Liquidsoap 2.x

The last 3 years in the rear-view mirror

Romain Beauxis, FOSDEM 2023

Liquidsoap in a nutshell

Liquidsoap in a nutshell

```
# A bunch of files and playlists,
# supposedly all located in the same base dir.

default = single("~/radio/default.ogg")

day     = playlist("~/radio/day.pls")
night   = playlist("~/radio/night.pls")
jingles = playlist("~/radio/jingles.pls")
clock   = single("~/radio/clock.ogg")

# Play user requests if there are any,
# otherwise one of our playlists,
# and the default file if anything goes wrong.
radio = fallback([ request.queue(id="request"),
                   switch([({ 6h-22h }, day),
                           ({ 22h-6h }, night)]),
                   default])
```

Liquidsoap in a nutshell

```
# A bunch of files and playlists,
# supposedly all located in the same base dir.

default = single("~/radio/default.ogg")

day     = playlist("~/radio/day.pls")
night   = playlist("~/radio/night.pls")
jingles = playlist("~/radio/jingles.pls")
clock   = single("~/radio/clock.ogg")

# Play user requests if there are any,
# otherwise one of our playlists,
# and the default file if anything goes wrong.
radio = fallback([ request.queue(id="request"),
                   switch([({ 6h-22h }, day),
                           ({ 22h-6h }, night)]),
                   default])
```

```
# Add the ability to relay live shows
full =
  fallback(track_sensitive=false,
           [input.http("http://localhost:8000/live.ogg"),
            radio])

# Output the full stream in OGG and MP3
output.icecast(%mp3,
  host="localhost",port=8000,password="hackme",
  mount="radio",full)
output.icecast(%vorbis,
  host="localhost",port=8000,password="hackme",
  mount="radio.ogg",full)
```

Liquidsoap since FOSDEM 2020

Liquidsoap since FOSDEM 2020

- Radio France collaboration

Liquidsoap since FOSDEM 2020

- Radio France collaboration
- Community growth

Liquidsoap since FOSDEM 2020

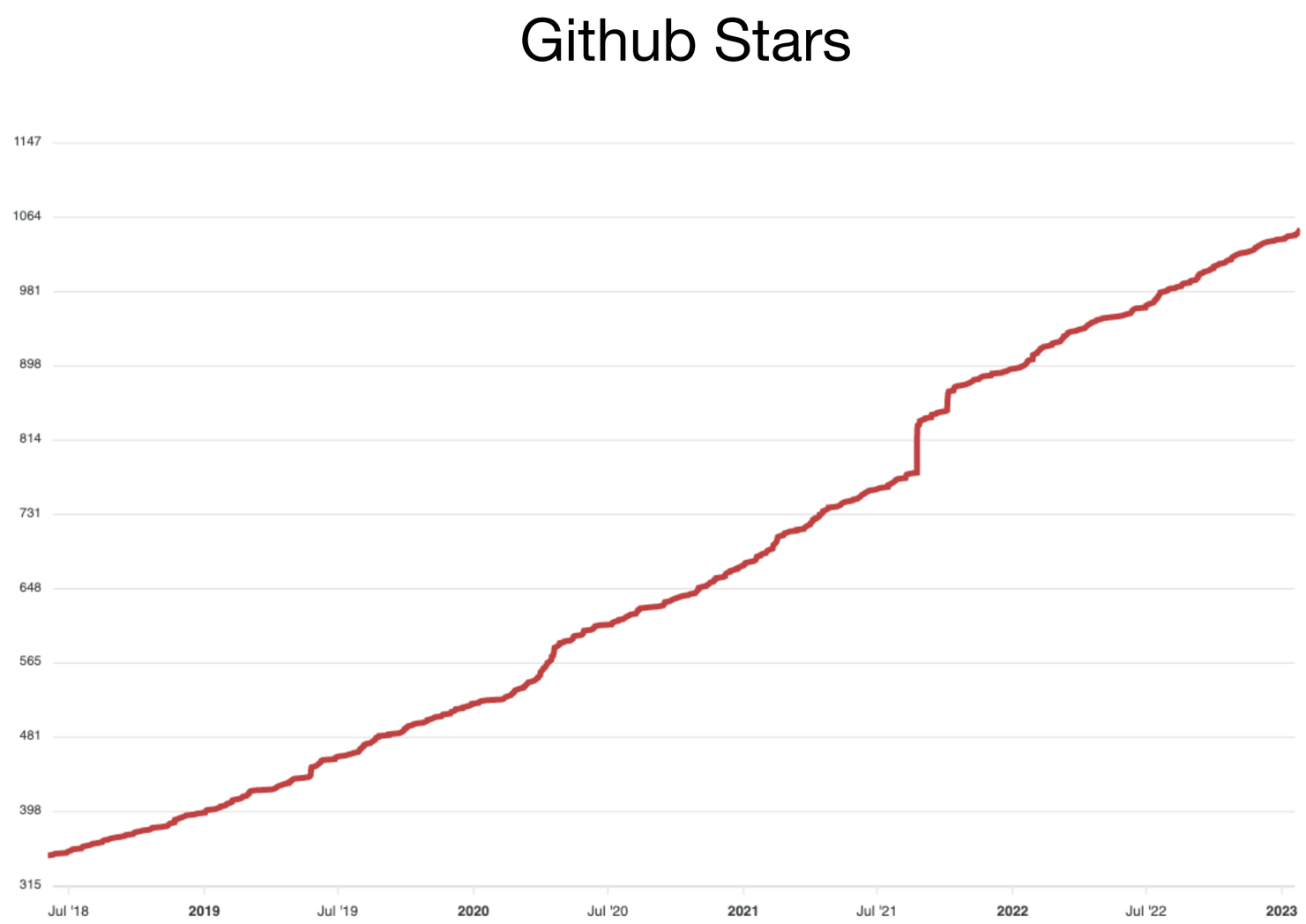
- Radio France collaboration
- Community growth
- New features

