Programando Aplicações Multimídia no GStreamer

Guilherme Lima Rodrigo Costa Roberto Gerson

Pontifícia Universidade Católica – PUC-Rio
TeleMídia Lab., Dep. de Informática.

{*gflima,rodrigocosta,robertogerson*}*@telemidia.puc-rio.br*

6 de novembro de 2016

Quem somos

Guilherme Lima

- Doutor em Informática pela PUC-Rio (2015)
- Interesses de Pesquisa: linguagens de programação e modelos para sincronismo multimídia.

Rodrigo Costa

- Doutorando em Informática na PUC-Rio
- Interesses de Pesquisa: sistemas multimídia distribuídos e modelos síncronos para autoria multimídia.

Roberto Gerson

- Doutor em Informática pela PUC-Rio (2015)
- Interesses de Pesquisa: representação e autoria de cenas multimídia interativas e representação e renderização de vídeos 3D.

Acessando o Repositório do Minicurso

https://github.com/TeleMidia/minicurso-webmedia16

- Pasta *src*
  - códigos fonte utilizados neste minicurso
- Pasta *slides*
  - slides utilizados neste minicurso

Sumário

1. Introdução

2. Hands on

Sumário

1. Introdução

2. Hands on

GStreamer

- Um dos principais *frameworks* de código aberto para processamento de dados multimídia
  - Projeto com mais de 15 anos
- Projetado facilitar o desenvolvimento de aplicações que apresentam ou processam conteúdo audiovisual
- Softwares que usam o GStreamer:
  - amaroK
  - Banshee
  - Eina
  - Empathy
  - Rhythmbox
  - Totem

