Auxiliary Stream Wrangling in playbin3

Jan Schmidt @thaytan www.centricular.com

GStreamer Conference
October 24th 2025 | London | United Kingdom

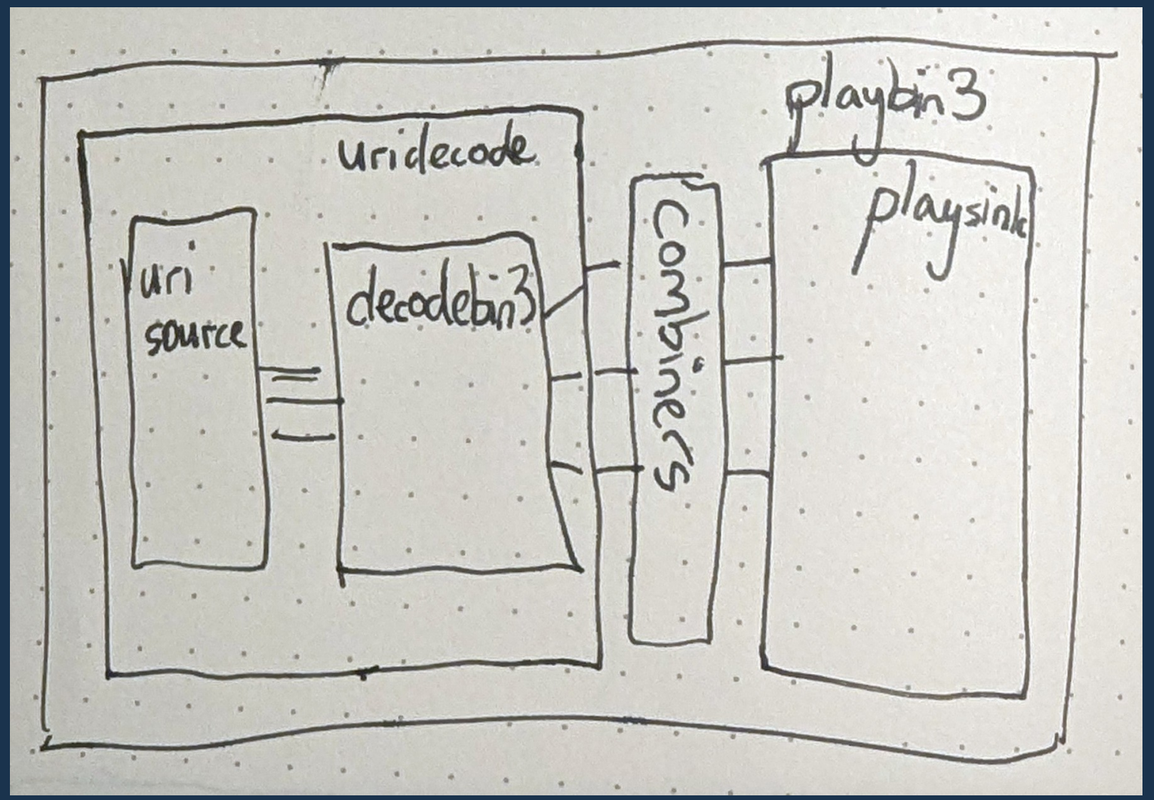
Centricular

A Brief History Of Playbin

- Simplifies media playback scenarios
  - One bin does everything from URI to display
- **playbin (2004)**: monolithic internals and logic
- **playbin2 (2007)**: Improved modularity / stream switching and playsink
- **playbin3 (2016)**: GstStream API support

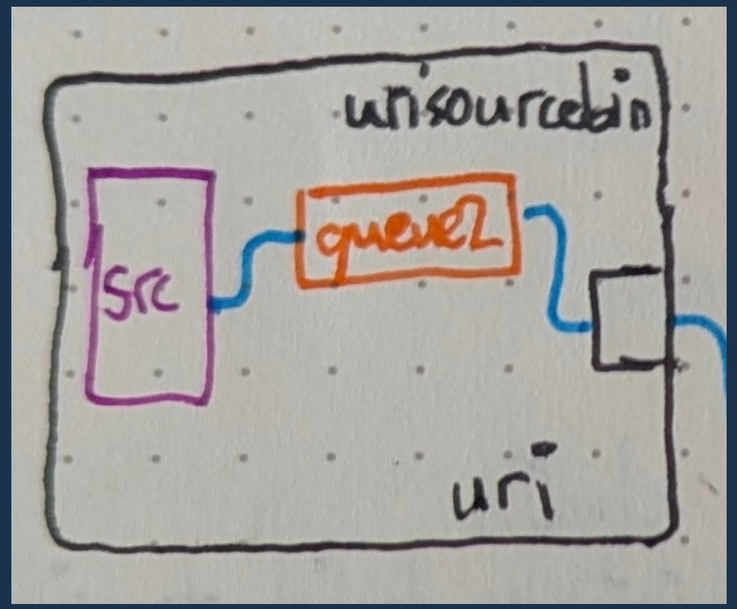
Playbin3

- **urisourcebin**: URI loading & source pad creation.
- **decodebin3**: Demuxing & decoding.
- **playsink**: Sinks, overlays, visualisation
- **Combiners**: optional stream mixing