Liquidsoap since FOSDEM 2020

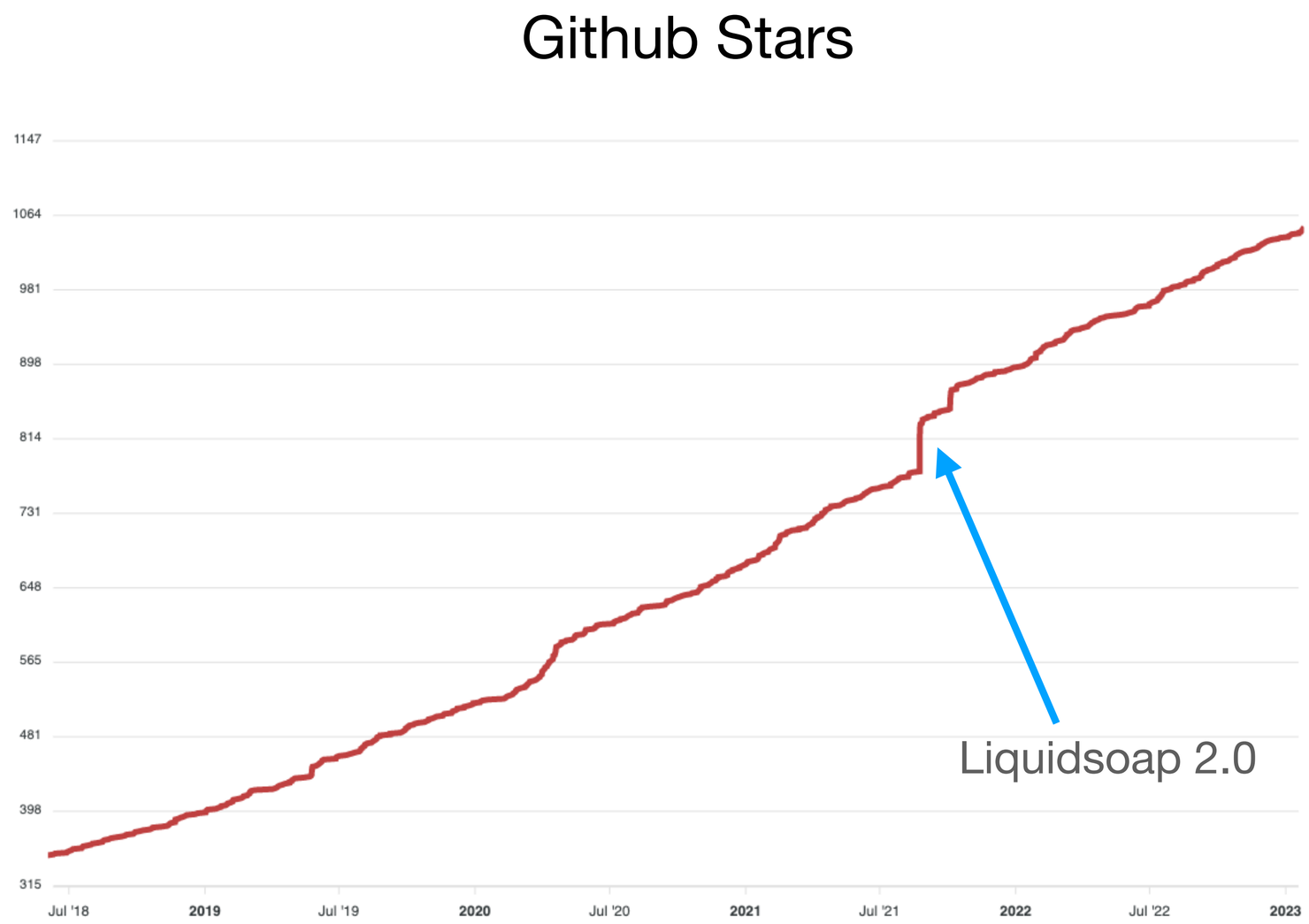
- Radio France collaboration
- Community growth
- New features
- Future Development and challenges