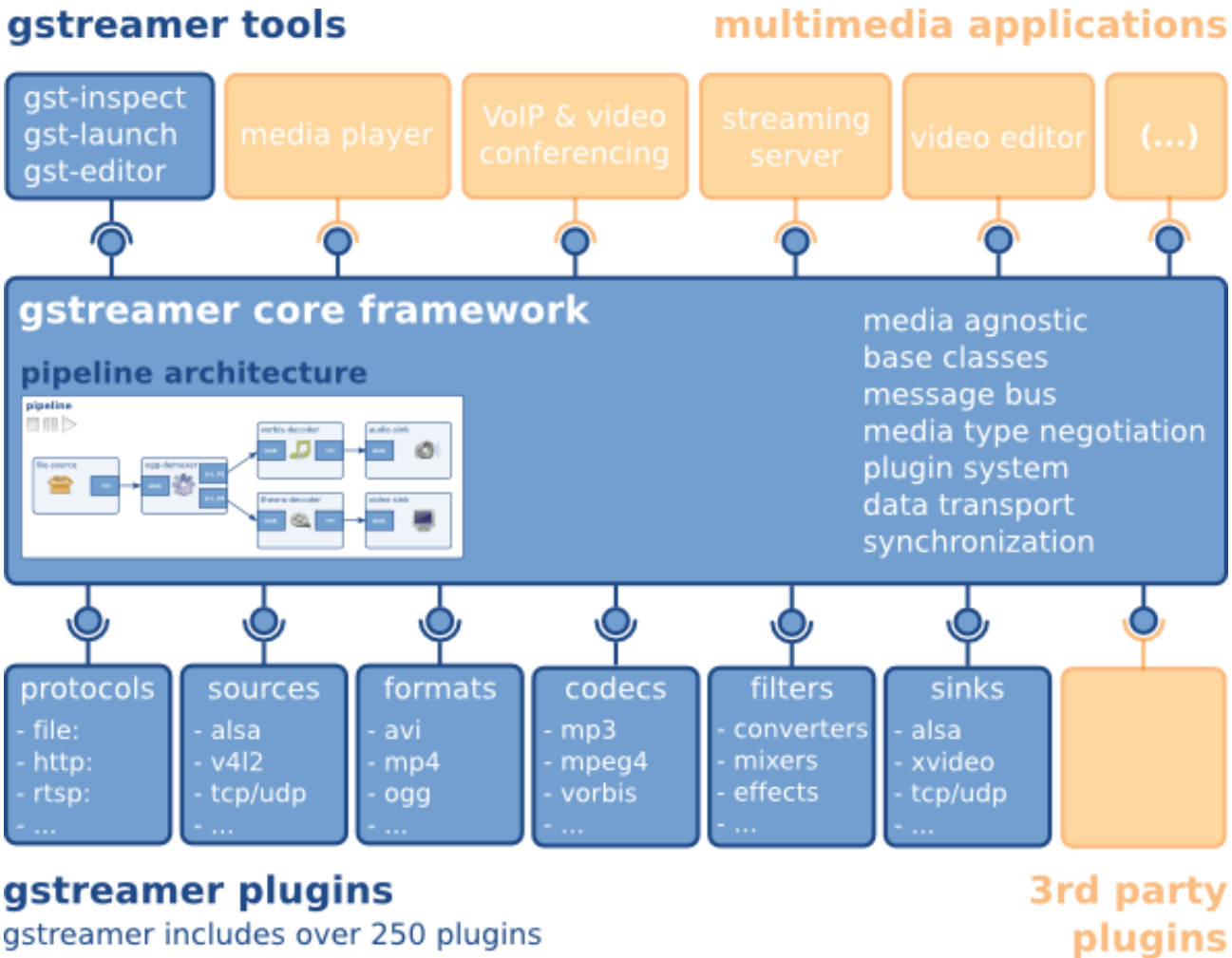
GStreamer

- Multiplataforma
- Robusto
- Flexível
- Extensível por Plugins
- APIs de alto e baixo nível
- Baseado no modelo de computação *dataflow*
  - *"Pipeline"* na terminologia do GStreamer
- Desenvolvido em C
  - Possui *bindings* para outras linguagens

O que o GStreamer NÃO é

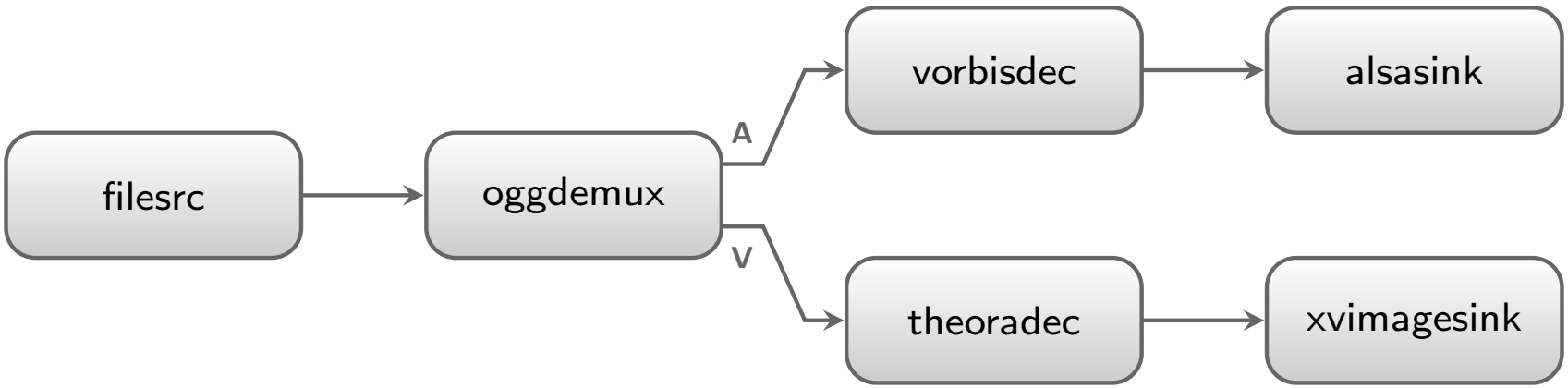
- Uma implementação de CODEC
- Uma aplicação *standalone*

GStreamer


gstreamer tools
multimedia applications
gst-inspect
gst-launch
gst-editor
media player
VoIP & video conferencing
streaming server
video editor
(...)
gstreamer core framework
pipeline architecture
media agnostic
base classes
message bus
media type negotiation
plugin system
data transport
synchronization
protocols
- file:
- http:
- rtsp:
- ...
sources
- alsa
- v4l2
- tcp/udp
- ...
formats
- avi
- mp4
- ogg
- ...
codecs
- mp3
- mpeg4
- vorbis
- ...
filters
- converters
- mixers
- effects
- ...
sinks
- alsa
- xvideo
- tcp/udp
- ...
gstreamer plugins
gstreamer includes over 250 plugins
3rd party plugins