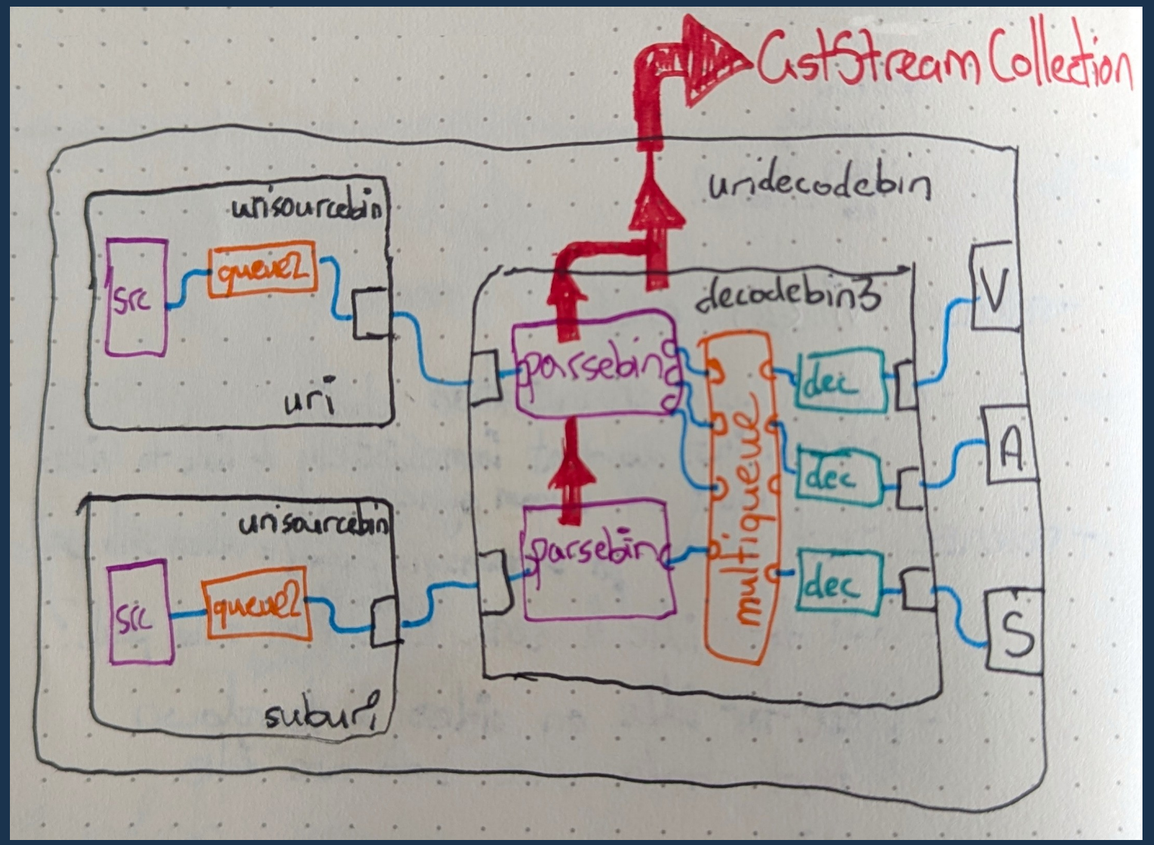
Urisourcebin

- Handling URIs, determining content type, creating source pads.
- Handles one uri, and buffering, traditionally



Existing Auxiliary Stream Support

- Uridecodebin **suburi** can plug 1 extra urisourcebin
- Designed for subtitle file
- Can/has been abused for (e.g.) an audio track instead



