Multimedia programming with Gstreamer and Python

http://gstreamer.freedesktop.org

gstreamer

from the glib/GTK/Gnome universe

runs everywhere

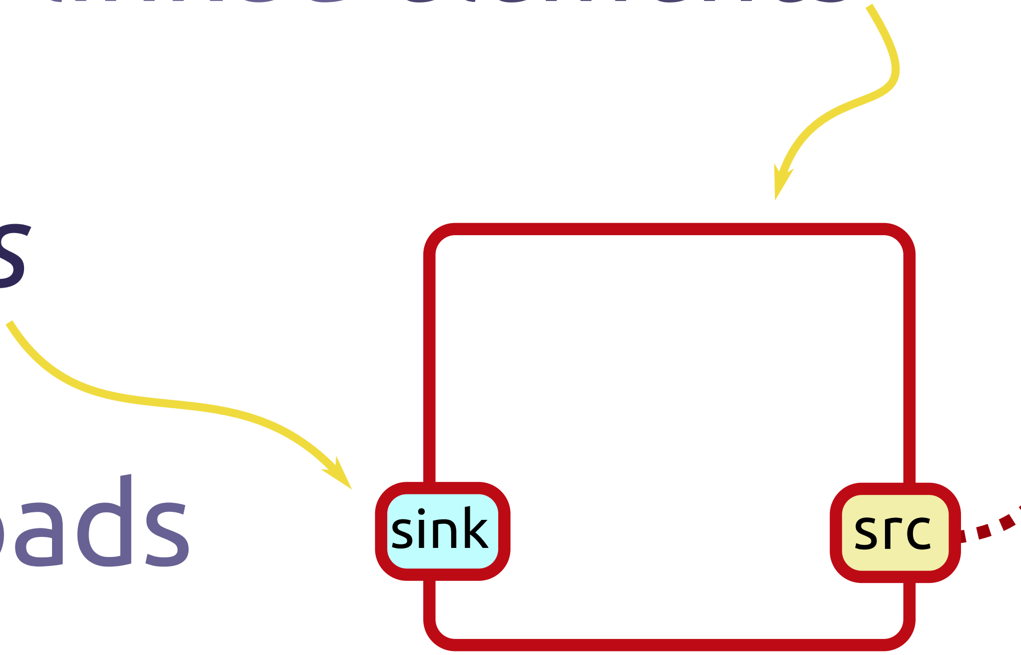
based on *pipelines*

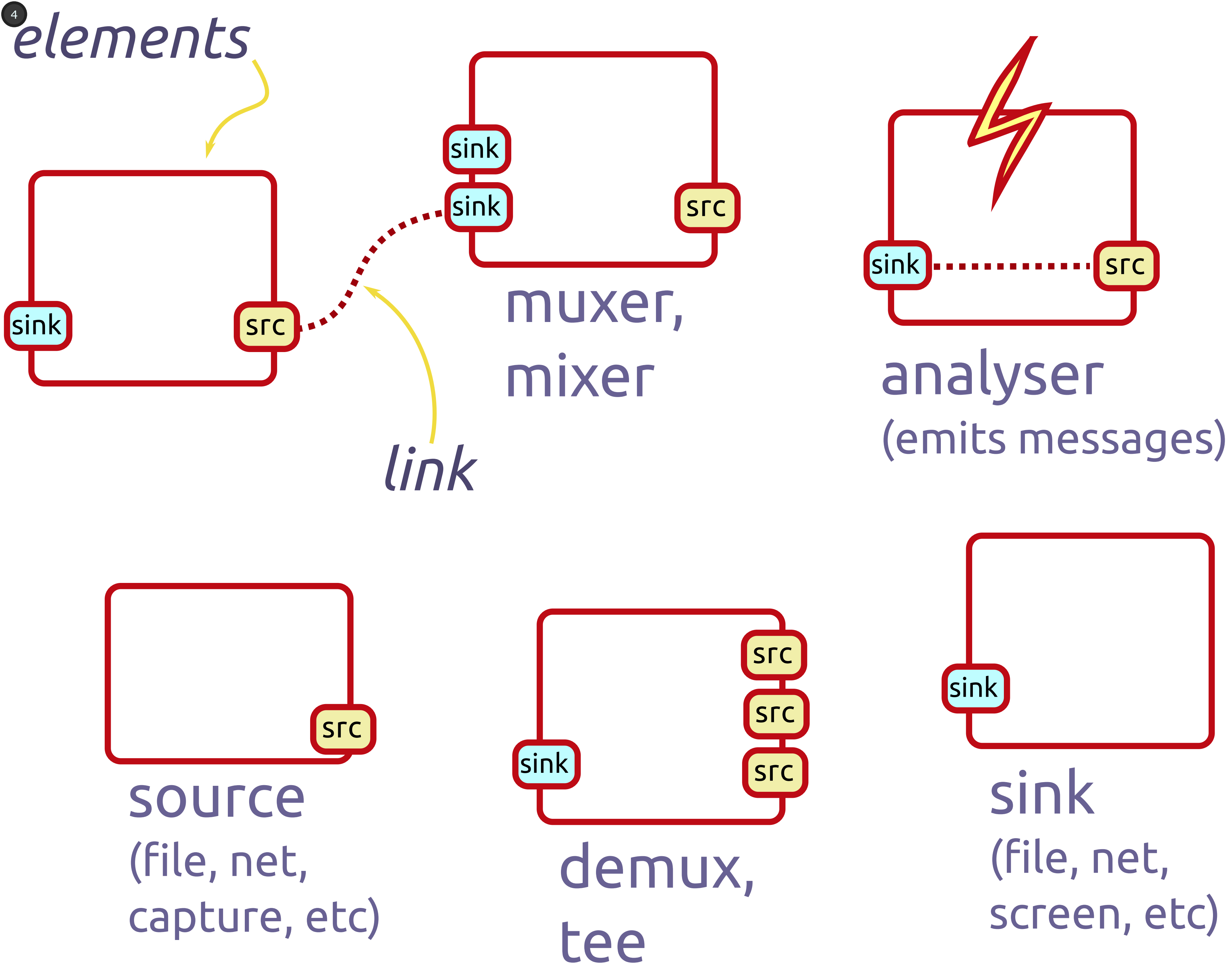
based on *pipelines*

pipelines consist of linked *elements*

elements have *pads*

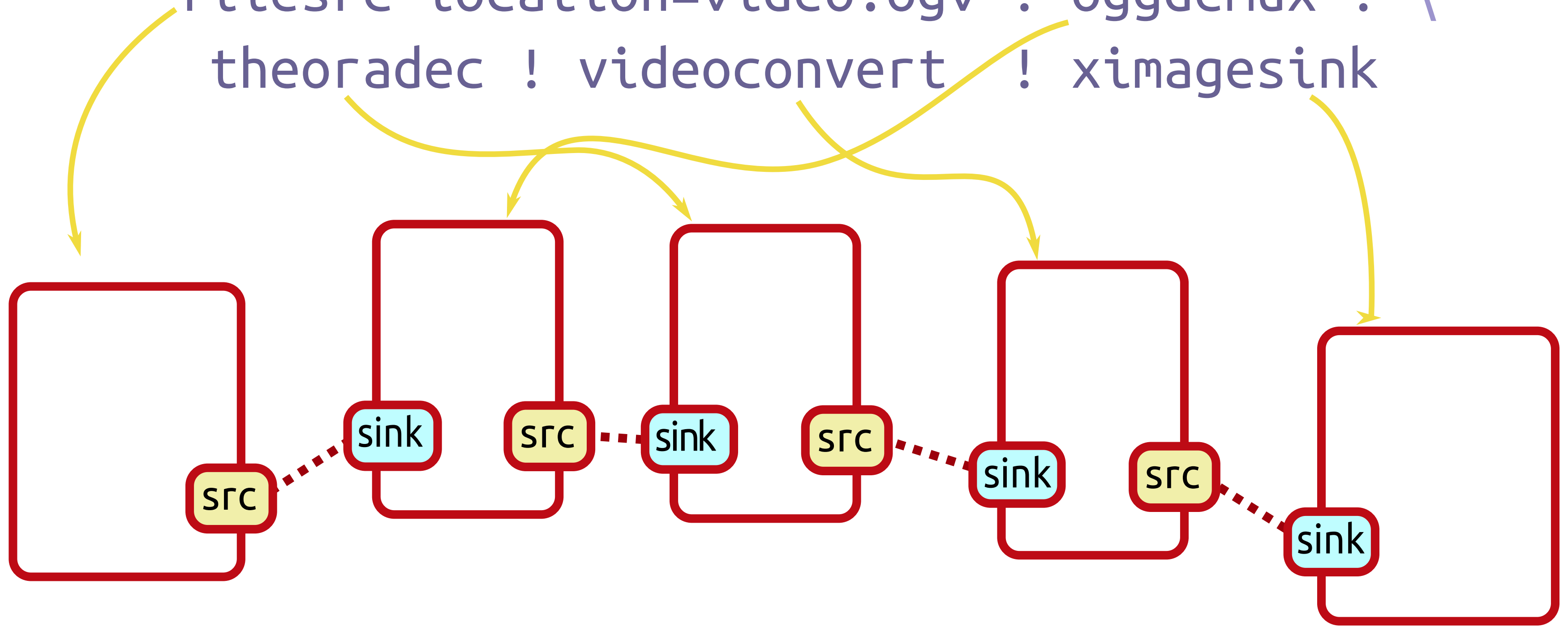
links are between pads




elements
sink
src
link
sink
sink
src
muxer,
mixer
sink
src
analyser
(emits messages)
src
source
(file, net,
capture, etc)
src
src
sink
src
demux,
tee
sink
sink
(file, net,
screen, etc)