Community Growth

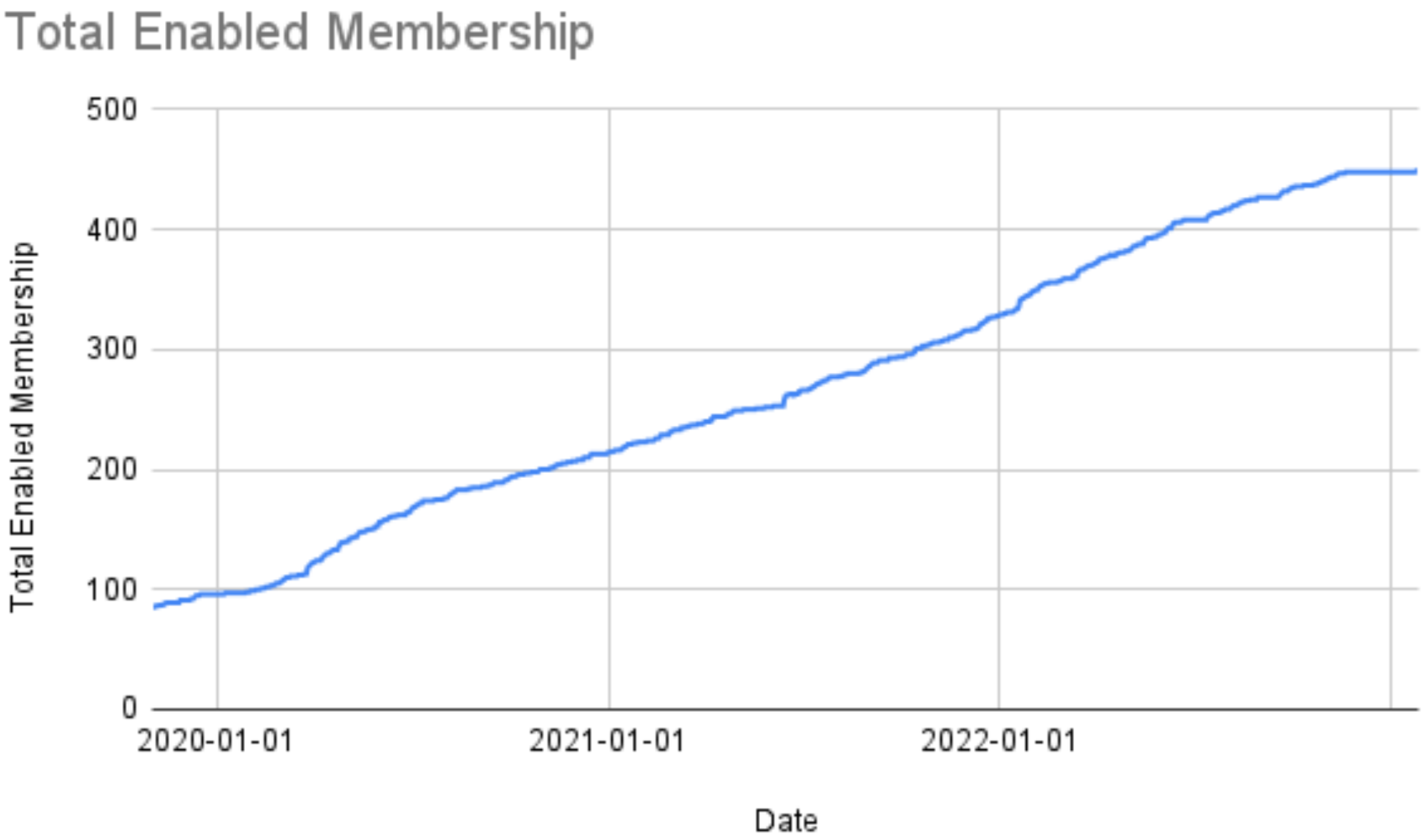
Community Growth



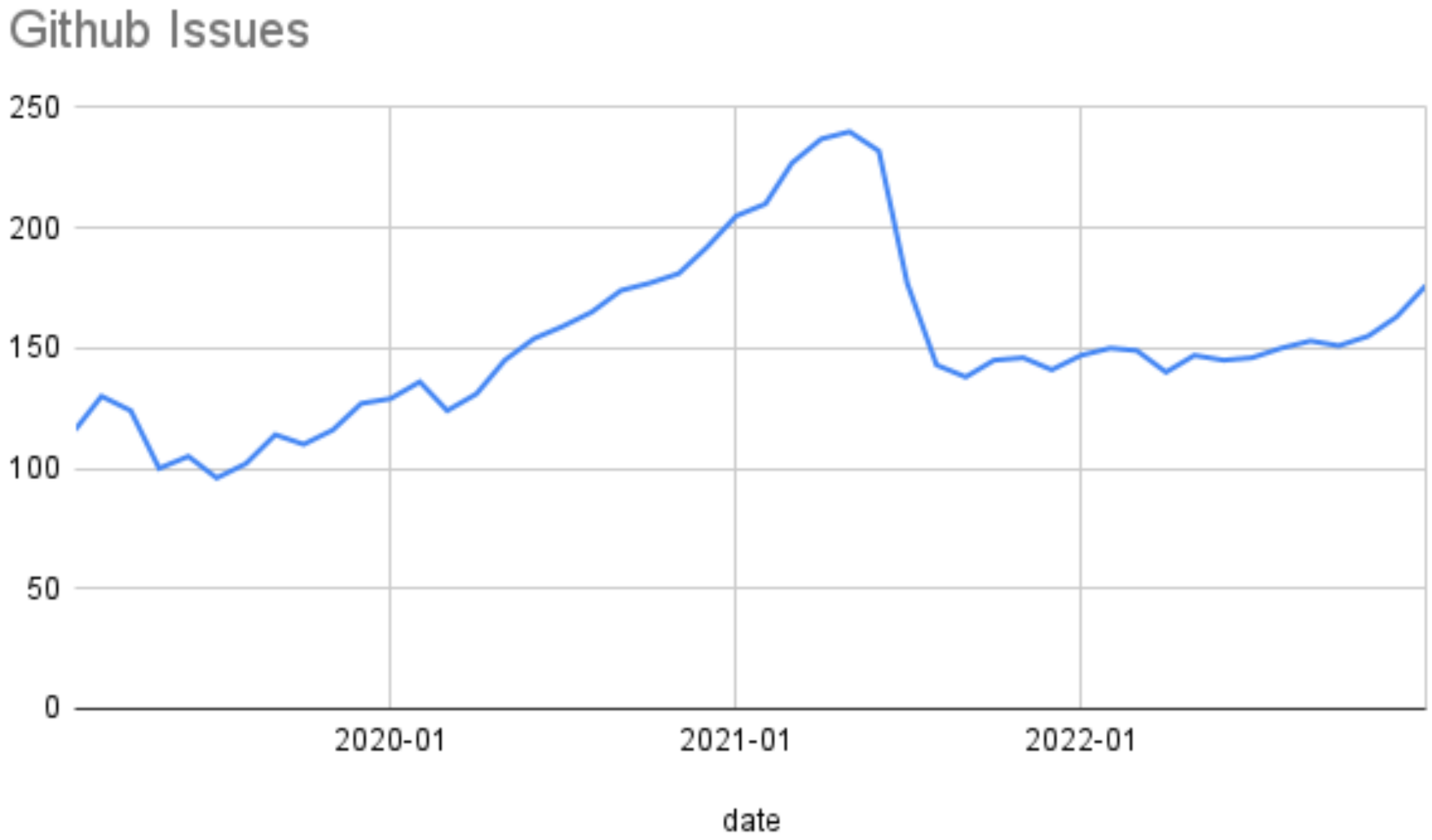
Community Growth



Community Growth



Community Growth



Community Growth

./ Liquidshop – the Liquidsoap workshop

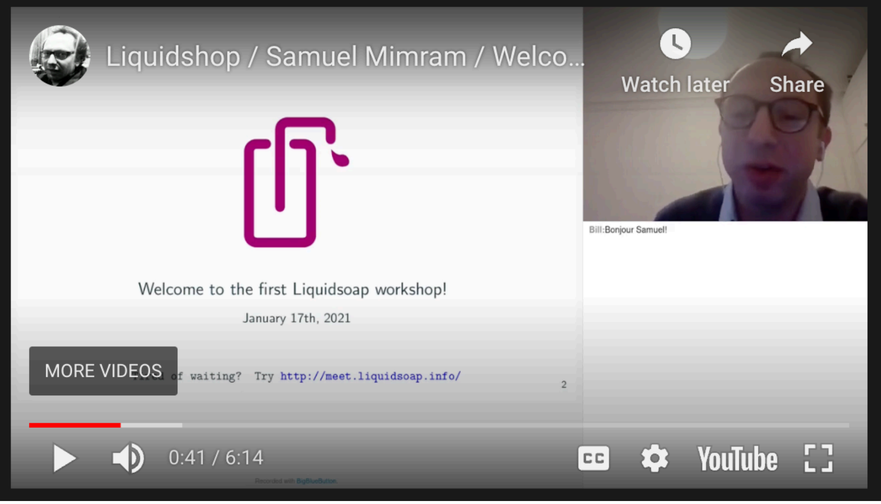
The workshop around Liquidsoap and related technologies.