Centricular

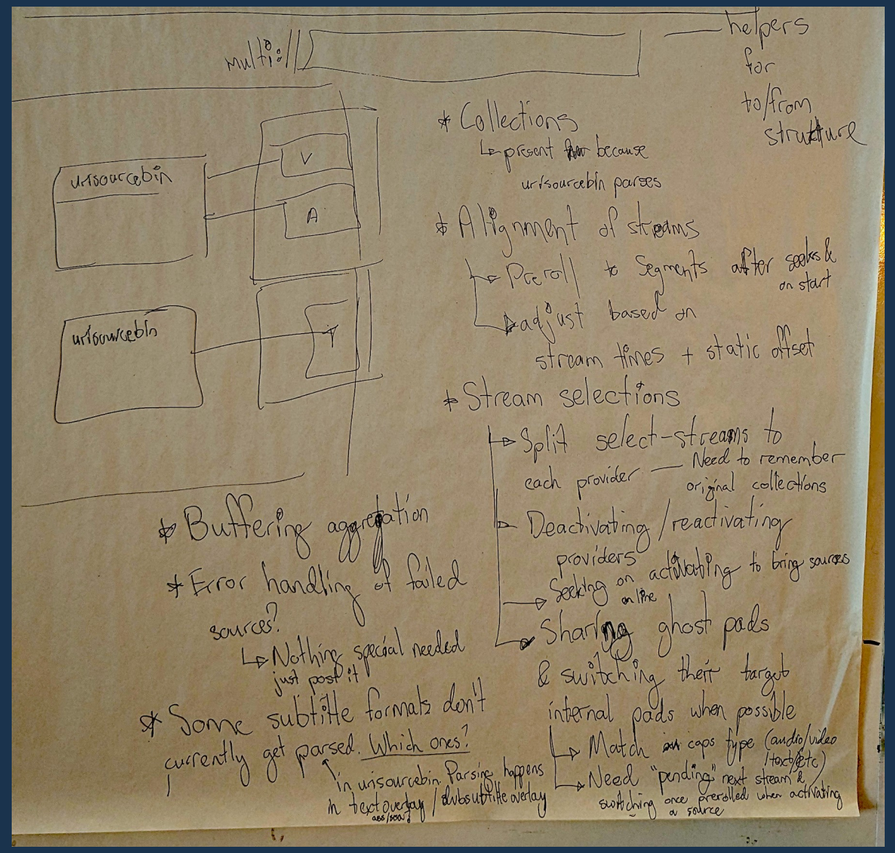
Stream Collection API

- Pipelines publish **GstStreamCollection**
- Applications send **GstSelectStreams** events
- Handler in the pipeline does selections
  - Demuxers (adaptive), or **decodebin3**

```
Processing new collection from <hlsdemux2-0>: collection 0x7fc57008b090 (7 streams) <
stream text 0x7fc58c008ee0, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/text-Chinese, flags 0x1, caps [], tags [taglist,
language-code=(string)zho;],
stream text 0x7fc58c008f90, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/text-French, flags 0x1, caps [], tags [taglist,
language-code=(string)fra;],
stream audio 0x7fc58c046830, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/audio-English, flags 0x2, caps [], tags [taglist,
language-code=(string)en;], stream audio 0x7fc58c04e9e0, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/audio-Spanish, flags
0x0, caps [], tags [taglist, language-name=(string)sp;],
stream audio 0x7fc58c050c10, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/audio-Commentary (eng), flags 0x0, caps [], tags
[taglist, language-code=(string)en;],
stream video 0x7fc570088470, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/main-video-0, flags 0x2, caps [video/x-h264], tags
[],
stream audio 0x7fc570089a40, ID d2ea67bf9ca757905cc9114d42461d10e7998c196457ec3ded02077e275728d6/main-audio-1, flags 0x2, caps [], tags [], >
```

Multiple Auxilliary Streams

- Idea:
  - Combining multiple sources into a single presentation
  - Assigning language tags
  - Timestamp offsets
  - Buffering aggregation