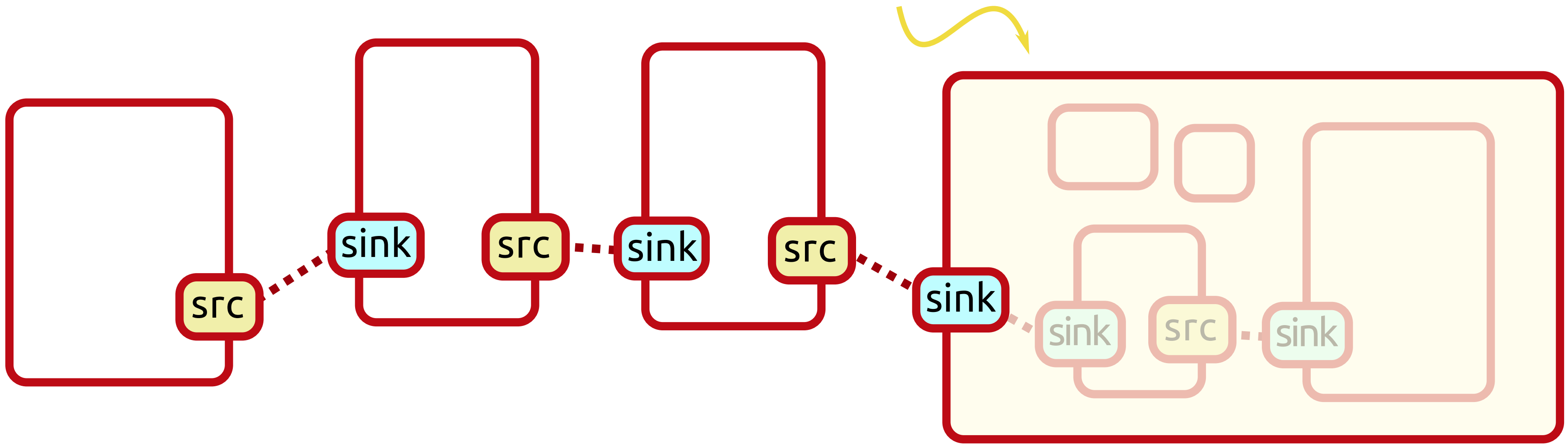
A simple pipeline

```
$ gst-launch-1.0  \
  filesrc location=video.ogv ! oggdemux ! \
   theoradec ! videoconvert  ! ximagesink
```



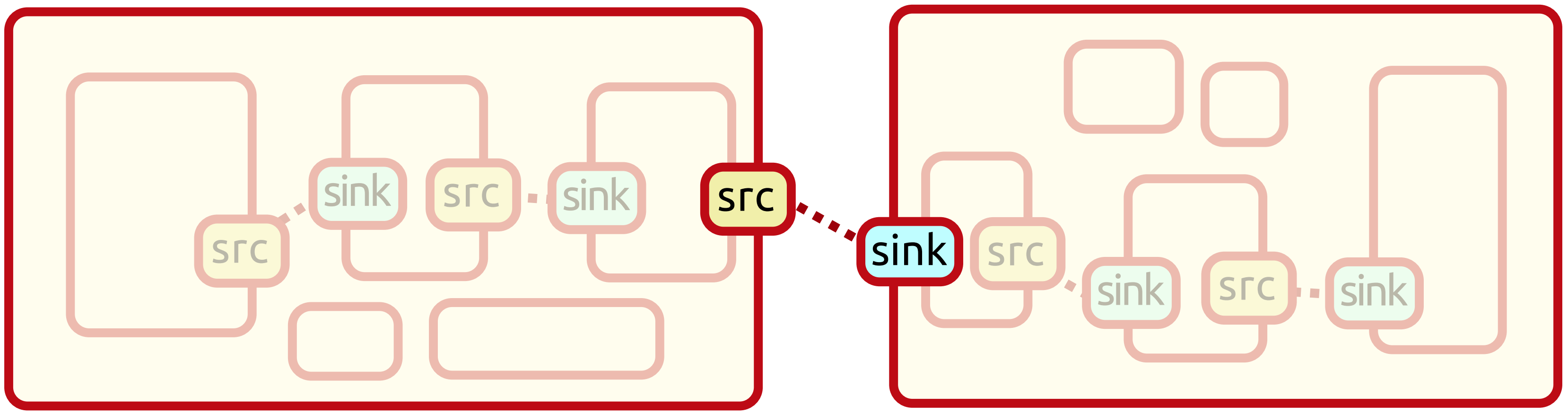
A simple pipeline containing a *bin*

```
$ gst-launch-1.0 \
  filesrc location=video.ogv ! oggdemux ! \
  theoradec ! autovideosink
```

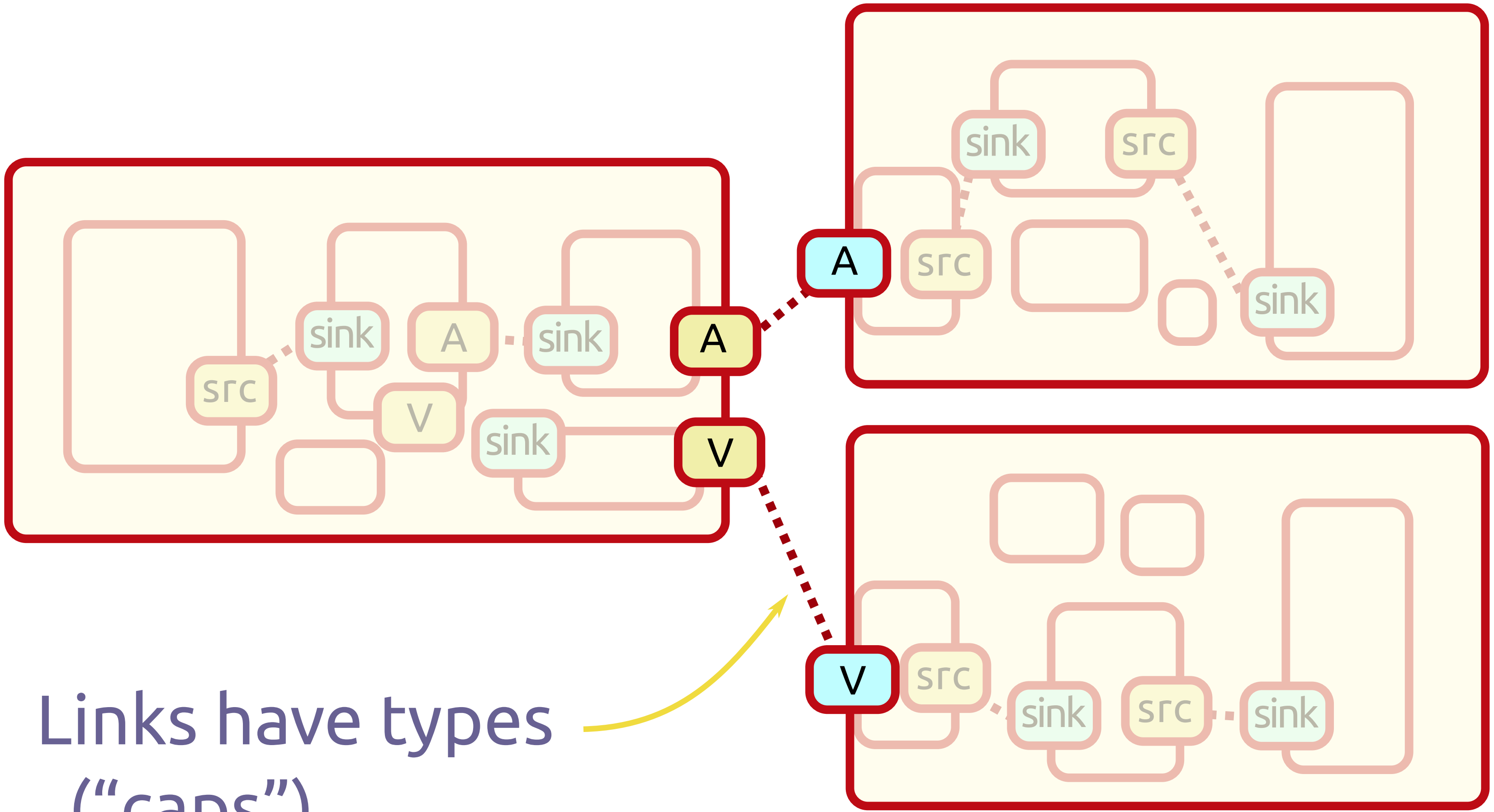


Two bins!

```
$ gst-launch-1.0 \
  uridecodebin uri=file://$PWD/video.ogv ! \
  autovideosink
```