Dataflow

- Dados são processados enquanto "fluem" através de uma rede
- Estrutura de grafo dirigido
  - Nós representam elementos de processamento (atores)
  - Arestas representam conexões unidirecionais por onde fluem os dados
- Atores recebem dados em portas de entrada e emitem dados através de portas de saída
- Um *pipeline* é um *dataflow* em que os dados fluem na mesma ordem em que foram produzidos

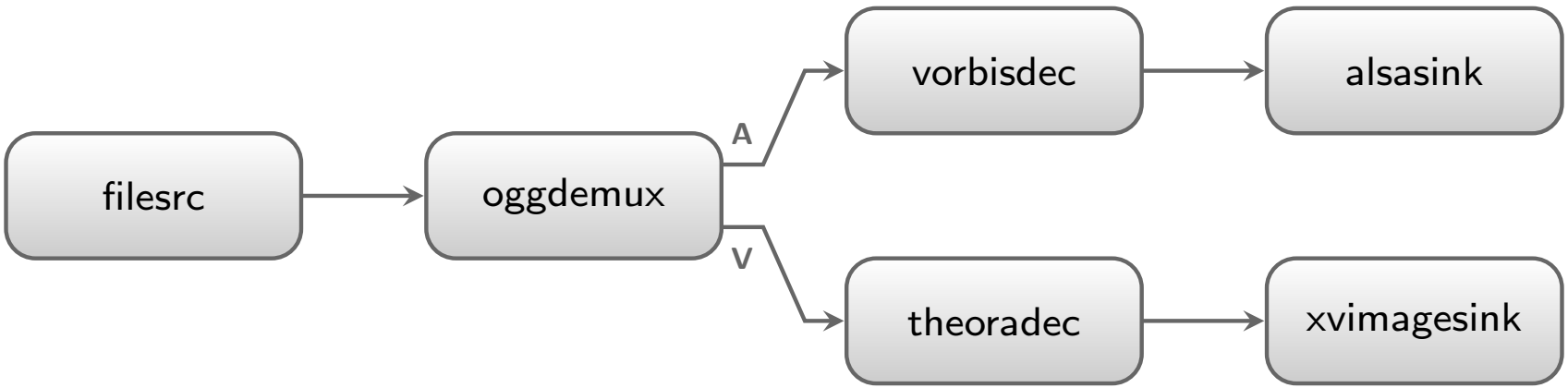


Pipelines no GStreamer

- Em um *pipeline* GStreamer os dados que fluem são tipicamente amostras de áudio e vídeo e dados de controle
- Nós do grafo (atores) são chamados elementos
- Portas por onde entram e saem dados dos elementos são chamados de *pads*
  - *Sink pad* – portas de entrada
  - *Source pad* – portas de saída
- Tipos de elementos
  - *Source* (produtores) – possuem apenas *source pads*
  - Processadores – possuem *source* e *sink pads*
  - *Sink* (consumidores) – possuem apenas *sink pads*

Pipelines no GStreamer

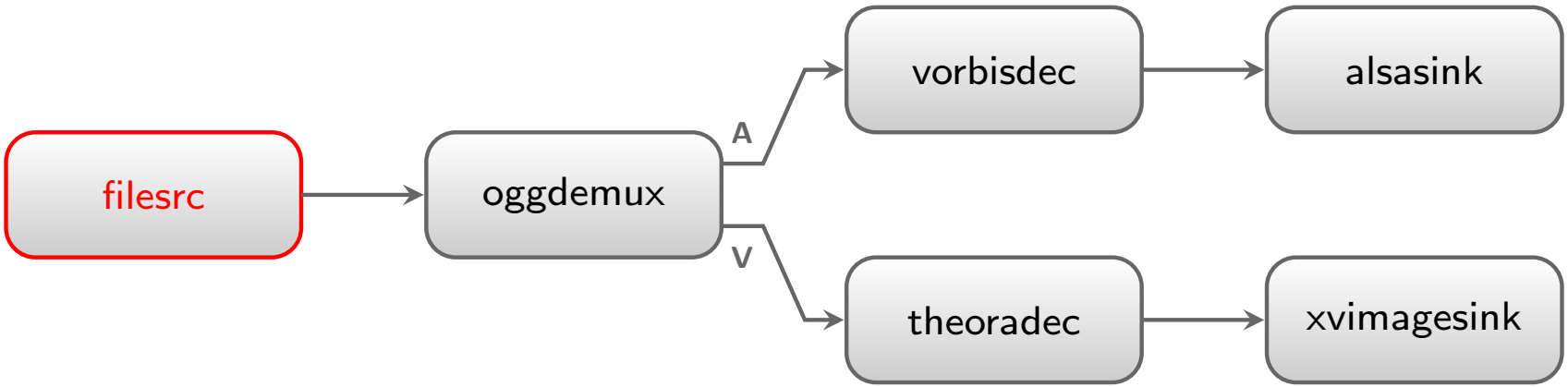
Um *pipeline* GStreamer simplificado que reproduz um arquivo de vídeo Ogg.