Centricular

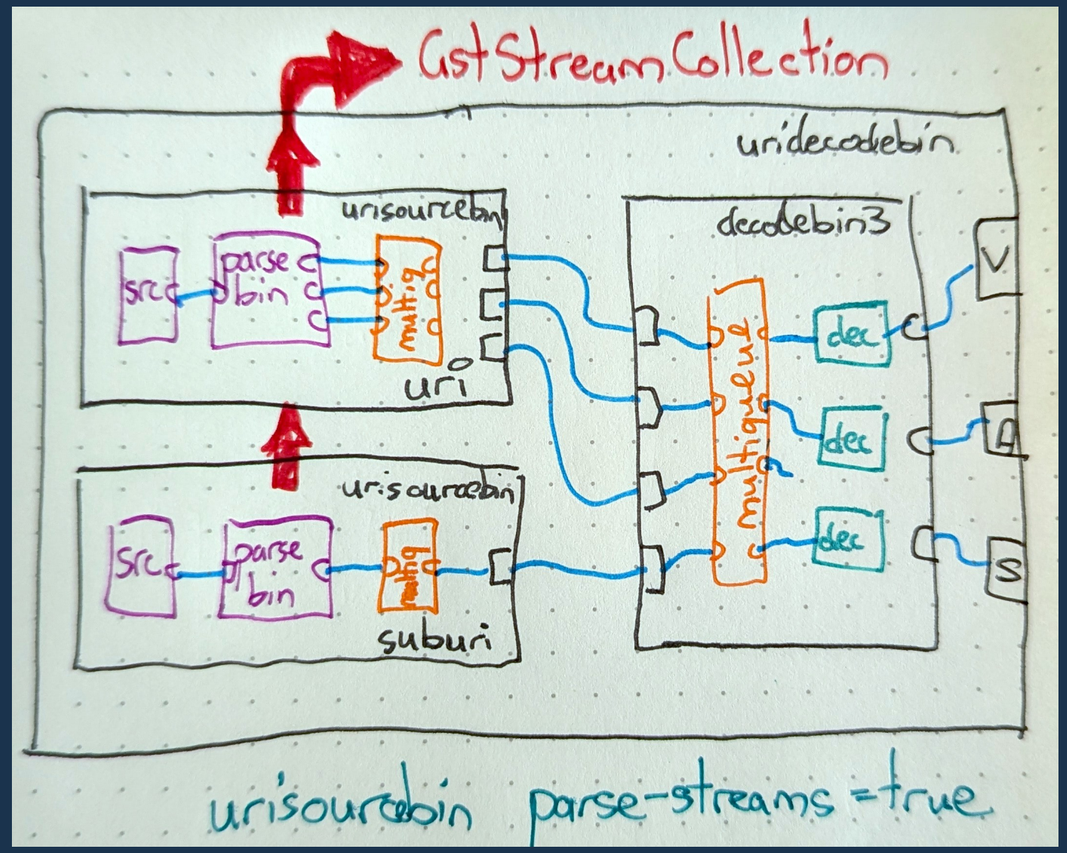
Multiuri://

- Describe the combination of inputs in a single stream
- API for creating + serialisation / deserialisation

```
multiuri://<(structure)"uri,uri=(string)https%3A%2F%2Fcdn.theoplayer.com%2Fvideo%2Felephants-
dream%2Fplaylist.m3u8;",
(structure)"uri,uri=(string)https%3A%2F%2Fgstreamer.freedesktop.org%2Fmedia%2Fsmall%2Ftest_sub.srt,
offset=(gint64)5000000000,language-tag=eng" >
```

Urisourcebin + Parsing

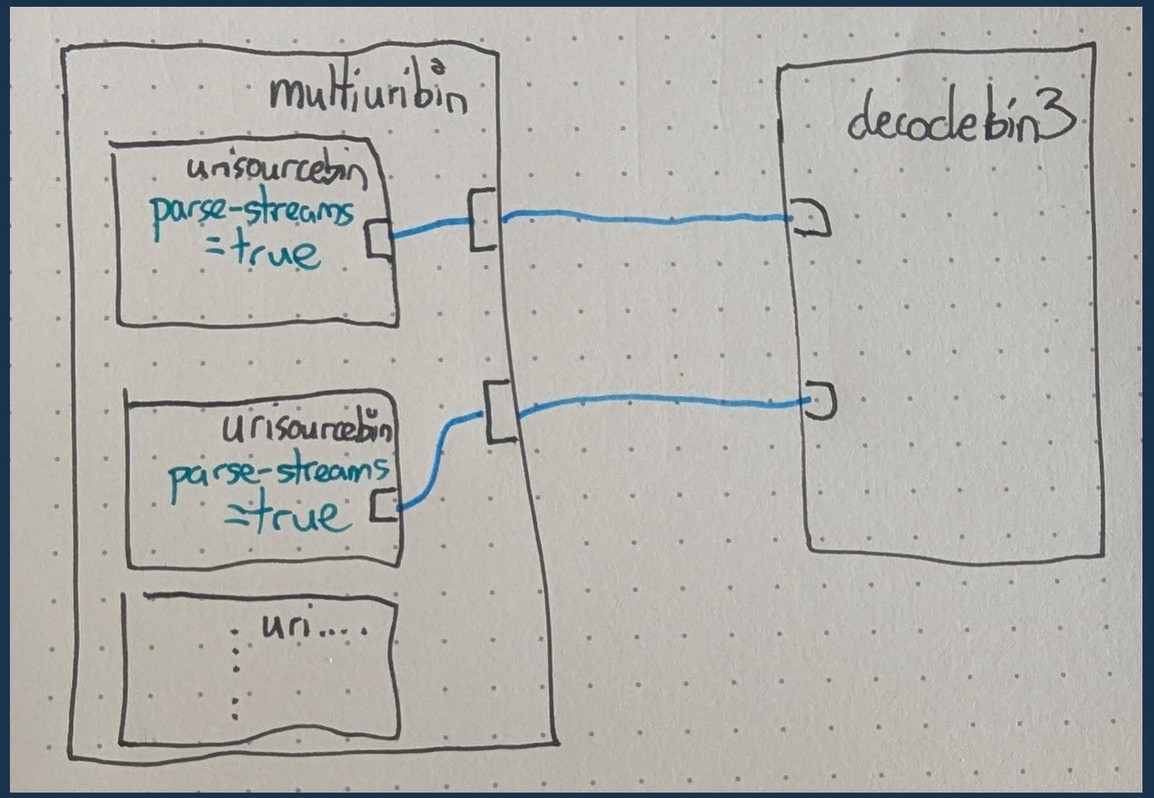
- **Parsebin** support added to **urisourcebin**
- `parse-streams=true` property
- Decodebin no longer does the parsing
- Stream Collections come from **urisourcebin**
- **Decodebin3** usually still does stream selection (except with Adaptive Demuxers)