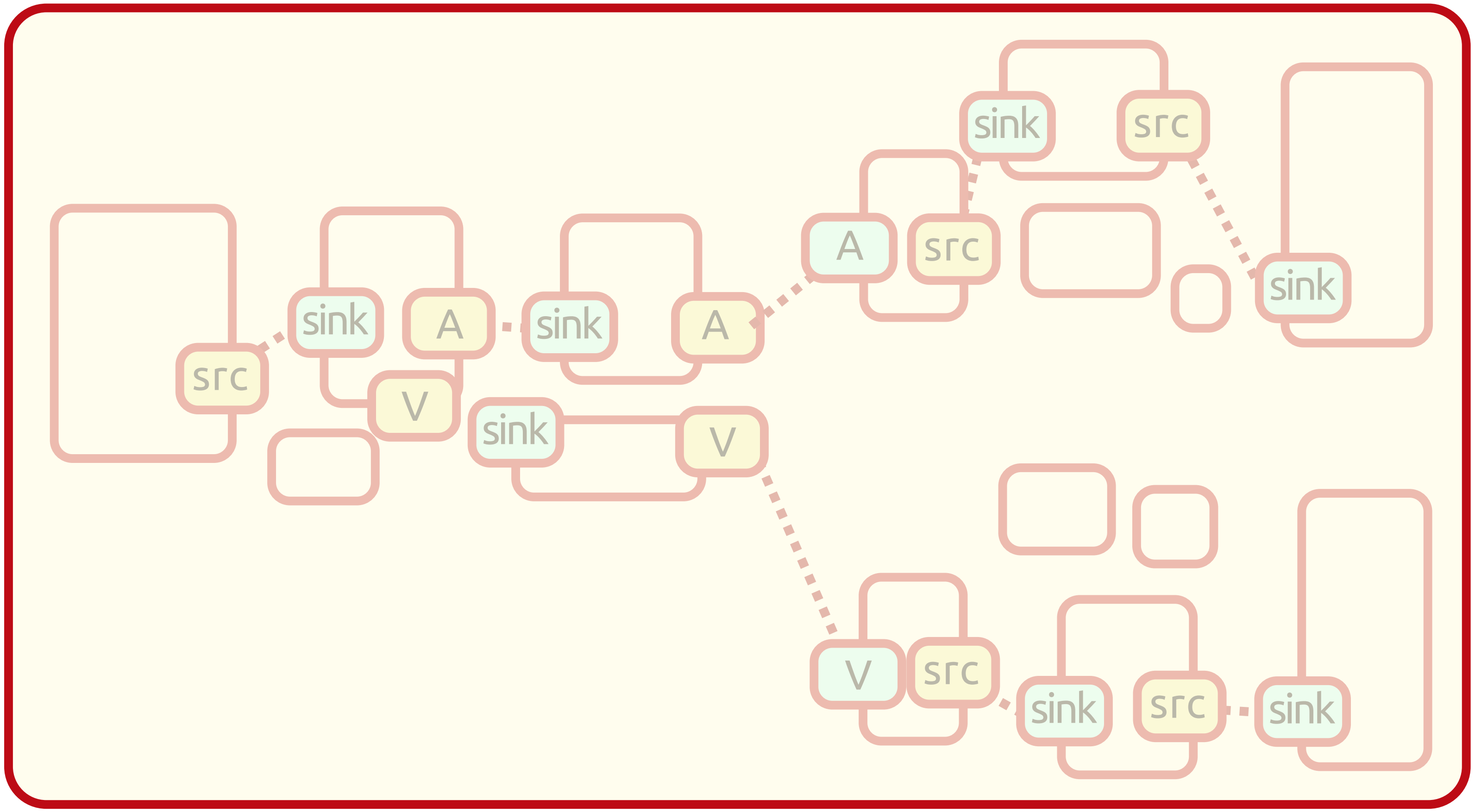
```
$ gst-launch-1.0 \
    uridecodebin uri=$whatever name=decoder \
    ! autovideosink  decoder. ! autoaudiosink
```



Links have types
("caps")

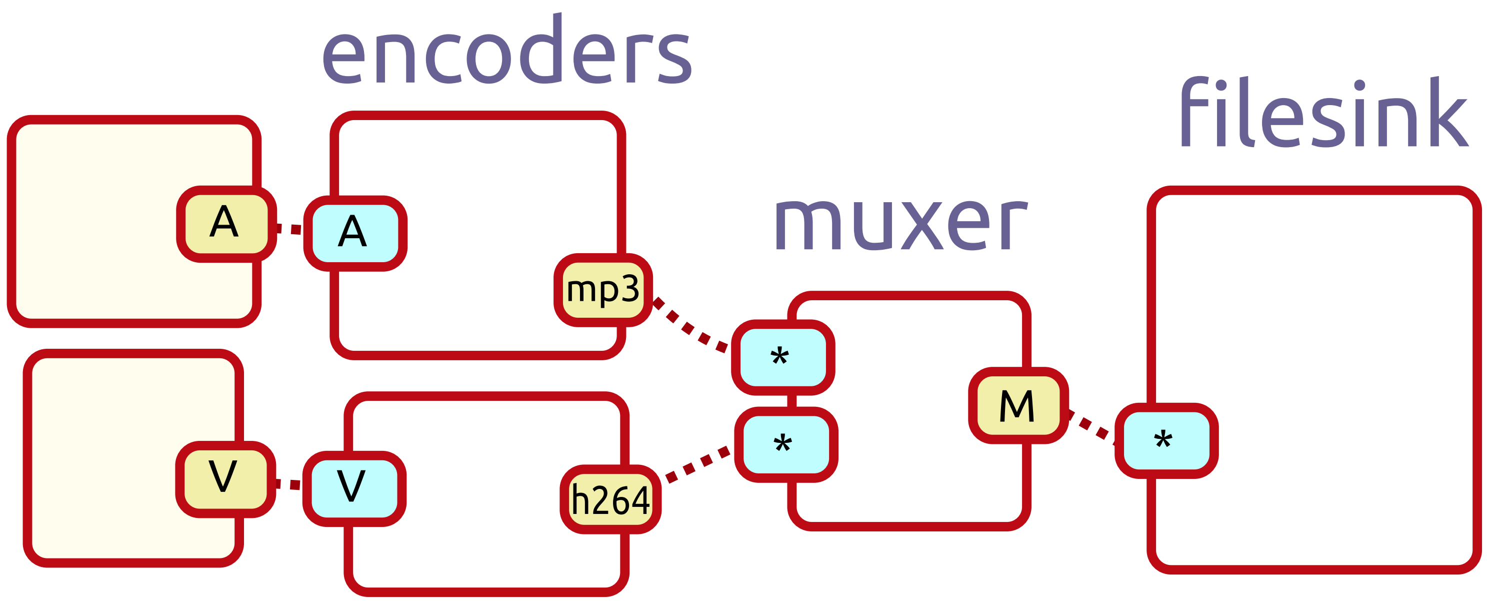
Playbin

```
$ gst-launch-1.0 \
    playbin uri=file://$PWD/video.ogv
```