The **liquidshop** was the first edition of a workshop about Liquidsoap and related technologies (for streaming audio and video, interacting through websites, etc.). It was held fully online.

The full video of the conference is available here (slightly better quality than individual videos below).

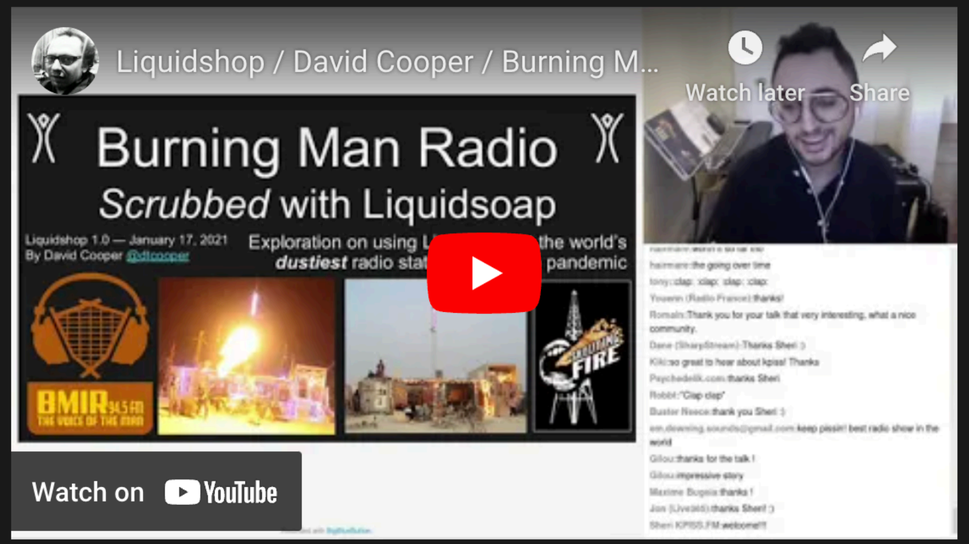
Videos

Samuel Mimram: *Welcome* / slides



Community Growth

David Cooper: *Burning Man Radio Scrubbed with Liquidsoap* / slides



We'll go over how we've mashed up Liquidsoap and other technologies in novel ways to support BMIR 94.5 FM and Shouting Fire radio. Our primary goals were,

>> Support for fully decentralized stations (helpful during the pandemic);
>> Getting as many technical and non-technical folks able to stream using an *Idiot-Tolerant™ DJing* approach;
>> Tight authorization windows for DJs;
>> Simple to schedule shows with a shared Google Calendar; and
>> Zero listener downtime.

The approach has motived an in-development project called Crazy Arms Radio Backend, straightforward backend-only playout software for the modern, fully decentralized station.

Community Growth

Antonia Folguera and Pedro Vílchez: *xrcb.cat – Barcelona Community Radio Network* / slides

Peter Bokor: *Liquidsoap and the Lahmacun community radio* / slides

Community Growth

New Features

- Language changes

New Features

- Language changes
- FFmpeg integration

Language Changes

More expressivity

Language Changes

More expressivity

```
# Forward capture:
let [x, y, z] = [1, 2, 3]
# x = 1, y = 2, z = 3

# Forward capture with spread:
let [x, y, ...z] = [1, 2, 3, 4]
# x = 1, y = 2, z = [3, 4]

# Forward capture with ignored values:
let [_, x, ...z] = [1, 2, 3, 4]
# x = 2, z = [3, 4]
```

Language Changes
More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

Language Changes

More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

```
def f(~opt=null(), x) =
  y = opt ?? "no arg"
  print(y)
  print(x)
end

f : (?opt : string?, 'a) -> unit = <fun>
```

Language Changes

More expressivity

- Nullable type

```
x = null()

y = x ?? "foo"
y : string = "foo"

x = null("bla")

y = x ?? "foo"
y : string = "bla"
```

```
def f(~opt=null(), x) =
  y = opt ?? "no arg"
  print(y)
  print(x)
end

f : (?opt : string?, 'a) -> unit = <fun>
```

Language Changes

More expressivity

```
song = "test.mp3".{duration = 123., bpm = 120.}
```

Language Changes

More expressivity

```
song = "test.mp3".{duration = 123., bpm = 120.}

song.duration
```

Language Changes

More expressivity

```
song = "test.mp3".{duration = 123., bpm = 120.}

song.duration

print("song: #{song}")
```

Language Changes

More expressivity

- Module and records

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

- Module and records

```
s = single("/path/to/file.mp3")

s.skip()
```

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

- Module and records

```
s = single("/path/to/file.mp3")

s.skip()


s = insert_metadata(s)

s.insert_metadata(
  [("title", "Awesome Track")
])
```

```
source(audio=?A, video=?B, midi=none)
  .{
    time : () -> float,
    shutdown : () -> unit,
    fallible : () -> bool,
    skip : () -> unit,
    seek : (float) -> float,
    is_up : () -> bool,
    remaining : () -> float,
    on_track : ((([string * string]) -> unit)) -> unit,
    on_shutdown : ((() -> unit)) -> unit,
    on_metadata : ((([string * string]) -> unit)) -> unit,
    is_ready : () -> bool,
    id : (() -> string)
  }
```

Language Changes

More expressivity

```
let json.parse ({
  name,
  version,
  scripts = {
    test
  }
} : {
  name: string,
  version: string,
  scripts: {
    test: string
  }
}) = file.contents("/path/to/package.json")
```

Language Changes

More expressivity

```
let yaml.parse ({
  name,
  version,
  scripts,
} : {
  name: string,
  version: string,
  scripts: {
    test: string?
  }?
}) = file.contents("/path/to/file.yaml")
```

FFmpeg integration

Anything is possible!

FFmpeg integration

Anything is possible!