Pipelines no GStreamer

Um *pipeline* GStreamer simplificado que reproduz um arquivo de vídeo Ogg.

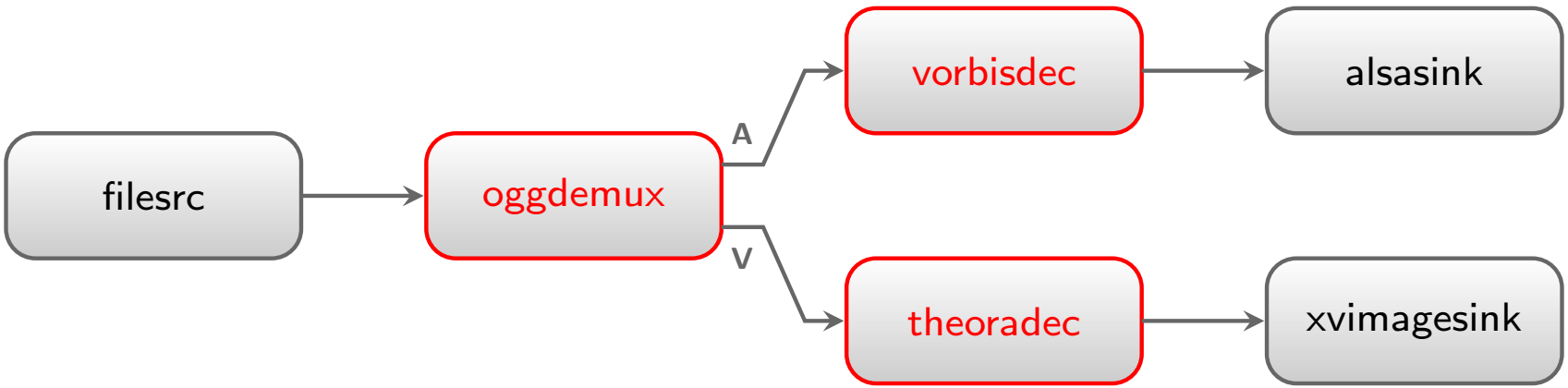
- *Source*
  - filesrc



Pipelines no GStreamer

Um *pipeline* GStreamer simplificado que reproduz um arquivo de vídeo Ogg.

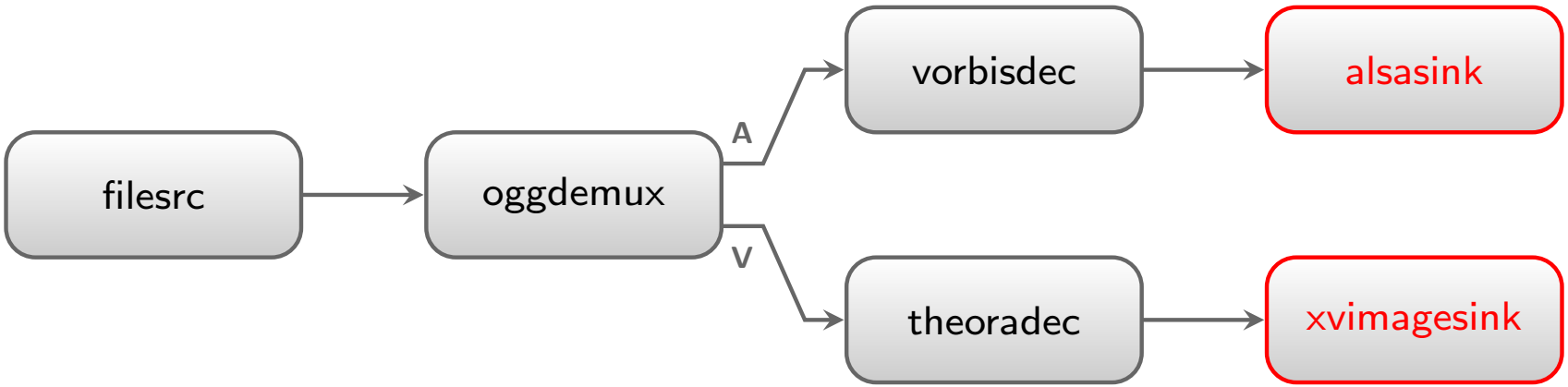
- Processadores
  - oggdemux
  - vorbisdec
  - theoradec