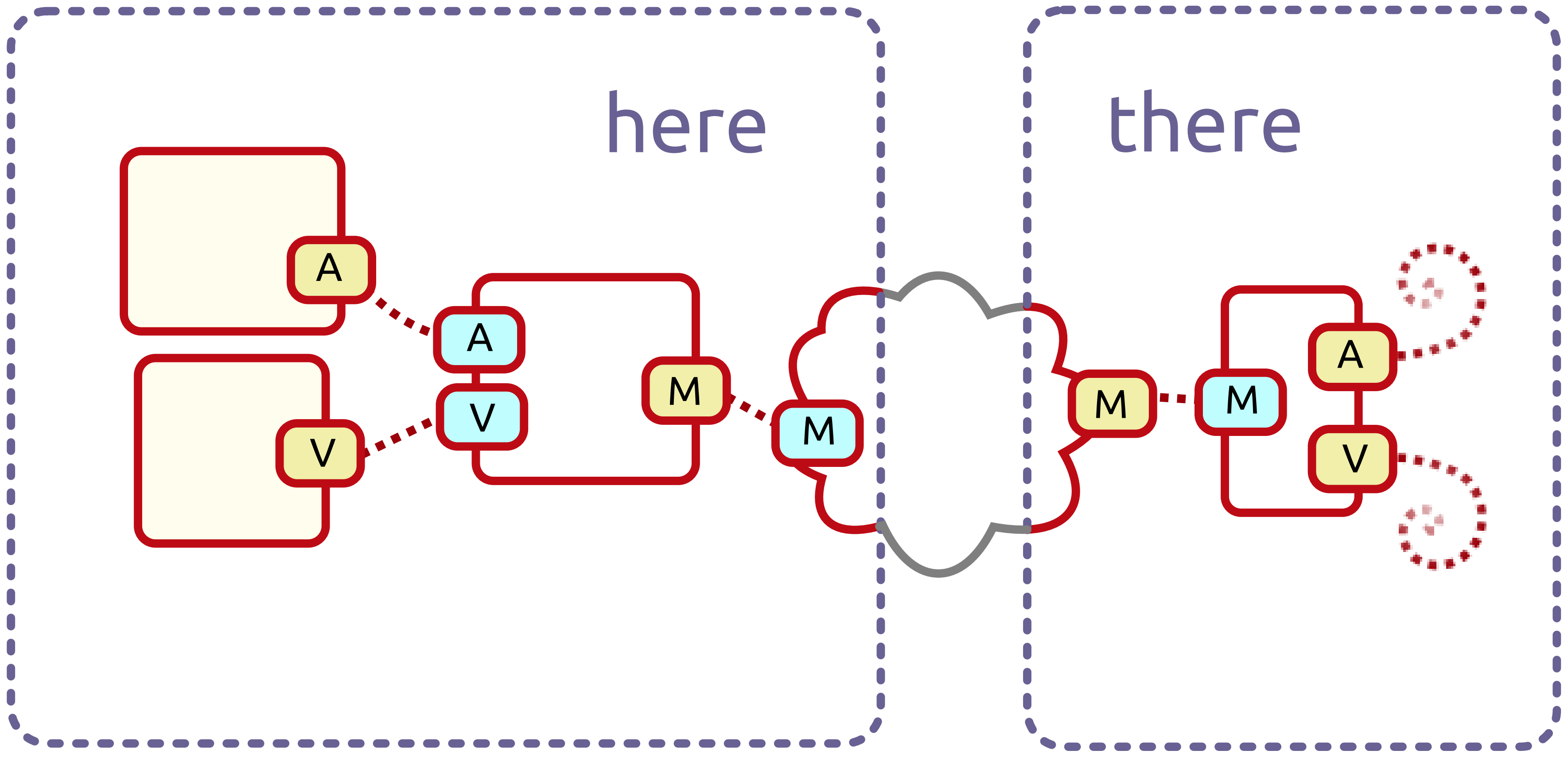


```
$ mplayer video.ogv
```

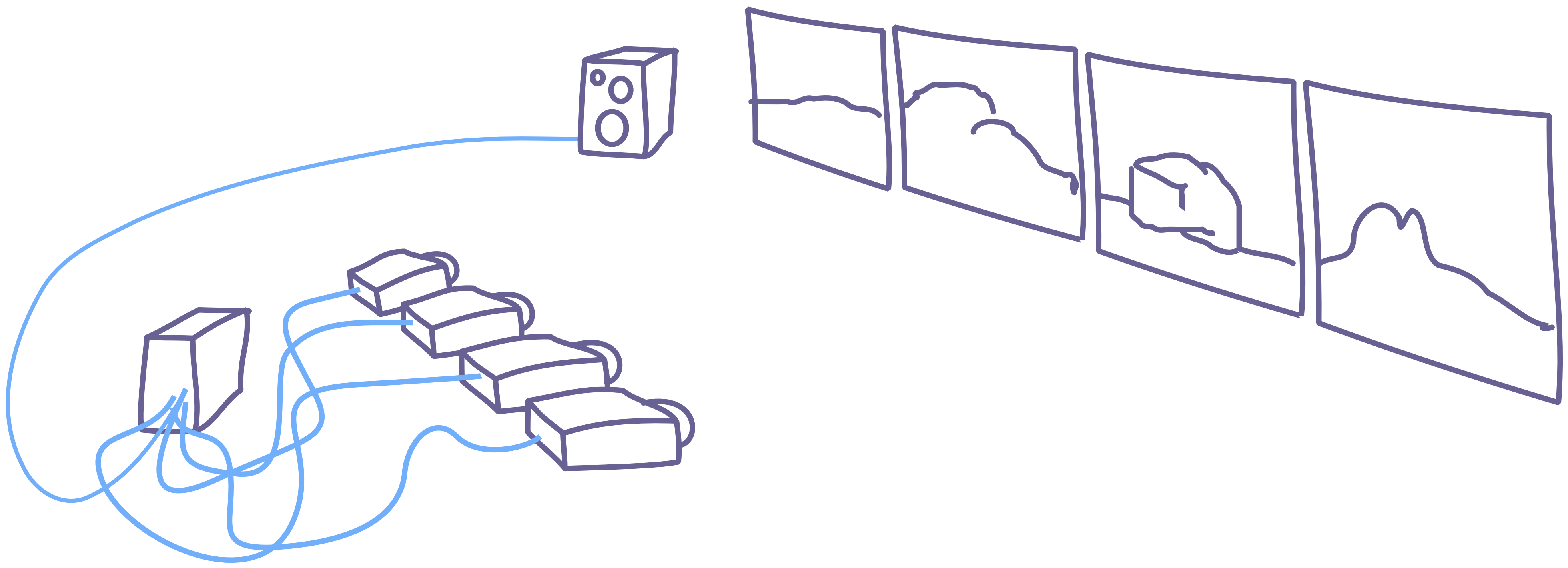
Combining streams



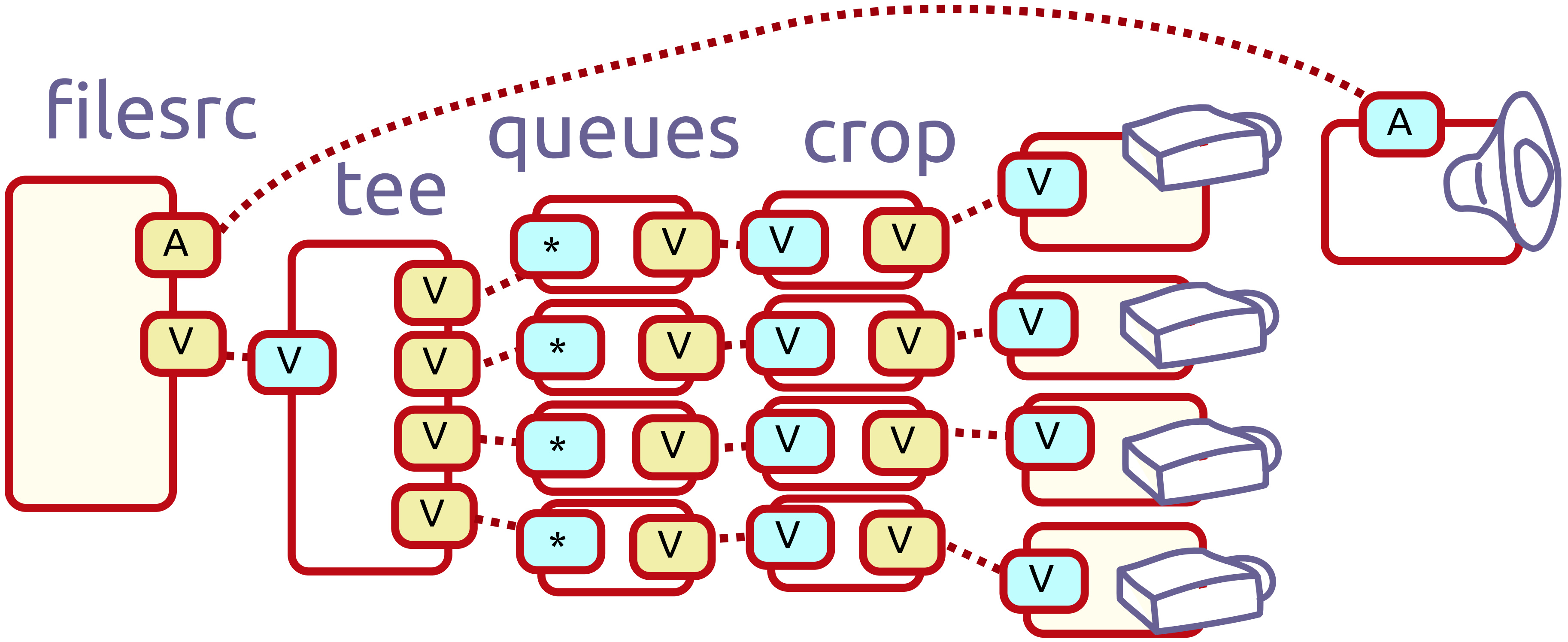
Network streaming