AVPacket Struct Reference

AVPacket

This structure stores compressed data. More...

```
#include <packet.h>
```

Data Fields

AVBufferRef *	**buf** A reference to the reference-counted buffer where the packet data is stored. More...
int64_t	**pts** Presentation timestamp in AVStream->time_base units; the time at which the decompressed packet will be presented to the user. More...
int64_t	**dts** Decompression timestamp in AVStream->time_base units; the time at which the packet is decompressed. More...
uint8_t *	**data**

FFmpeg integration

Anything is possible!

```
%ffmpeg(format="mpegts",
  %audio(codec="ac3",channel_coupling=0),
  %video(codec="libx264",b="2600k",
         "x264-params"="scenecut=0:open_gop=0:min-keyint=150:keyint=150",
         preset="ultrafast"))
```

FFmpeg integration

Anything is possible!

```
%ffmpeg(format="mpegts",
  %audio(codec="ac3",channel_coupling=0),
  %video(codec="libx264",b="2600k",
         "x264-params"="scenecut=0:open_gop=0:min-keyint=150:keyint=150",
         preset="ultrafast"))

%ffmpeg(format="mp3",
  %audio(codec="libmp3lame"),
  %video.copy)
```

FFmpeg integration

Anything is possible!

```
static const AVOption options[] = {
    { "preset",       "Set the encoding preset (cf. x264 --fullhelp)",   OFFSET(preset),       AV_OPT_TYPE_STRING, { .str = "medium" }, 0, 0, VE},
    { "tune",         "Tune the encoding params (cf. x264 --fullhelp)",  OFFSET(tune),         AV_OPT_TYPE_STRING, { 0 }, 0, 0, VE},
    { "profile",      "Set profile restrictions (cf. x264 --fullhelp)",  OFFSET(profile_opt),      AV_OPT_TYPE_STRING, { 0 }, 0, 0, VE},
    { "fastfirstpass", "Use fast settings when encoding first pass",     OFFSET(fastfirstpass), AV_OPT_TYPE_BOOL, { .i64 = 1 }, 0, 1, VE},
    {"level", "Specify level (as defined by Annex A)", OFFSET(level), AV_OPT_TYPE_STRING, {.str=NULL}, 0, 0, VE},
    {"passlogfile", "Filename for 2 pass stats", OFFSET(stats), AV_OPT_TYPE_STRING, {.str=NULL}, 0, 0, VE},
    {"wpredp", "Weighted prediction for P-frames", OFFSET(wpredp), AV_OPT_TYPE_STRING, {.str=NULL}, 0, 0, VE},
    {"a53cc",         "Use A53 Closed Captions (if available)",          OFFSET(a53_cc),       AV_OPT_TYPE_BOOL,   {.i64 = 1}, 0, 1, VE},
    {"x264opts", "x264 options", OFFSET(x264opts), AV_OPT_TYPE_STRING, {.str=NULL}, 0, 0, VE},
    { "crf",          "Select the quality for constant quality mode",    OFFSET(crf),          AV_OPT_TYPE_FLOAT,  {.dbl = -1 }, -1, FLT_MAX, VE },
    { "crf_max",      "In CRF mode, prevents VBV from lowering quality beyond this point.",OFFSET(crf_max), AV_OPT_TYPE_FLOAT, {.dbl = -1 }, -1, FLT_MAX, VE },
    { "qp",           "Constant quantization parameter rate control method",OFFSET(cqp),       AV_OPT_TYPE_INT,    { .i64 = -1 }, -1, INT_MAX, VE },
    { "aq-mode",      "AQ method",                                       OFFSET(aq_mode),      AV_OPT_TYPE_INT,    { .i64 = -1 }, -1, INT_MAX, VE, "aq_mode"},
    { "none",         NULL,                                 0, AV_OPT_TYPE_CONST, {.i64 = X264_AQ_NONE},         INT_MIN, INT_MAX, VE, "aq_mode" },
    { "variance",     "Variance AQ (complexity mask)",      0, AV_OPT_TYPE_CONST, {.i64 = X264_AQ_VARIANCE},     INT_MIN, INT_MAX, VE, "aq_mode" },
    { "autovariance", "Auto-variance AQ",                   0, AV_OPT_TYPE_CONST, {.i64 = X264_AQ_AUTOVARIANCE}, INT_MIN, INT_MAX, VE, "aq_mode" },
#if X264_BUILD >= 144
    { "autovariance-biased", "Auto-variance AQ with bias to dark scenes", 0, AV_OPT_TYPE_CONST, {.i64 = X264_AQ_AUTOVARIANCE_BIASED}, INT_MIN, INT_MAX, VE, "aq_mode" },
#endif
    { "aq-strength",  "AQ strength. Reduces blocking and blurring in flat and textured areas.", OFFSET(aq_strength), AV_OPT_TYPE_FLOAT, {.dbl = -1}, -1, FLT_MAX, VE},
    { "psy",          "Use psychovisual optimizations.",                 OFFSET(psy),          AV_OPT_TYPE_BOOL,   { .i64 = -1 }, -1, 1, VE },
    { "psy-rd",       "Strength of psychovisual optimization, in <psy-rd>:<psy-trellis> format.", OFFSET(psy_rd), AV_OPT_TYPE_STRING,  {0 }, 0, 0, VE},
    { "rc-lookahead", "Number of frames to look ahead for frametype and ratecontrol", OFFSET(rc_lookahead), AV_OPT_TYPE_INT, { .i64 = -1 }, -1, INT_MAX, VE },
    { "weightb",      "Weighted prediction for B-frames.",               OFFSET(weightb),      AV_OPT_TYPE_BOOL,   { .i64 = -1 }, -1, 1, VE },
    { "weightp",      "Weighted prediction analysis method.",            OFFSET(weightp),      AV_OPT_TYPE_INT,    { .i64 = -1 }, -1, INT_MAX, VE, "weightp" },
```

FFmpeg integration

Anything is possible!

```
static const AVOption flanger_options[] = {
    { "delay", "base delay in milliseconds",        OFFSET(delay_min),   AV_OPT_TYPE_DOUBLE, {.dbl=0}, 0, 30, A },
    { "depth", "added swept delay in milliseconds", OFFSET(delay_depth), AV_OPT_TYPE_DOUBLE, {.dbl=2}, 0, 10, A },
    { "regen", "percentage regeneration (delayed signal feedback)", OFFSET(feedback_gain), AV_OPT_TYPE_DOUBLE, {.dbl=0}, -95, 95, A },
    { "width", "percentage of delayed signal mixed with original", OFFSET(delay_gain), AV_OPT_TYPE_DOUBLE, {.dbl=71}, 0, 100, A },
    { "speed", "sweeps per second (Hz)", OFFSET(speed), AV_OPT_TYPE_DOUBLE, {.dbl=0.5}, 0.1, 10, A },
    { "shape", "swept wave shape", OFFSET(wave_shape), AV_OPT_TYPE_INT, {.i64=WAVE_SIN}, WAVE_SIN, WAVE_NB-1, A, "type" },
    { "triangular",  NULL, 0, AV_OPT_TYPE_CONST,  {.i64=WAVE_TRI}, 0, 0, A, "type" },
    { "t",           NULL, 0, AV_OPT_TYPE_CONST,  {.i64=WAVE_TRI}, 0, 0, A, "type" },
    { "sinusoidal",  NULL, 0, AV_OPT_TYPE_CONST,  {.i64=WAVE_SIN}, 0, 0, A, "type" },
    { "s",           NULL, 0, AV_OPT_TYPE_CONST,  {.i64=WAVE_SIN}, 0, 0, A, "type" },
    { "phase", "swept wave percentage phase-shift for multi-channel", OFFSET(channel_phase), AV_OPT_TYPE_DOUBLE, {.dbl=25}, 0, 100, A },
    { "interp", "delay-line interpolation", OFFSET(interpolation), AV_OPT_TYPE_INT, {.i64=0}, 0, 1, A, "itype" },
    { "linear",     NULL, 0, AV_OPT_TYPE_CONST,  {.i64=INTERPOLATION_LINEAR},    0, 0, A, "itype" },
    { "quadratic",  NULL, 0, AV_OPT_TYPE_CONST,  {.i64=INTERPOLATION_QUADRATIC}, 0, 0, A, "itype" },
    { NULL }
};
```

