=96 )"虽然raspivid输出就是h264，但还是要经过h264parse，才能被rtph264pay接受。test-launch是将自己当成rtsp服务器。

-g, --intra : Specify the intra refresh period (key frame rate/GoP size). Zero to produce an initial I-frame and then just P-frames.

-f, --fullscreen : Fullscreen preview mode

-n, --nopreview : Do not display a preview window

如果插件错误，rtsp-server会一切照常但是vlc连不上去。不过现在这条命令虽说能用vlc连接，但是效果还不如没有，遇到的问题有：只能看到第一帧的图片，同时如果你退出以后，就连接不上去了。

而下面这条命令运行地很好，需要的客户端下载网速也就70KB/s：

raspivid -n -w 800 -h 800 -b 600000 -fps 25 -fl -t 0 -o - | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/test}' :demux=h264

使用top或者安装htop（可以看cpu支持几个线程）来查看硬件消耗：下面是有一个vlc客户端来连接的时候

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

pi 20 0 50040 12760 9136 S 1.3 1.4 0:00.87 vlc

pi 20 0 110932 7092 5596 S 2.3 0.8 0:00.60 lt-test-launch

　　PID：进程的ID

　　USER：进程所有者

　　PR：进程的优先级别，越小越优先被执行

　　NInice：值

　　VIRT：进程占用的虚拟内存

　　RES：进程占用的物理内存

　　SHR：进程使用的共享内存

　　S：进程的状态。S表示休眠，R表示正在运行，Z表示僵死状态，N表示该进程优先值为负数

　　%CPU：进程占用CPU的使用率

　　%MEM：进程使用的物理内存和总内存的百分比

　　TIME+：该进程启动后占用的总的CPU时间，即占用CPU使用时间的累加值。

　　COMMAND：进程启动命令名称

raspivid -t 999999 -h 1920 -w 1080 -fps 30 -b 2000000 -o - | ./test-launch " fdsrc ! h264parse ! rtph264pay name=pay0 pt=96 "则直接连第一帧都没有显示，什么图像都没有。这是因为我的windows vlc客户端的问题。换了一个WMPlayer.exe就好了。如果要查看vlc日志的话，工具-》消息。

raspivid -t 0 -w 800 -h 600 -g 10 -b 8000000 -v -o - | gst-launch-1.0 fdsrc ! h264parse ! rtph264pay name=pay0 pt=100 ! tcpserversink host=192.168.1.30 port=8554 使用tcpserversink打开失败

不知道是不是老师版本太高了，真的是没法和我apt-get到的gstreamer配合。

http://dongseopshin.tistory.com/11

以及

raspivid -t 0 -w 1080 -h 720 -fps 25 -hf -b 2000000 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=1 pt=96 ! gdppay ! tcpserversink host=VIDSERVERIP port=5000

gst-launch-1.0 -v tcpclientsrc host=VIDSERVERIP port=5000 ! gdpdepay ! rtph264depay ! avdec\_h264 ! videoconvert ! autovideosink sync=false

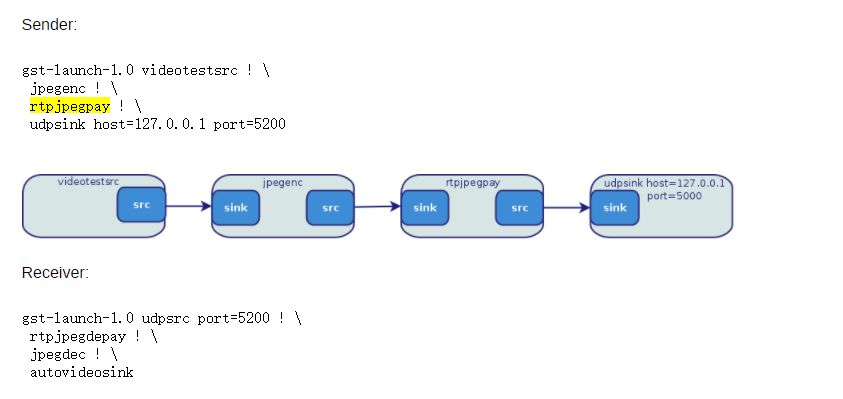
重新编译gstreamer

libtool: warning: relinking 'libgstcoretracers.la'

gst-launch-1.0 videotestsrc ! jpegenc ! rtpjpegpay ! udpsink host=127.0.0.1 port=5200