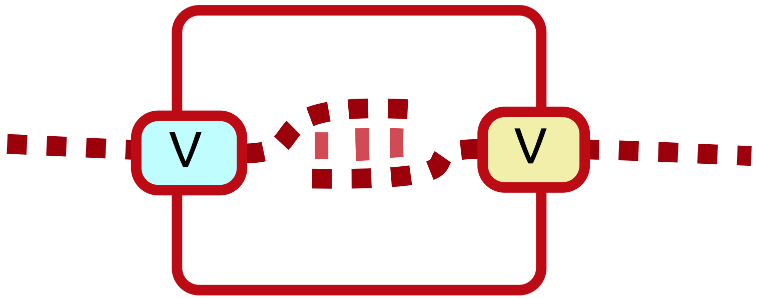
Video wall



Video wall

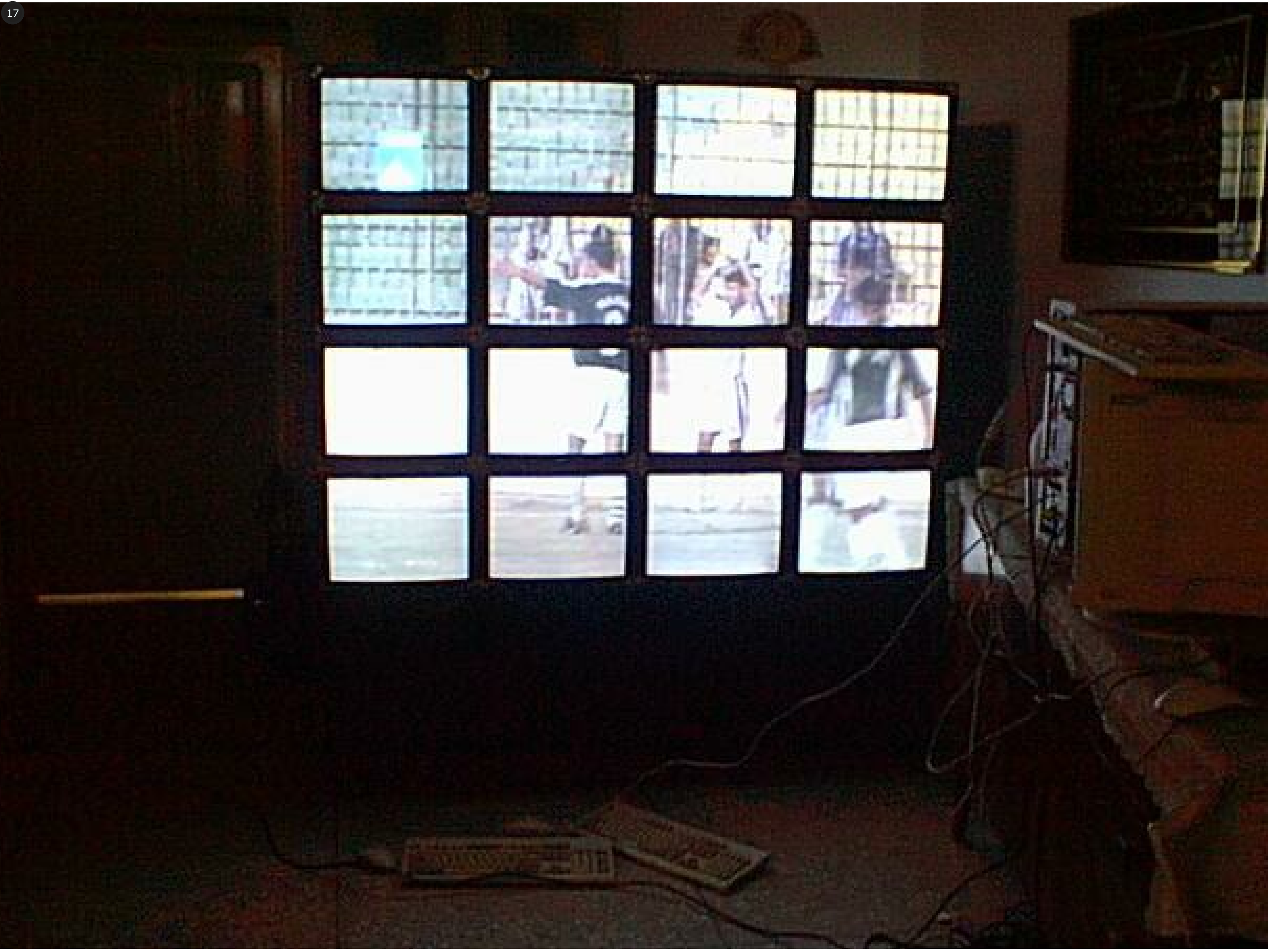


Queues decouple threads

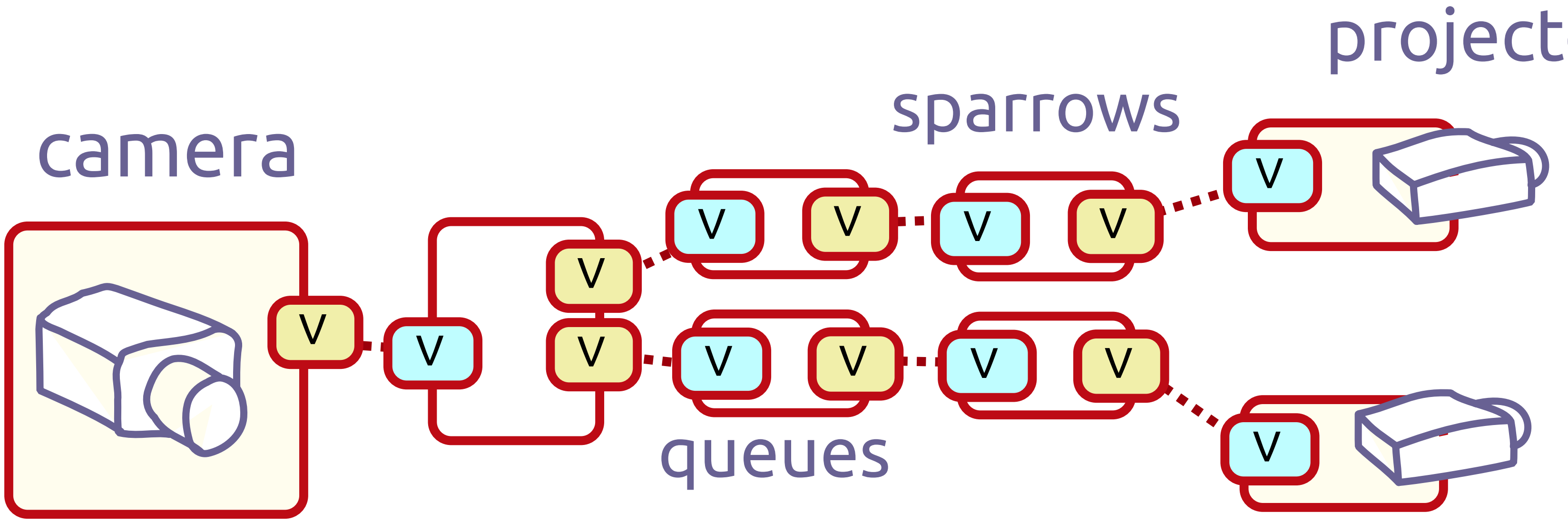


2002, Pakistan:
Video Whale
Zeeshan Ali Khattak,
et. al.