Pipelines no GStreamer

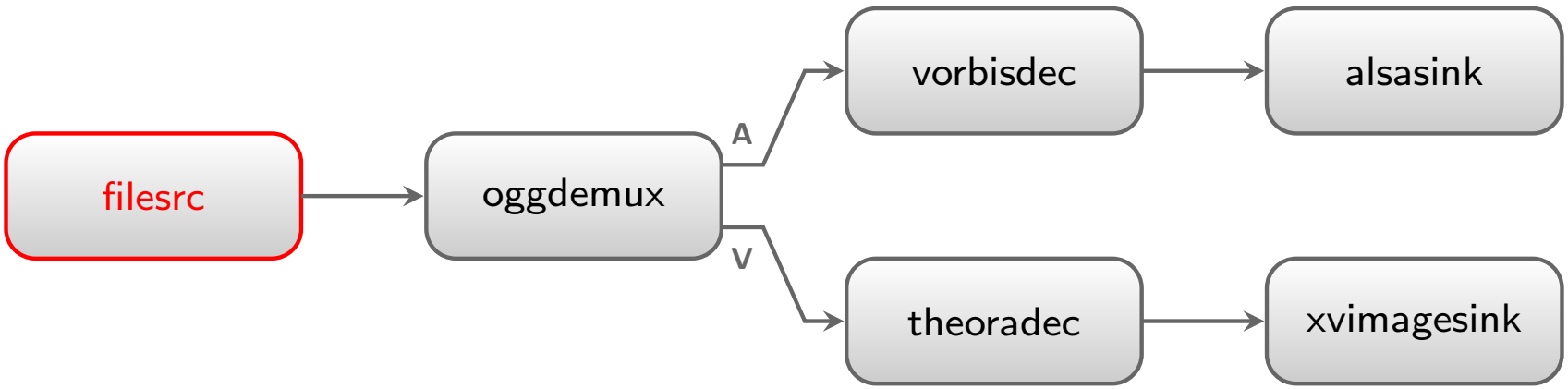
Um *pipeline* GStreamer simplificado que reproduz um arquivo de vídeo Ogg.

- *Sinks*
  - alsasink
  - xvimagesink



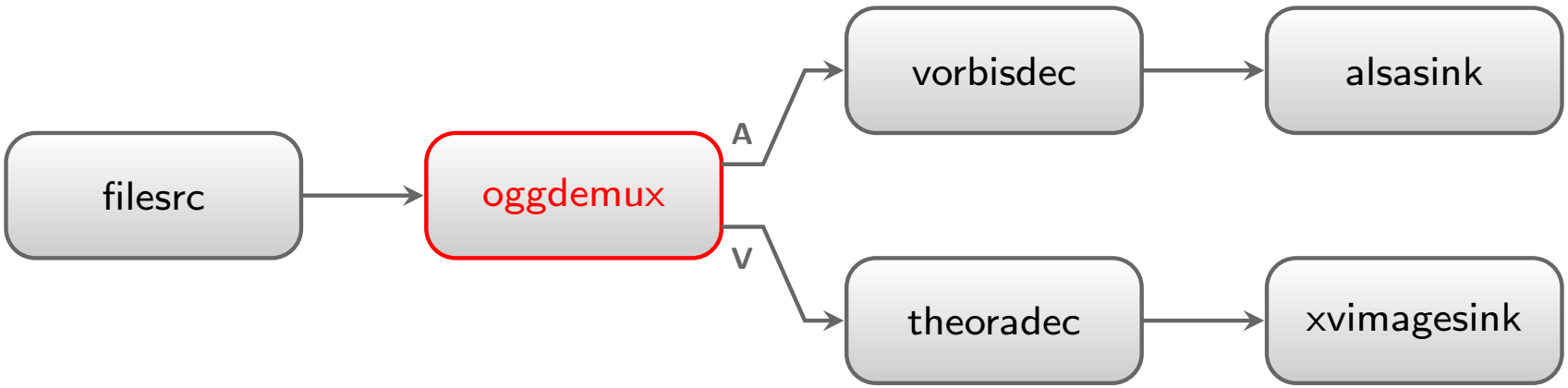
filesrc

Lê um arquivo do disco (vamos assumir um arquivo Ogg) e escreve o fluxo de bytes resultante na sua *source pad*