从视频中截取图片并rtsp发送

将jpeg以rtp发出，要求最终能被vlc查看



我将上图中的Sender变换成test-launch 并去掉了最后的udpsink，却失败了：No RTP format was negotiated….… Input buffers need to have RTP caps set on them…….将接收端指定加上cap的指定就好了：gst-launch-1.0.exe udpsrc port=8554 caps="application/x-rtp" ! rtpjpegdepay ! jpegdec ! autovideosink 理论上即使用capsfilter的指定gst-launch udpsrc port=8554 ! application/x-rtp ! rtpjpegdepay ! jpegdec ! autovideosink 但是只有前者有效。

raspivid结合rtpjpegpay的发送：raspivid -t 0 -h 800 -w 600 -cd MJPEG -o - | gst-launch-1.0 fdsrc ! image/jpeg ! jpegparse ! rtpjpegpay ! udpsink host=192.168.1.147 port=8554 。client：gst-launch-1.0.exe udpsrc port=8554 caps="application/x-rtp" ! rtpjpegdepay ! jpegparse ! multifilesink location="%d.jpeg"

raspistill结合rtpjpegpay的使用：raspistill -w 800 -h 600 -tl 0 -t 0 -o - | test-launch "fdsrc ! image/jpeg,width=800,height=600 ! jpegparse ! rtpjpegpay name=pay0 pt=96"。client：gst-launch-1.0.exe rtspsrc location=rtsp://192.168.1.30:8554/test ! rtpjpegdepay ! jpegparse !  multifilesink location="%d.jpeg"     或者client：gst-launch-1.0.exe rtspsrc location=rtsp://192.168.1.30:8554/test ! rtpjpegdepay ! jpegparse !  jpegdec ! videoconvert ! autovideosink  不过真的很卡

gstreamer直接调用摄像头： v4l2src 有必要的话可以具体指定device

test-launch "( v4l2src device=/dev/video0 ! image/jpeg,width=1280,height=720,framerate=30/1 ! jpegparse ! videorate ! rtpjpegpay name=pay0 pt=96 )"

sudo ldconfig的作用

gst-rtsp-server其实都是调用了gst-launch-1.0来包装成为rtp数据，再加一个对rtsp的服务器控制

#include <gst/gst.h>

#include <gst/rtsp-server/rtsp-server.h>

int main (int argc, char \*argv[])

{

GMainLoop \*loop;

GstRTSPServer \*server;

GstRTSPMountPoints \*mounts;

GstRTSPMediaFactory \*factory;

gst\_init (&argc, &argv);

loop = g\_main\_loop\_new (NULL, FALSE);

/\* create a server instance \*/

server = gst\_rtsp\_server\_new ();

/\* get the mount points for this server, every server has a default object

\* that be used to map uri mount points to media factories \*/

mounts = gst\_rtsp\_server\_get\_mount\_points (server);

/\* make a media factory for a test stream. The default media factory can use

\* gst-launch syntax to create pipelines.

\* any launch line works as long as it contains elements named pay%d. Each

\* element with pay%d names will be a stream \*/

factory = gst\_rtsp\_media\_factory\_new ();

gst\_rtsp\_media\_factory\_set\_launch (factory,

"( videotestsrc is-live=1 ! x264enc ! rtph264pay name=pay0 pt=96 )");

gst\_rtsp\_media\_factory\_set\_shared (factory, TRUE);

/\* attach the test factory to the /test url \*/

gst\_rtsp\_mount\_points\_add\_factory (mounts, "/test", factory);

/\* don't need the ref to the mapper anymore \*/

g\_object\_unref (mounts);

/\* attach the server to the default maincontext \*/

gst\_rtsp\_server\_attach (server, NULL);

/\* start serving \*/

g\_print ("stream ready at rtsp://127.0.0.1:8554/test\n");

g\_main\_loop\_run (loop);

return 0;

}

读取：

1. vlc打开网络串流（或者使用mplayer，h264流的rtsp可以使用WMPlayer.exe，最后的方法就是gstreamer自己解码以后autovideosink显示）

2.ffmplay rtsp://127.0.0.1:8554/test

3.gst-launch -v rtspsrc location=rtsp://127.0.0.1:8554/test ! queue ! decodebin ! ffmpegcolorspace ! autovideosink

奇怪的错误：

gst-launch-1.0 videotestsrc ! atuovideosink

libva info : VA-API version 0.39.0 xxxx 我只有核显，但是在Vmware中核显虚拟的有问题，笔记本有独显倒是可以

gst-launch videotestsrc ! atuovideosink

success!