FFmpeg integration

Anything is possible!

```
❯ liquidsoap -h ffmpeg.filter.flanger

Ffmpeg filter: Apply a flanging effect to the audio.

Type: (?delay : float?, ?depth : float?, ?regen : float?,
 ?width : float?, ?speed : float?, ?shape : int?,
 ?phase : float?, ?interp : int?, ffmpeg.filter.graph,
 ffmpeg.filter.audio) -> ffmpeg.filter.audio

Category: Filter
Flag: extra

Parameters:

 * delay : float? (default: null)
     base delay in milliseconds. (default: 0.)

 * depth : float? (default: null)
     added swept delay in milliseconds. (default: 2.)

 * regen : float? (default: null)
     percentage regeneration (delayed signal feedback). (default: 0.)

 * width : float? (default: null)
     percentage of delayed signal mixed with original. (default: 71.)

 * speed : float? (default: null)
     sweeps per second (Hz). (default: 0.5)

 * shape : int? (default: null)
     swept wave shape. (default: 0, possible values: 1 (triangular), 1 (t), 0
     (sinusoidal), 0 (s))

 * phase : float? (default: null)
     swept wave percentage phase-shift for multi-channel. (default: 25.)

 * interp : int? (default: null)
     delay-line interpolation. (default: 0, possible values: 0 (linear), 1
     (quadratic))

 * (unlabeled) : ffmpeg.filter.graph

 * (unlabeled) : ffmpeg.filter.audio
```

FFmpeg integration

Anything is possible!

```
def flanger_highpass(s) =
  def mkfilter(graph) =
    s = ffmpeg.filter.audio.input(graph, s)
    s = ffmpeg.filter.flanger(graph, s, delay=10.)
    s = ffmpeg.filter.highpass(graph, s, frequency=4000.)
    ffmpeg.filter.audio.output(graph, s)
  end

  ffmpeg.filter.create(mkfilter)
end
```

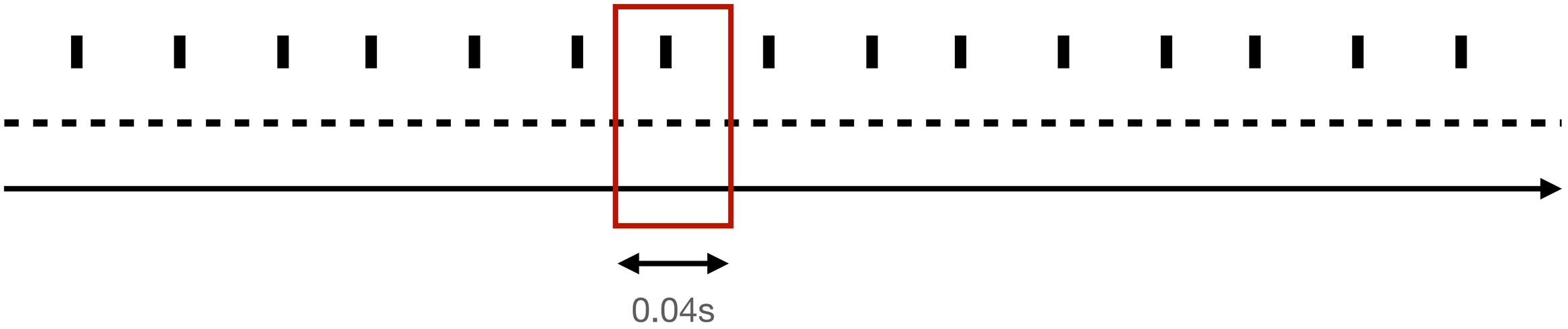
FFmpeg integration

Anything is possible!

```
def parallel_flanger_highpass(s) =
  def mkfilter(graph) =
    s = ffmpeg.filter.audio.input(graph, s)

    let (audio, _) = ffmpeg.filter.asplit(outputs=2, graph, s)

    let [s1, s2] = audio

    s1 = ffmpeg.filter.flanger(graph, s1, delay=10.)
    s2 = ffmpeg.filter.highpass(graph, s2, frequency=4000.)

    # For some reason, we need to enforce the format here.
    s1 = ffmpeg.filter.aformat(sample_fmts="s16", sample_rates="44100", channel_layouts="stereo", graph, s1)
    s2 = ffmpeg.filter.aformat(sample_fmts="s16", sample_rates="44100", channel_layouts="stereo", graph, s2)

    s = ffmpeg.filter.amerge(inputs=2, graph, [s1, s2], [])

    ffmpeg.filter.audio.output(graph, s)
  end

  ffmpeg.filter.create(mkfilter)
end
```