Centricular

Multiuribin

- Manage multiple **urisourcebin**
- Combine **StreamCollections** to make everything appear as one source
- Parsing is done in the **urisourcebin**s
- Disable unused sources

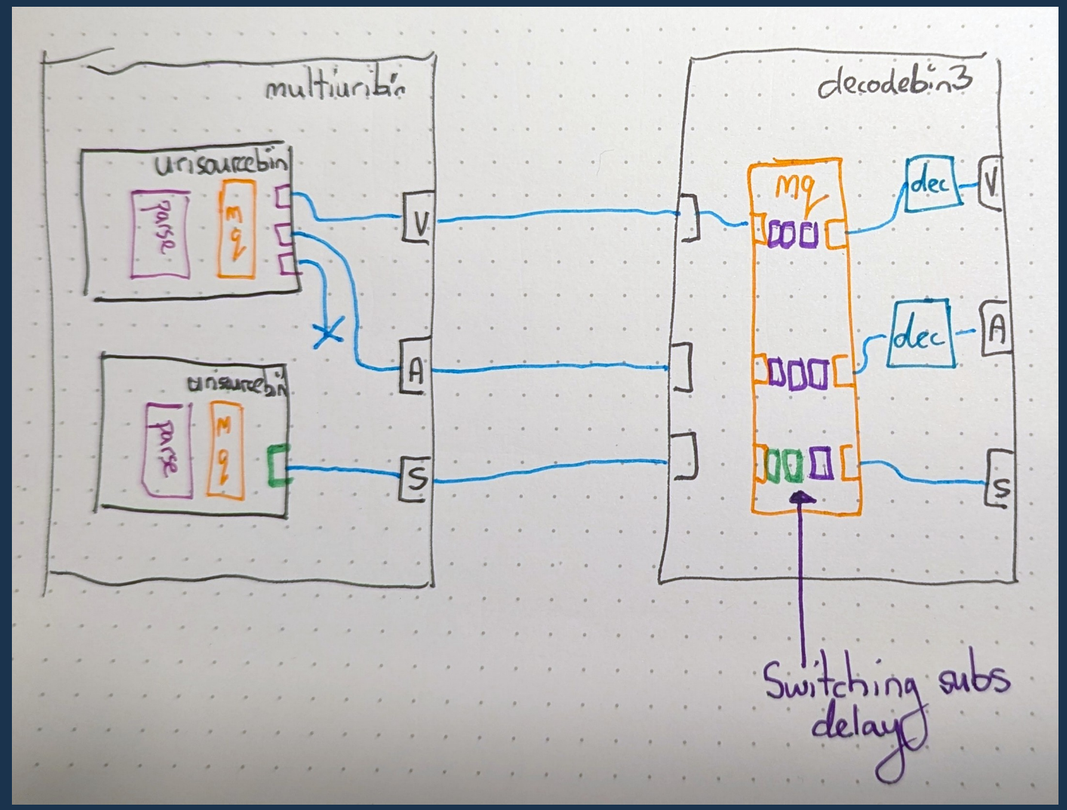


Handling Stream Selections

- Who is going to do it now?
- Multiuribin!
  - Receive select-streams events
  - Map back to the urisourcebin providing each stream
  - Disable/enable urisourcebin sources
  - Maybe send filtered selection upstream when sources can already handle stream selection
  - Otherwise stream selection handling inside multiuribin by exposing only selected pads

Stream Selection Details

- Switching delay / latency problems
  - Downstream queues cause delays
  - Serialising switching causes delays
- Streams still need their own pads so they can flow in parallel



Stream Selection Details

- Previously decodebin3 did a ‘default’ selection (1 stream of each type)
  - But now, decodebin3 passes through what it gets
- Playbin3 now does a default selection
  - If the application doesn’t do one
  - **Multiuribin** will otherwise output everything, like a **urisourcebin** would