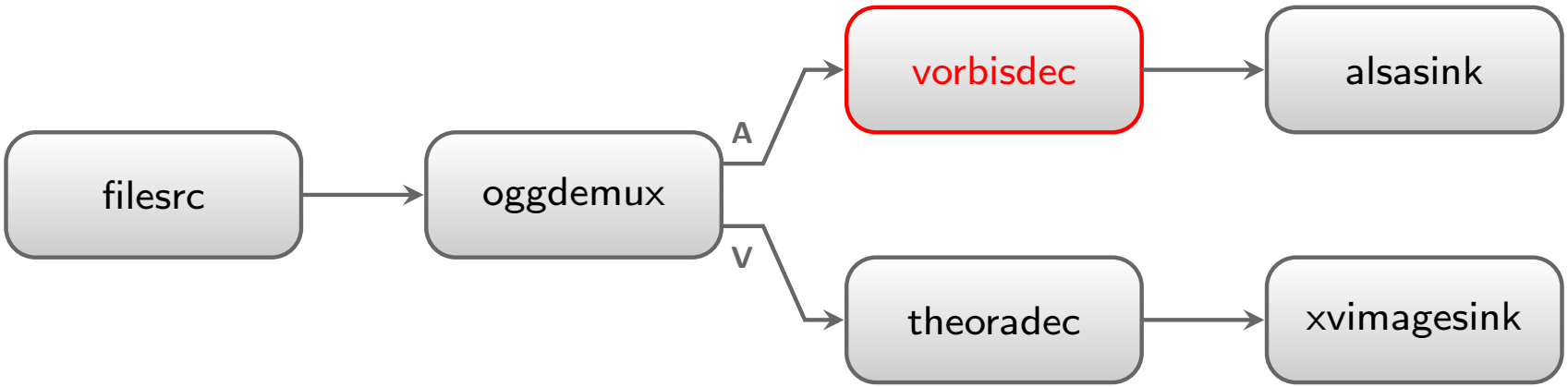
oggdemux

Lê da sua *sink pad* um fluxo de bytes codificado no formato Ogg, demultiplexa-o e escreve os fluxos Vorbis (áudio) e Theora (vídeo) resultantes nas *source pads* correspondentes



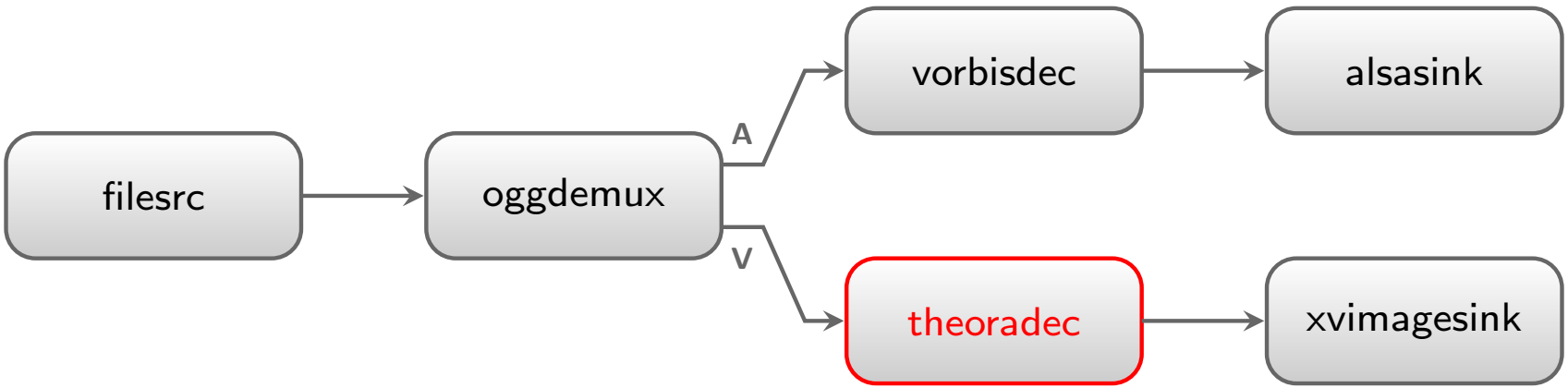
vorbisdec

Lê da sua *sink pad* um fluxo de bytes codificado no formato Vorbis, decodifica-o e escreve o fluxo de áudio PCM resultante (áudio *raw* descomprimido) na sua *source pad*