FFmpeg integration

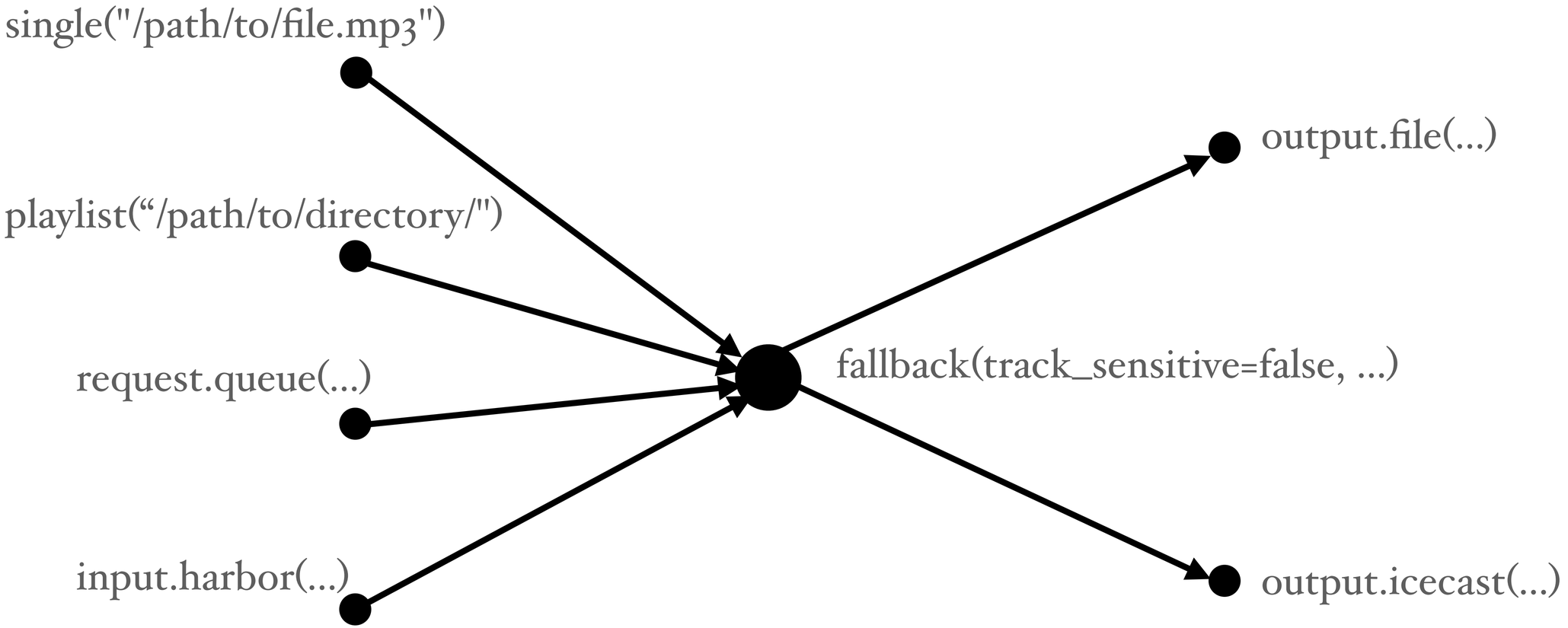
Anything is possible!



[frame:3] Frame size must be a multiple of 1764 ticks = 1764 audio samples = 1 video samples.
[frame:3] Targetting 'frame.duration': 0.04s = 1764 audio samples = 1764 ticks.
[frame:3] Frames last 0.04s = 1764 audio samples = 1 video samples = 1764 ticks.

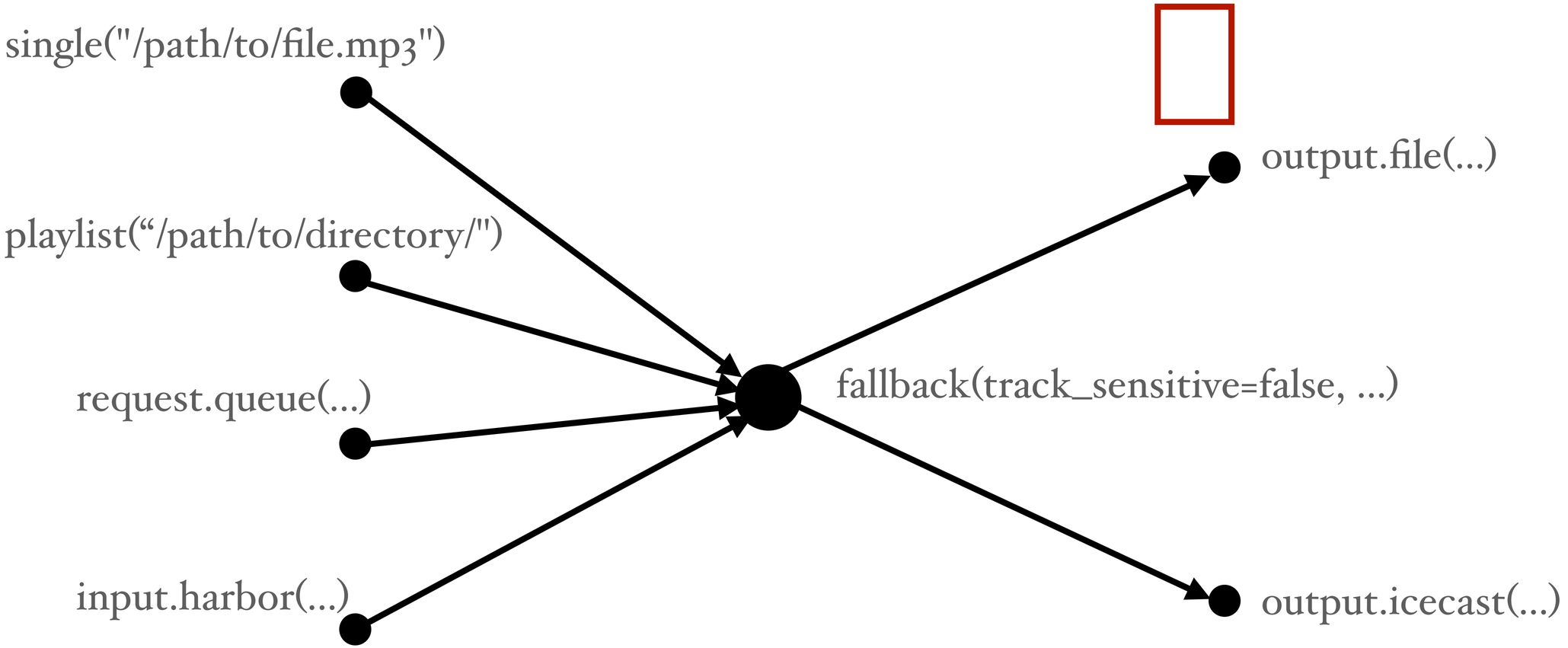
FFmpeg integration

Anything is possible!