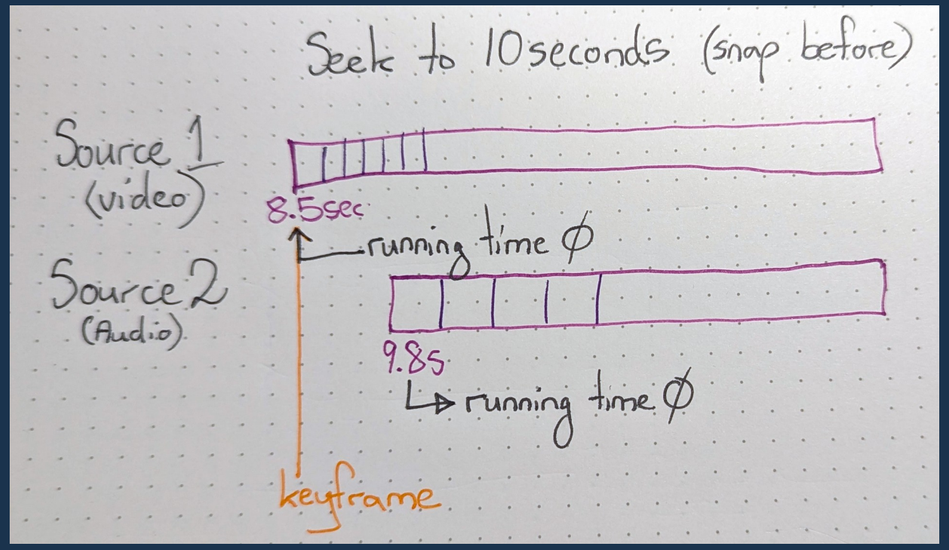
```
/* Iterate the collection and choose the streams that match
 * either the current-* setting, the default, or all streams of a type if there's
 * a combiner for that type */
for (i = 0; i < nb_streams; i++) {
  GstStream *stream = gst_stream_collection_get_stream (collection, i);
  GstStreamType stream_type = gst_stream_get_stream_type (stream);
  const gchar *stream_id = gst_stream_get_stream_id (stream);
  gint pb_stream_type = -1;
  gboolean select_this = FALSE;

  GST_LOG_OBJECT (playbin, "Looking at stream #%d : %s", i, stream_id);

  if (stream_type & GST_STREAM_TYPE_AUDIO) {
    pb_stream_type = PLAYBIN_STREAM_AUDIO;
    /* Select the stream if it's the current one or if there's a custom selector */
    select_this =
        (nb_audio == playbin->current_audio ||
        (playbin->current_audio == -1 && nb_audio == default_nb_audio) ||
        playbin->audio_stream_combiner != NULL);
    nb_audio++;
  } else if (stream_type & GST_STREAM_TYPE_VIDEO) {
    pb_stream_type = PLAYBIN_STREAM_VIDEO;
    select_this =
        (nb_video == playbin->current_video ||
        (playbin->current_video == -1 && nb_video == default_nb_video) ||
        playbin->video_stream_combiner != NULL);
    nb_video++;
  } else if (stream_type & GST_STREAM_TYPE_TEXT) {
    pb_stream_type = PLAYBIN_STREAM_TEXT;
    select_this =
        (nb_text == playbin->current_text ||
        (playbin->current_text == -1 && nb_text == default_nb_text) ||
        playbin->text_stream_combiner != NULL);
```

