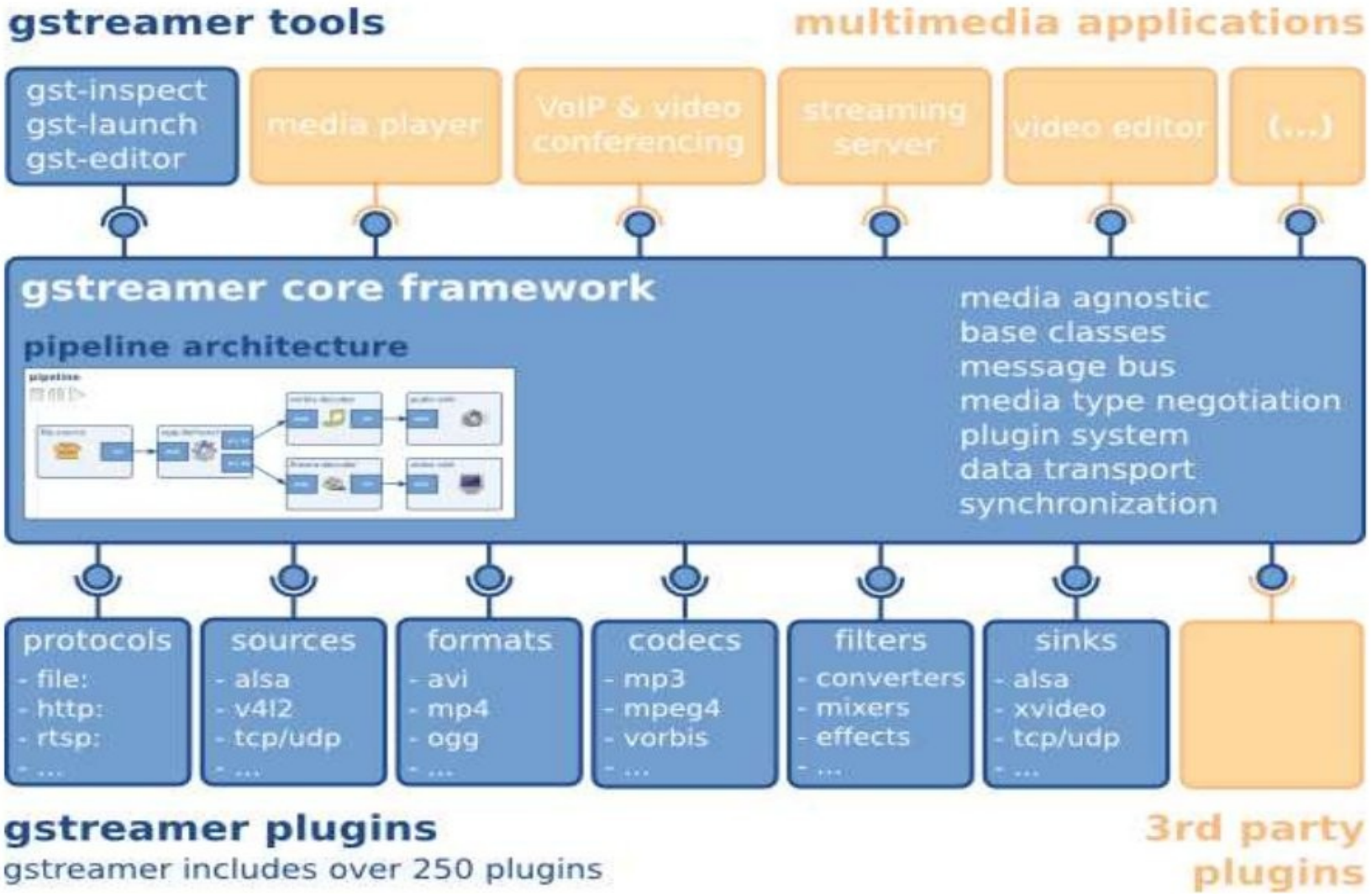
Internet & Multimedia

Exercises: using GStreamer

P. Bakowski

bako@ieee.org

GStreamer concepts



GStreamer concepts

Gstreamer and input/output: sources and sinks

files – input and output files

input devices, e.g. webcam; output devices, e.g. speaker, screen

network protocols: UDP,TCP, RTP/UDP, RTP/RTCP/UDP, ..

Gstreamer pipeline

GStreamer : source file & decoding

First example:

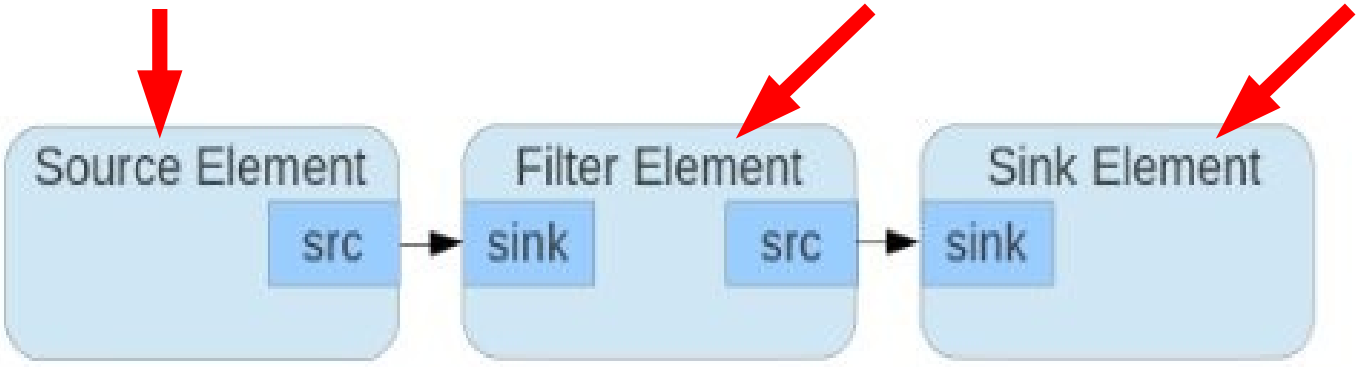
```
gst-launch-0.10  filesrc  location=hello.ogg !
oggdemux ! vorbisdec ! alsasink
```

Exercise: write the similar pipeline that decodes and plays an mp3 file. The mp3 decoder is called `mad`. (no **`demux`** needed), mp3 file has only audio track.

