How to work with dynamic pipelines using GStreamer

gstreamer
Conference 2016

11-12 October 2016
Berlin

Jose A. Santos
santoscadenas@gmail.com

About me

José Antonio Santos Cadenas

- Software Engineer
- Telematic Systems Master's
- Kurento Media Server (KMS) manager
- santoscadenas@gmail.com

Gstreamer static pipelines

- gst-launch
  - gst-launch-1.0 filesrc location=sample.mp4 ! qtdemux ! avdec_h264 ! queue ! vp8enc ! webmmux ! filesink location=sample.webm
- This creates a complex media pipelie that transcodes
- Easy to be created

Gstreamer dynamic elements

- Gstreamer already has dynamic elements that simplify the creation of some dynamic pipelines
  - autovideosrc
  - autovideosink
  - decodebin
  - playbin
- Previous pipeline will be like this
  - gst-launch-1.0 filesrc location=sample.mp4 ! decodebin ! vp8enc ! webmmux ! filesink location=sample.webm

Dynamic pipelines not dependent on media (I)

- Previous elments allow to create dynamic pipelines dependent on media flow, not on external conditions
- For example: adding an replacing elements depending on user actions

Dynamic pipelines not dependent on media (II)

- Creating this applications requires a deep understanding of GStreamer:
  - How media flows between pads
  - How are the negotiations done
  - How streaming thread works
  - How probes work

Adding and removing elements while playing

- Working with this example: https://github.com/jcaden/gst-dynamic-examples
- The wrong way:
  - directly calling gst_pad_unlink
- Problems:
  - Streaming threads continues pushing buffers events and queries
  - Produce a deadlock if the streaming thread was running into elements that are being removed
- Conclusion: fails depending on race conditions

Adding and removing elements while playing

- The correct way:

  using a probe waiting for the pad to be idle

- How it works:

  Probe is called when the pad is not pushing media and is guaranteed that no media will be flowing while idle callbacks are being called

- Conclusion: Works always even if the disconnection time is long

Adding elements after a tee

- Tee takes care of disconnected pads
- Nevertheless, it is a good idea to handle connection of elements into idle or block callbacks to avoid problems while changing elements states or during negotiation

Be carefull with negotiations

- When connecting elements after a tee, you have to be aware of the fact that a negotiation will happen, and this negotiation could affect other branches
- You have to add capsfilter or converter to ease the negotiation

Removing elements

- Some times is important to allow elements to process all the queued buffers. (eg: when recording)
- Once disconnection is done, EOS has to be sent and waited at the end of the pipeline
  - If you send the event but not wait, you don't have guarantees that the event is really processed because there can be queues

Advices for live pipelines

- When working with live pipelines, where realtime is important elements should be configured to work as much fast as possible (see encoders configuration in sample program)
- Add queues to separate process in different threads

Remember

- Dynamic pipeline are not an easy tast, understand what to attemp to do and the consequences on other parts of the pipeline
- Block pads before disconnecting
- Sync state of new elements before connecting
- Always connect sink elements firts to avoid media leaks

Thank you

http://www.nubomedia.eu

http://www.kurento.org
http://www.github.com/kurento
info@kurento.org
Twitter: @kurentoms

http://www.fi-ware.org

http://ec.europa.eu

Suggestions, comments and complains:
santoscadenas@gmail.com