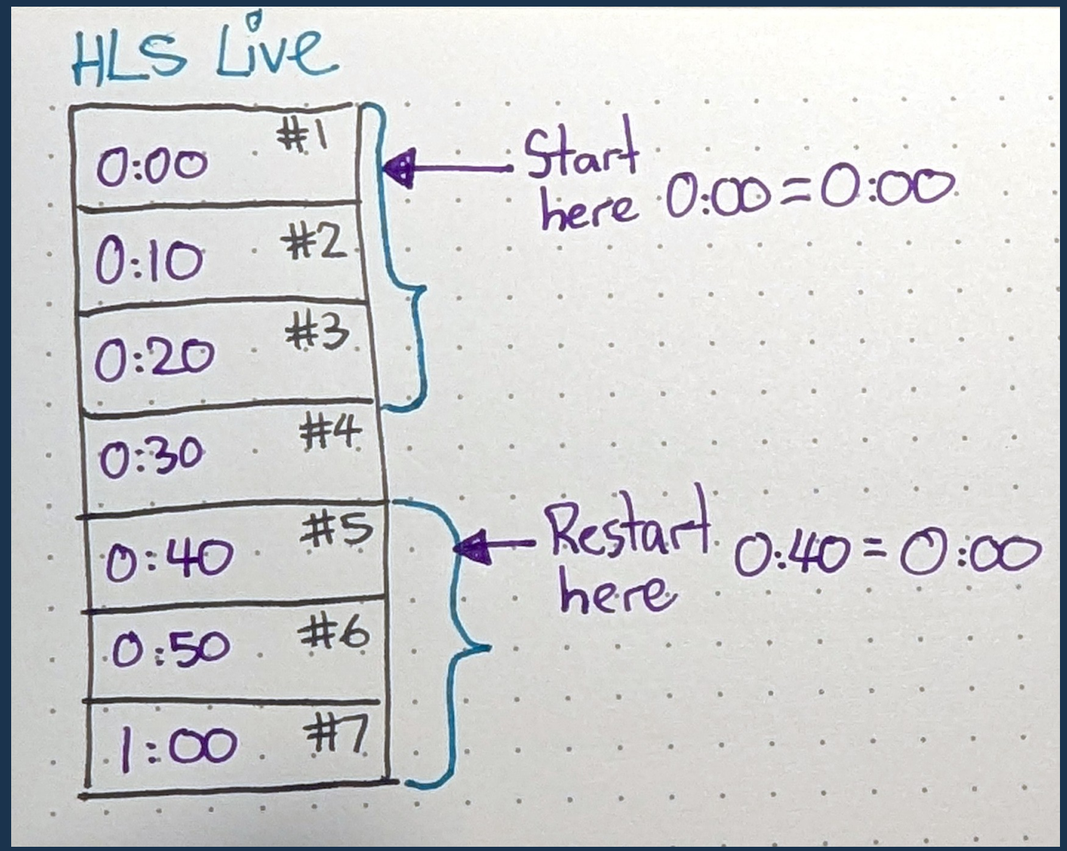
Seeking Challenges

- Streams may not align after seeks
  - Each source may ‘snap’ to a different position on a seek
  - The starts don’t align → synchronisation error
- How to handle?
  - Current idea is to seek 1 stream first, wait, then seek others to match
  - Need to choose the right 1st stream to seek

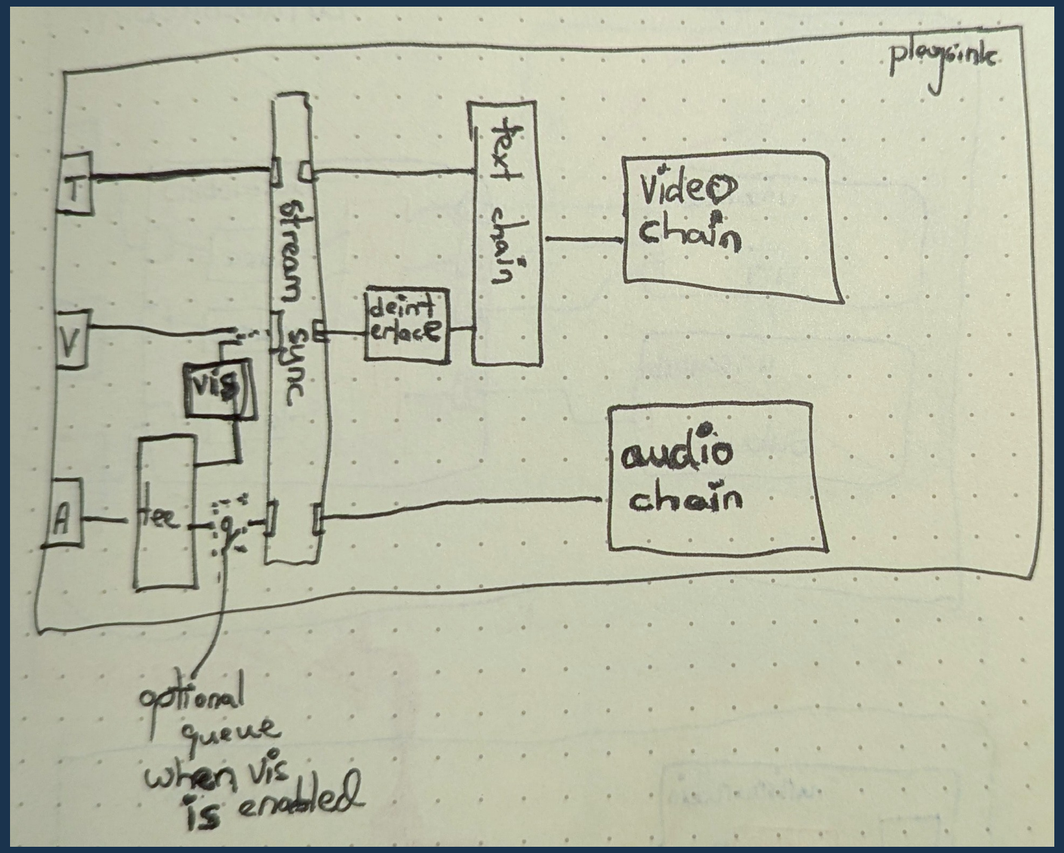


Stream Selection (cont'd)

- Reactivated sources will naturally start at '0'
  - Need a seek to bring them into the current playback position
- The new '0' may not be the same as the old '0'
  - HLS/Dash or other live sources
  - Need a way to 'align' streams independent of *stream time*



Playsink Reconfiguration Problems



- Stream types can be added/removed dynamically
- Playsink waits for input pads to be blocked
- Works OK for adding, not for removing
  - Preroll failures
  - Input chains get stuck