2010, NZ: Opo

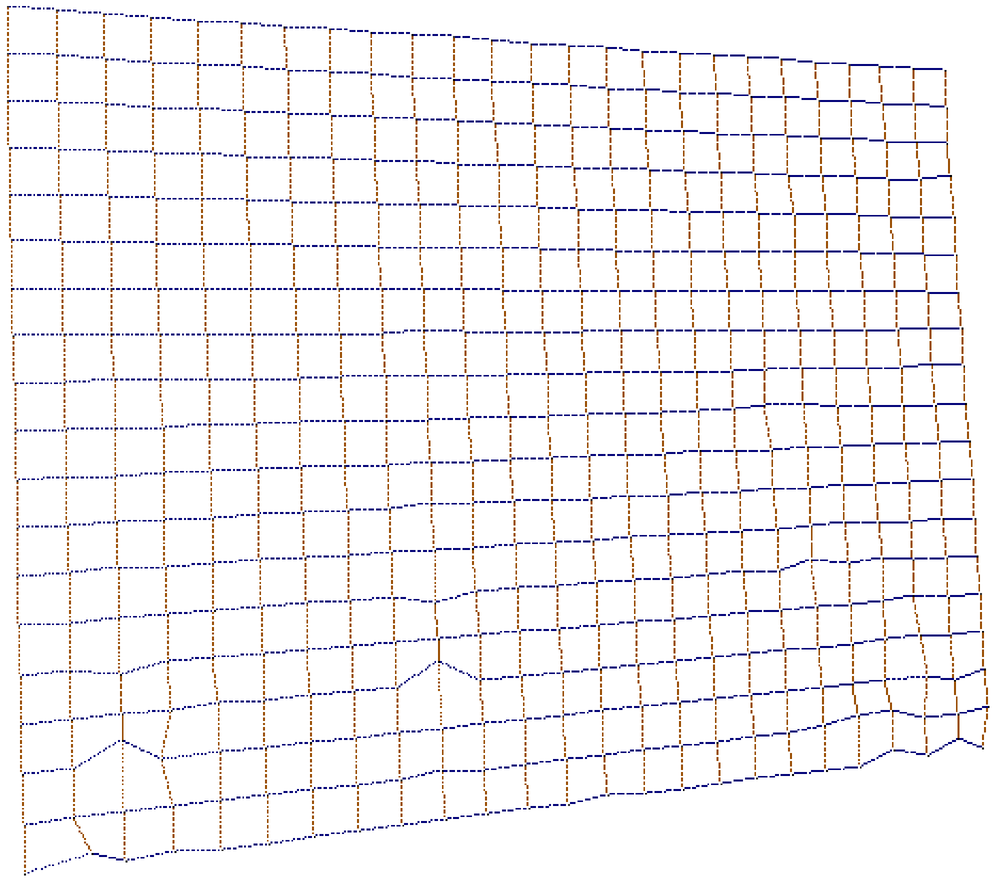


Tele Fast
PCO
PCO

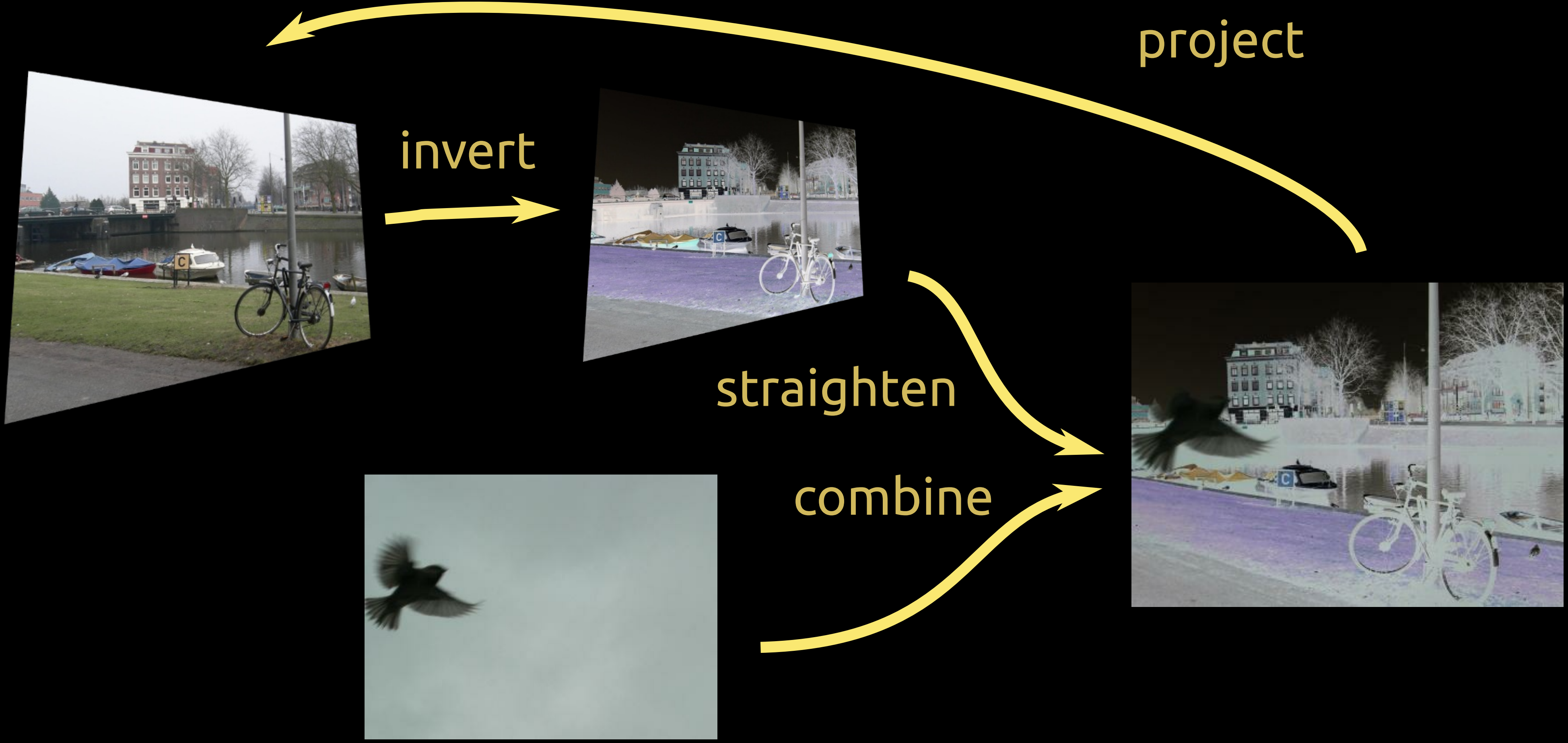

camera
sparrows
projectors
queues
V

Sparrow transform

1. map projection to camera



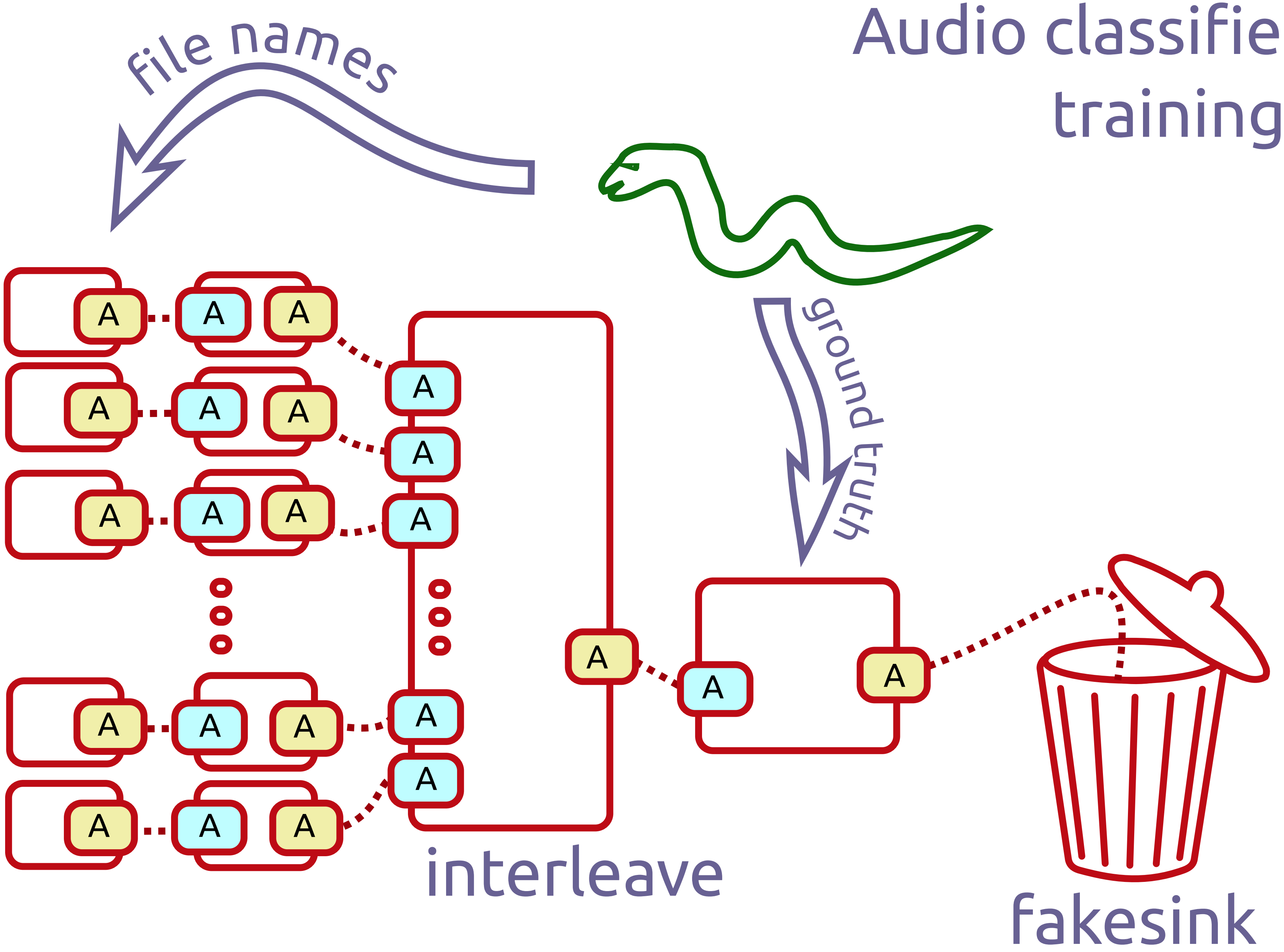
2. project inverse of camera input eradicating what is there replacing it with sparrow videos.