GStreamer : source file & decoding

Second example:

```
gst-launch-0.10
filesrc location=hello.ogg ! oggdemux ! vorbisdec !
audioconvert ! audioresample ! autoaudiosink
```



Exercise: write the similar pipeline that decodes and plays an mp3 file with **`resample`**. The mp3 decoder is called `mad`.

Decoding & coding

Third example:

```
gst-launch-0.10 -v filesrc location=Coffee.mp3 !
mad ! audioconvert ! audioresample !
vorbisenc ! oggmux ! filesink location=Coffee.ogg
```

Exercise: Draw the scheme of the pipeline and write a similar pipeline that decodes an **ogg** file and codes it as **mp3** file. The **mp3** encoder is called **lamemp3enc**.

Decoding with decodebin

Fourth example:

```
gst-launch-0.10  filesrc  location=hello.ogg !
decodebin ! autoaudiosink
```

decodebin finds automatically the necessary elements – **oggdemux** and **vorbisdec**

Decoding and coding

Fifth example – pipeline with 2 threads:

```
gst-lauch filesrc location="video.mp4" ! decodebin
name=d \
d. ! queue ! audioconvert ! audioresample !
vorbisenc ! oggmux name=mux \
d. ! queue ! theoraenc ! mux. \
mux. ! filesink location="out.ogv"
```

Exercise: Draw the scheme of the pipeline and write a similar pipeline that decodes an **ogv** file and codes it as **mp4** file. **Theora** decoder is called **theoradec**, **Vorbis** decoder is called **vorbisdec.** The **mp3** encoder is called **lamemp3enc**. The **h264** encoder is called **x264enc**.

Using webcam

Sixth example :

```
gst-launch-0.10 -v v4l2src ! video/x-raw-
yuv,width=320,height=240 ! ffmpegcolorspace !
autovideosink
```

Exercise: Draw the scheme of the pipeline and write a similar pipeline that creates a smaller (**160x120 pixels)** image.

Using webcam

Seventh example :

```
gst-launch-0.10 v4l2src ! 'video/x-raw-yuv,
width=640,height=480,framerate=30/1' !
theoraenc bitrate=256 !
oggmux ! filesink location=webcam.ogg
```

Exercise: Draw the scheme of the pipeline and write a similar pipeline that creates an **avi** file with **h264** video encoder. Use **x264enc** encoder and **avimux** container creator. Comment on **bitrate** value !

Using audio with webcam

Eight example :

```
gst-launch-0.10 -v alsasrc device=hw:1,0 !
audioconvert ! 'audio/x-raw-
int,rate=48000,channels=1' ! autoaudiosink
```

Exercise: Draw the scheme of the pipeline and write a similar pipeline that creates an audio file (mp3)

Sending media over UDP protocol

Ninth example :

```
gst-launch-0.10 filesrc location=$1 ! queue !
udpsink host=localhost port=9999

gst-launch-0.10 udpsrc port=9999 ! queue !
decodebin ! audioconvert ! autoaudiosink
```

Exercise: Draw the corresponding pipelines and write a similar example that sends the audio to external host (use an IP address). Note the presence of **queue** that operates as buffer.

Sending media over TCP protocol

Tenth example :

```
#the client of tcpserver - to be launched first !
echo 'waiting for the connection from server'
gst-launch-0.10 tcpserversrc host=172.19.64.141
port=5002 ! decodebin ! autoaudiosink

gst-launch-0.10 -v  alsasrc device=hw:1,0 !
audioconvert ! 'audio/x-raw-int,
rate=48000,channels=1' ! speexenc ! queue !
tcpserversink host=172.19.64.141 port=5002
```

Exercise: Explain the differences between UDP and TCP solutions !

Sending media over RTP/UDP

Eleventh example :

```
gst-launch -v v4l2src ! 'video/x-raw-
yuv,width=320,height=240' ! jpegenc  ! rtpjpegpay ! udpsink
host=172.19.64.141 port=5011
```

```
gst-launch udpsrc port=5011 \
caps="application/x-rtp, media=(string)video,
clock-rate=(int)90000, encoding-name=(string)JPEG,
encoding-params=(string)1, payload=(int)26" ! \
gstrtpjitterbuffer ! rtpjpegdepay ! jpegdec ! autovideosink
```

Exercise: Explain the usage of **caps** parameters and their relation to RTP protocol

Sending media over RTP/UDP

Twelve example :

```
gst-launch -v alsasrc device=hw:1,0 ! audioconvert !
audioresample ! \
'audio/x-raw-int,rate=16000,width=16,channels=1' ! speexenc !
rtpspeexpay ! udpsink host=172.19.64.141 port=5001

gst-launch udpsrc port=5001 caps="application/x-rtp,
media=(string)audio, clock-rate=(int)16000,
encoding-name=(string)SPEEX, encoding-params=(string)1,
payload=(int)110" ! gstrtpjitterbuffer name=jbuf latency=70
drop-on-latency=True ! rtpspeexdepay ! speexdec !
audioconvert ! audioresample ! volume volume=10 !
autoaudiosink
```

Summary