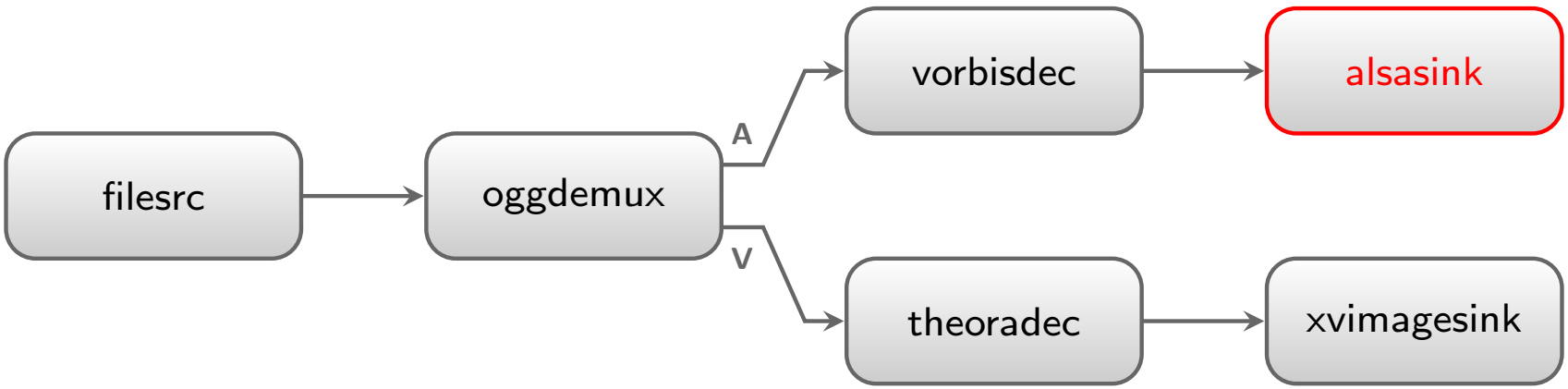
theoradec

Lê da sua *sink pad* um fluxo de bytes codificado no formato Theora, decodifica-o e escreve o fluxo de vídeo *raw* descomprimido resultante na sua *source pad*



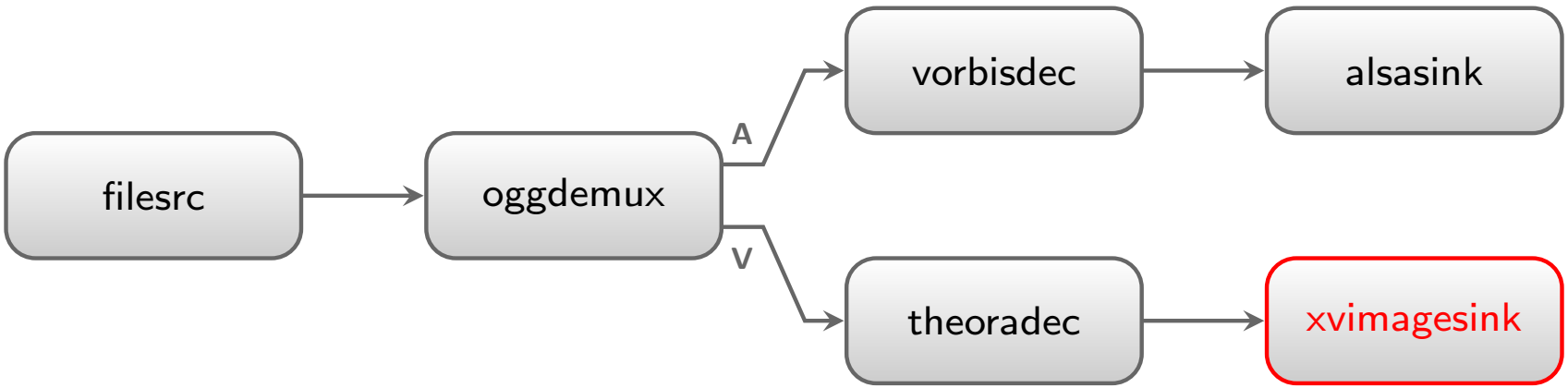
alsasink

Lê um fluxo de áudio descomprimido da sua *sink pad* e utiliza a biblioteca ALSA para reproduzir as amostras do fluxo nos alto-falantes