Extending Queue

- Add the ability to wake up queue sinkpad thread without flushing
  - Like a flush that stops at the queue
  - **wakeup-start**, wait for pad idle with a probe and unlink/relink, then **wakeup-stop**
- Generally useful in dynamic pipelines
- Approach could be extended to other elements

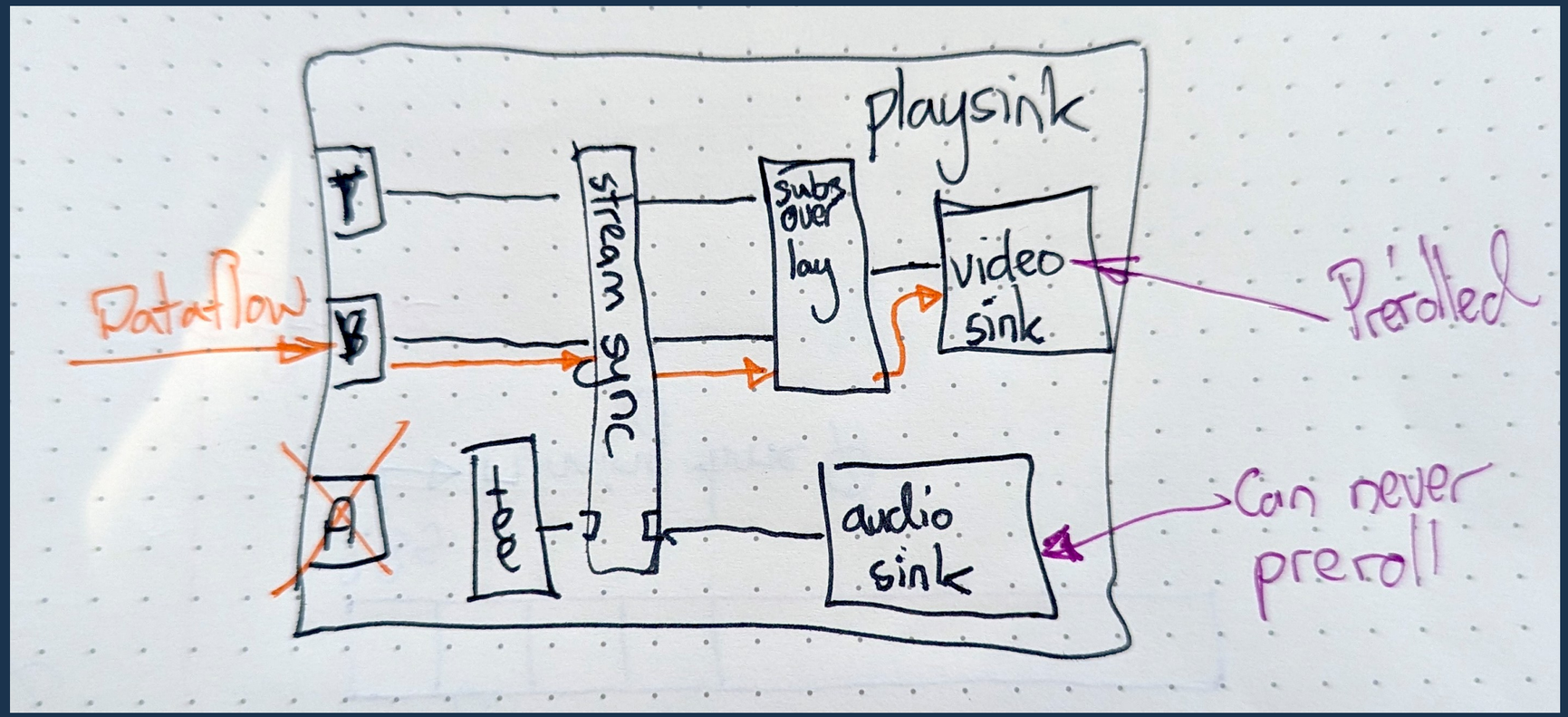
```
Element Actions:

  "sink-wakeup-start" -> void  :  g_signal_emit_by_name (element, "sink-wakeup-start",
                                                         GstFlowReturn arg0);

  "sink-wakeup-stop" -> void  :  g_signal_emit_by_name (element, "sink-wakeup-stop");
```

Playsink Reconfiguration Problems

- Stuck in preroll



Idle Probes

- Idle probes are unique
  - They might be called immediately!
  - But there's no way to know

```
* @GST_PAD_PROBE_TYPE_SCHEDULING: probe push and pull
* @GST_PAD_PROBE_TYPE_IDLE_IMMEDIATE: Used internally to inform IDLE probe
*     callbacks that the callback is happening immediately from the add_probe()
*     method, to differentiate delayed background callbacks. (Since: 1.28)
*
* The different probing types that can occur. When either one of
```

- Simplifies probe addition / callbacks

Questions?

https://gitlab.freedesktop.org/thaytan/gstreamer/-/commits/multiuribin