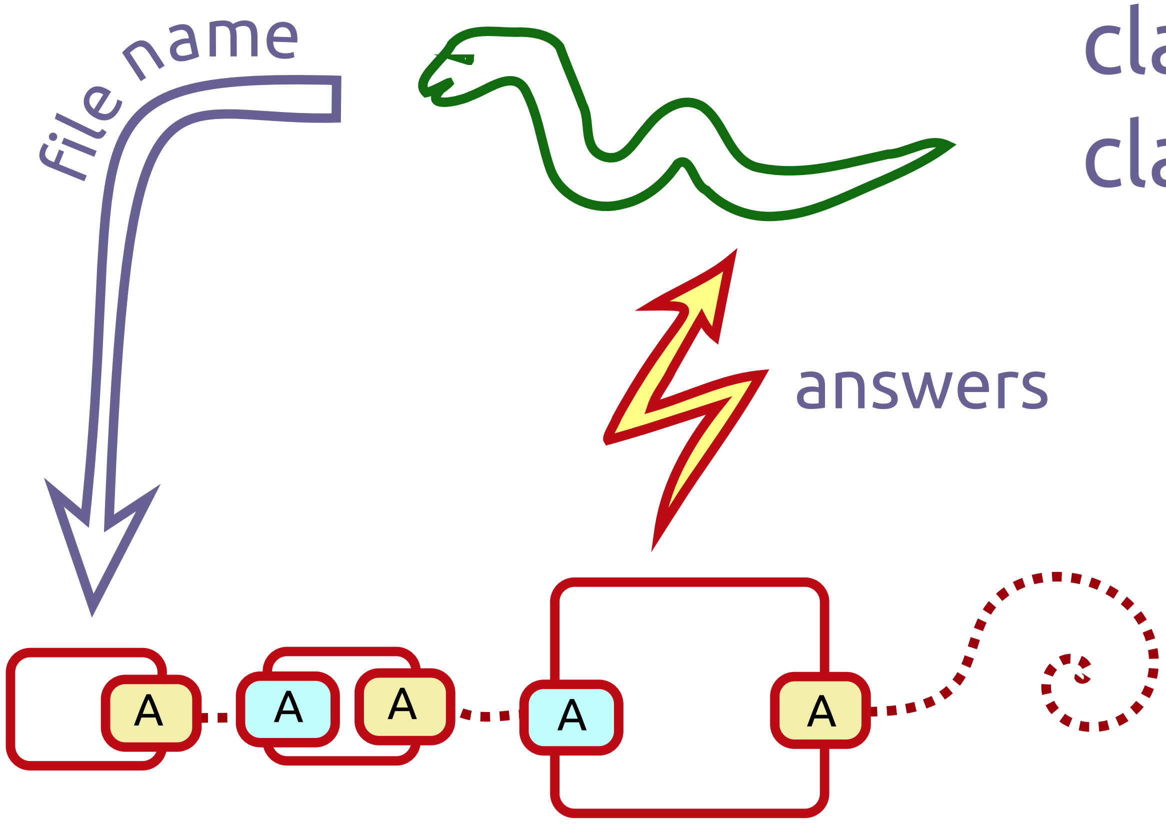
you can write elements in Python

(I don‘t know how)

Audio classifier training

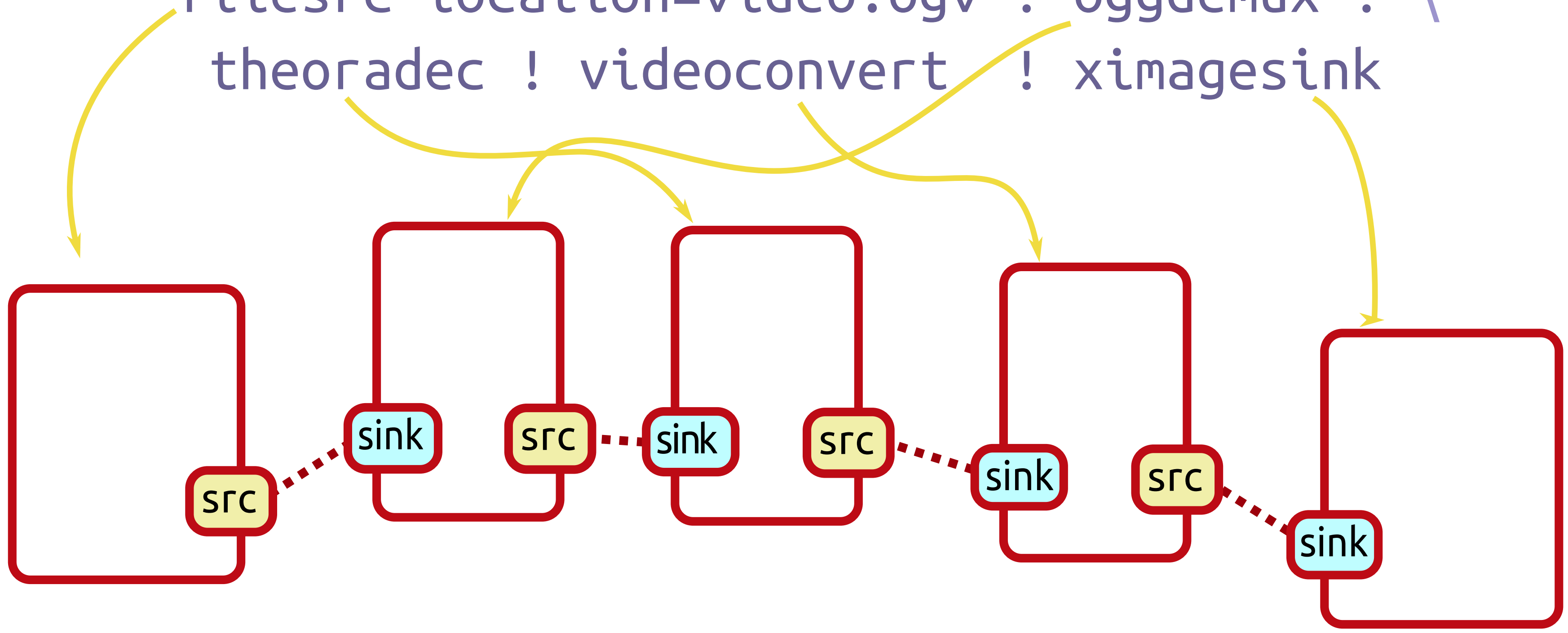


Audio classifier classifying



A simple pipeline

```
$ gst-launch-1.0  \
  filesrc location=video.ogv ! oggdemux ! \
   theoradec ! videoconvert  ! ximagesink
```

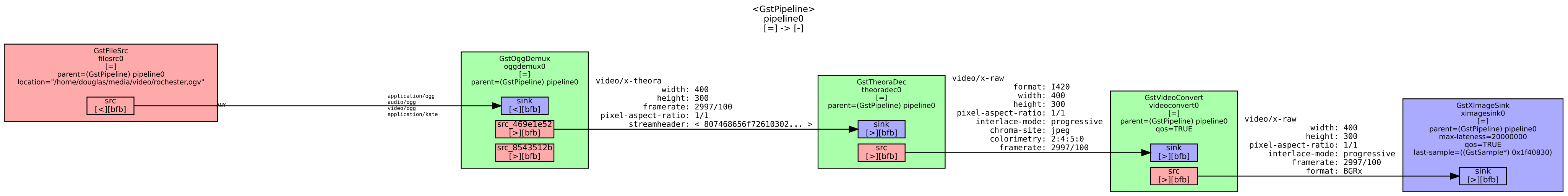


```
#!/usr/bin/python
import os
import gi
gi.require_version('Gst', '1.0') # sort of optional
from gi.repository import Gst, GObject


GObject.threads_init()
Gst.init(None)


class SimplePipeline(object):

    def __init__(self):
        self.pipeline = Gst.Pipeline()

        self.filesrc = Gst.ElementFactory.make('filesrc')
        self.oggdemux = Gst.ElementFactory.make('oggdemux')
        self.pipeline.add(self.filesrc)
        self.pipeline.add(self.oggdemux)
        self.pipeline.add(self.theoradec)
        self.pipeline.add(self.videoconvert)
        self.pipeline.add(self.ximagesink)

        self.filesrc.link(self.oggdemux)
        self.oggdemux.link(self.theoradec)
        self.theoradec.link(self.videoconvert)
        self.videoconvert.link(self.ximagesink)


loop = GObject.MainLoop()

p = SimplePipeline()
p.filesrc.set_property('location', 'video.ogv')
p.pipeline.set_state(Gst.State.PLAYING)
loop.run()
```