xvimagesink

Lê um fluxo de vídeo descomprimido da sua *sink pad* e utiliza a biblioteca X11 para reproduzir os quadros do fluxo na tela



Sincronização

- *Pipelines* possuem um relógio para controlar a sincronização dos fluxos
- Cada amostra de áudio e vídeo possuem um tempo de apresentação (PTS – *presentation timestamp*) e uma duração
- Elementos *sink* controlam a taxa de reprodução de cada fluxo
  - Amostras recebidas antes do seu tempo de apresentação são armazenadas em uma fila interna para serem exibidas no tempo adequado
  - Amostras recebidas após o seu tempo de apresentação são descartadas
- Todos os outros elementos do *pipeline* operam livremente
  - Consumem e produzem dados em taxas arbitrárias

Pads

- *Pads* são pontos de conexão entre elementos
  - source → sink ✓
  - sink → source ×
- Dois tipos de dados trafegam entre *pads*
  - Dados (*buffers*)
    - Amostras de áudio/vídeo
    - Fluem exclusivamente na direção das conexões
  - Eventos (*events*)
    - Informações de controle
    - Podem fluir em ambos os sentidos das conexões
    - Ex: QoS, seek, flush, ...
- Dados e eventos podem percorrer as conexões em paralelo

Estrutura conceitual de uma conexão entre *pads* no GStreamer.

Sumário

1. Introdução

2. Hands on

Primeira Aplicação: Olá mundo

Tocando um vídeo no GStreamer usando o elemento “playbin”.

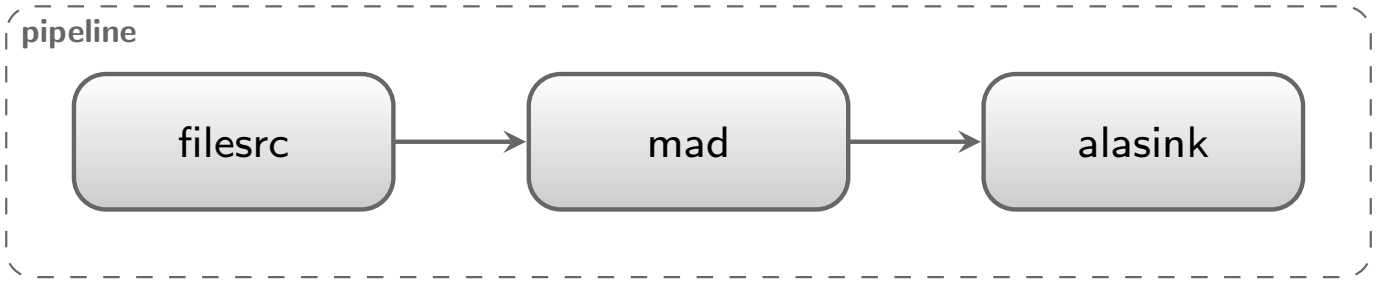
- Arquivo: src/hello.c

Compilando o código fonte hello.c:

```
$ cc hello.c -o hello `pkg-config --cflags --libs
      glib-2.0 gstreamer-1.0`
```

Tocando áudio e vídeo a partir de elementos básicos

Player MP3.



Pipeline que reproduz um arquivo de áudio MP3.

- Arquivo: src/mp3.c

Máquinas de Estado de um GstElement

- Todo elemento possui um estado, que pode ser:
  - GST_STATE_NULL (nulo)
  - GST_STATE_READY (pronto)
  - GST_STATE_PAUSED (pausado)
  - GST_STATE_PLAYING (tocando)
- Mudanças de estado são propagadas para elementos filhos (*bins*)
  - Direção de propagação é sempre dos *sinks* para os *sources*
  - Isso evita que dados sejam gerados antes que elementos posteriores no *pipeline* estejam prontos para recebê-los