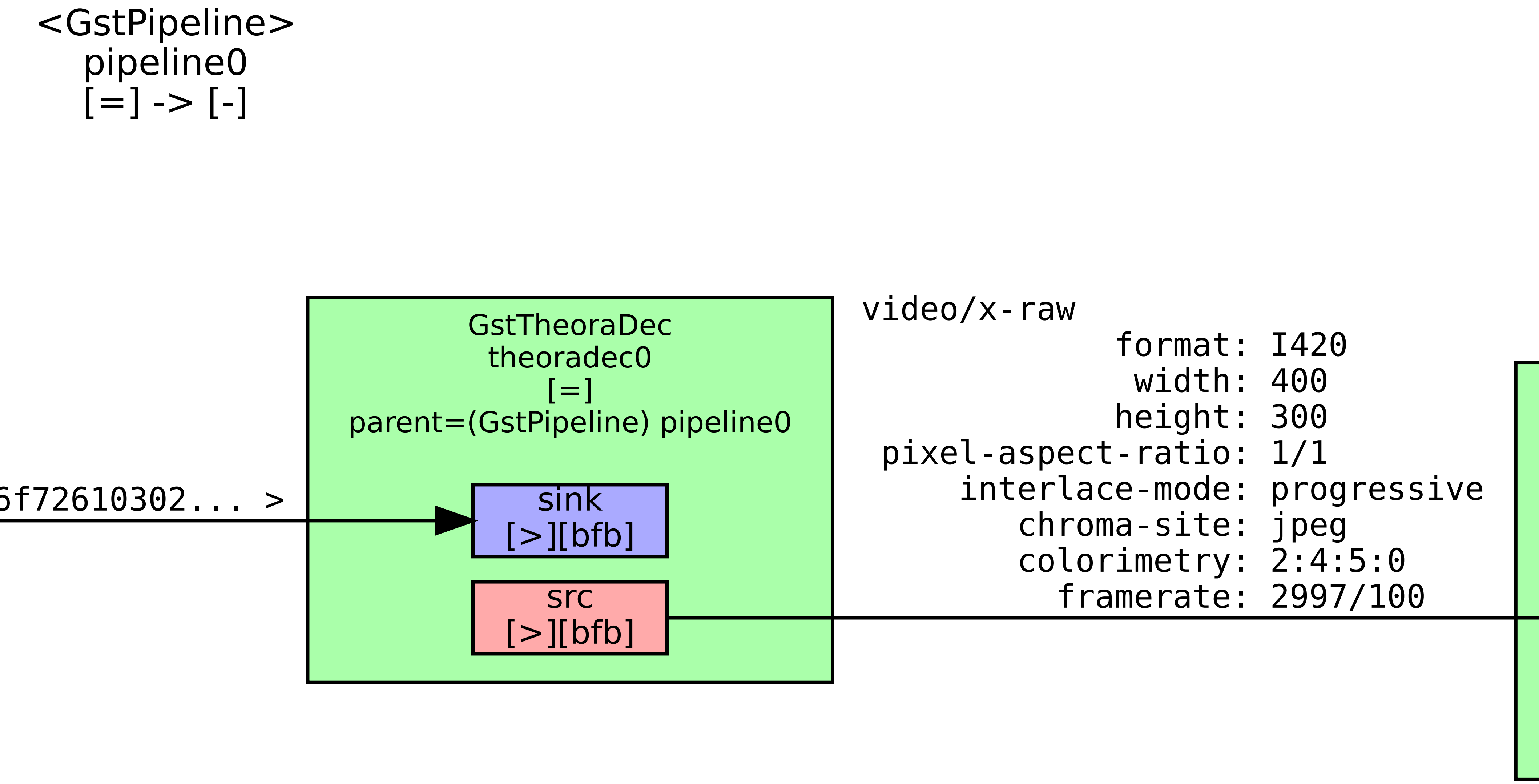
```python
class SimplePipeline2(object):
    def __init__(self):
        pipe_desc = ("filesrc name=src !"
                     "oggdemux ! theoradec ! "
                     "videoconvert ! ximagesink")

        self.pipeline = Gst.parse_launch(pipe_desc)
        self.filesrc = self.pipeline.get_by_name('src')
```

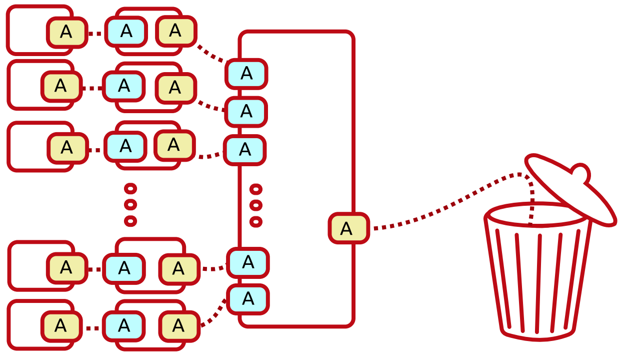
```python
class SimplePipeline3(object):
    def make_add_link(self, el, link=None, name=None):
        x = Gst.ElementFactory.make(el, name)
        self.pipeline.add(x)
        if link is not None:
            x.link(link)
        return x

    def __init__(self):
        self.pipeline = Gst.Pipeline()
        sink = self.make_add_link('ximagesink')
        videoconvert = self.make_add_link('videoconvert', sink)
        theoradec = self.make_add_link('theoradec', videoconvert)
        oggdemux = self.make_add_link('oggdemux', theoradec)
        self.filesrc = self.make_add_link('filesrc', oggdemux, 'src')
        self.filesrc.set_property('location', 'video.ogv')

        Gst.debug_bin_to_dot_file(self.pipeline, Gst.DebugGraphDetails.ALL,
                                  "pipeline.dot")
```

```
$ # make a graph in dot format
$ GST_DEBUG_DUMP_DOT_DIR=/tmp/ python simple-pipeline.py
```



```
$ # spew out a LOT of debug noise
$ GST_DEBUG=5 python simple-pipeline.py
```


<GstPipeline>
pipeline0
[=] -> [-]
GstTheoraDec
theoradec0
[=]
parent=(GstPipeline) pipeline0
sink
[>][bfb]
src
[>][bfb]
6f72610302... >
video/x-raw
format: I420
width: 400
height: 300
pixel-aspect-ratio: 1/1
interlace-mode: progressive
chroma-site: jpeg
colorimetry: 2:4:5:0
framerate: 2997/100



```
class UselessPipeline(object):
    def __init__(self, n_channels):
        self.pipeline = Gst.Pipeline()
        self.sink = self.make_add_link('fakesink', None)
        self.sources = []
        interleave = self.make_add_link('interleave', self.sink)

        for i in range(n_channels):
            capsfilter = self.make_add_link('capsfilter', self.interleave)
            capsfilter.set_property("caps",
                                    Gst.caps_from_string("audio/x-raw, "
                                                         "rate=16000, channels=1")
            converter = self.make_add_link('audioconvert', capsfilter)
            resampler = self.make_add_link('audioresample', converter)
            parser = self.make_add_link('wavparse', resampler)
            source = self.make_add_link('filesrc', parser)
            self.sources.append(source)
```

caps

```python
class TalkativePipeline(object):
    def __init__(self):
        pipe_desc = ("filesrc name=src !"
                     "oggdemux ! theoradec ! "
                     "videoconvert ! ximagesink")

        self.pipeline = Gst.parse_launch(pipe_desc)
        self.filesrc = self.pipeline.get_by_name('src')

        self.bus = self.pipeline.get_bus()
        self.bus.add_signal_watch()

        self.bus = self.pipeline.get_bus()
        self.bus.add_signal_watch()
        self.bus.connect("message", self.on_message)
        #self.bus.connect('message::eos', self.on_eos)
        #self.bus.connect('message::error', self.on_error)
        #self.bus.connect('message::element', self.on_element)

    def on_message(self, bus, msg):
        s = msg.get_structure()
        print(s.get_name())
        print(s.to_string())
```

```
D, pending-state=(GstState)GST_STATE_PLAYING;







ING, pending-state=(GstState)GST_STATE_VOID_PENDING;

ING, pending-state=(GstState)GST_STATE_VOID_PENDING;

ING, pending-state=(GstState)GST_STATE_VOID_PENDING;

ING, pending-state=(GstState)GST_STATE_VOID_PENDING;

ING, pending-state=(GstState)GST_STATE_VOID_PENDING;
```

gstreamer

http://gstreamer.freedesktop.org

Douglas Bagnall

douglas@halo.gen.nz

github: douglasbagnall