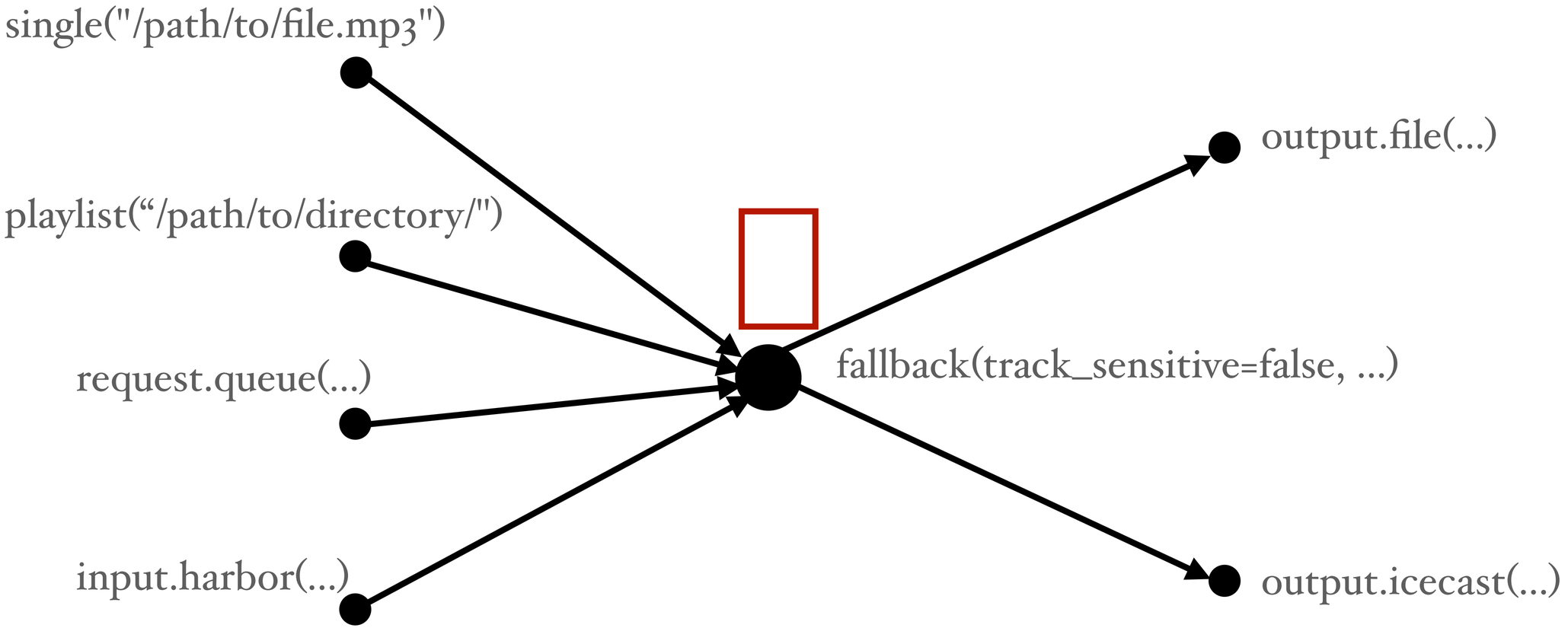
FFmpeg integration

Anything is possible!



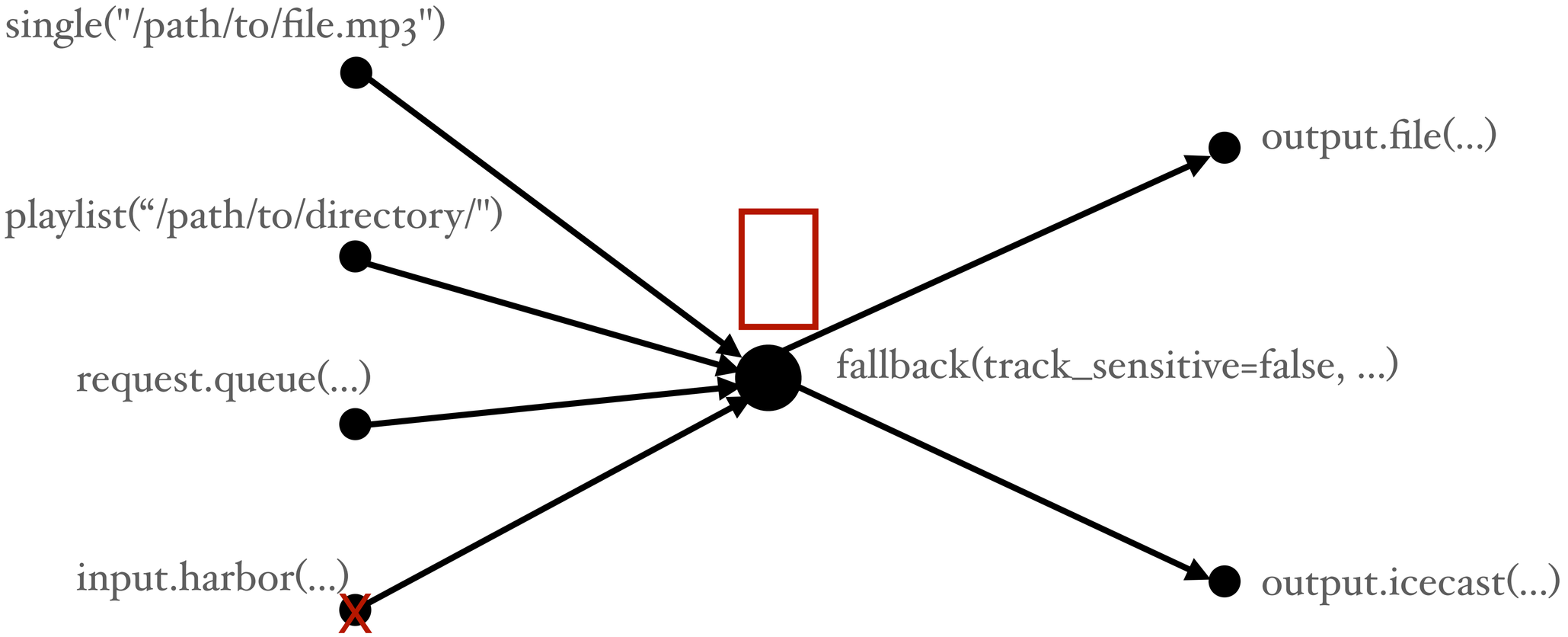
FFmpeg integration

Anything is possible!



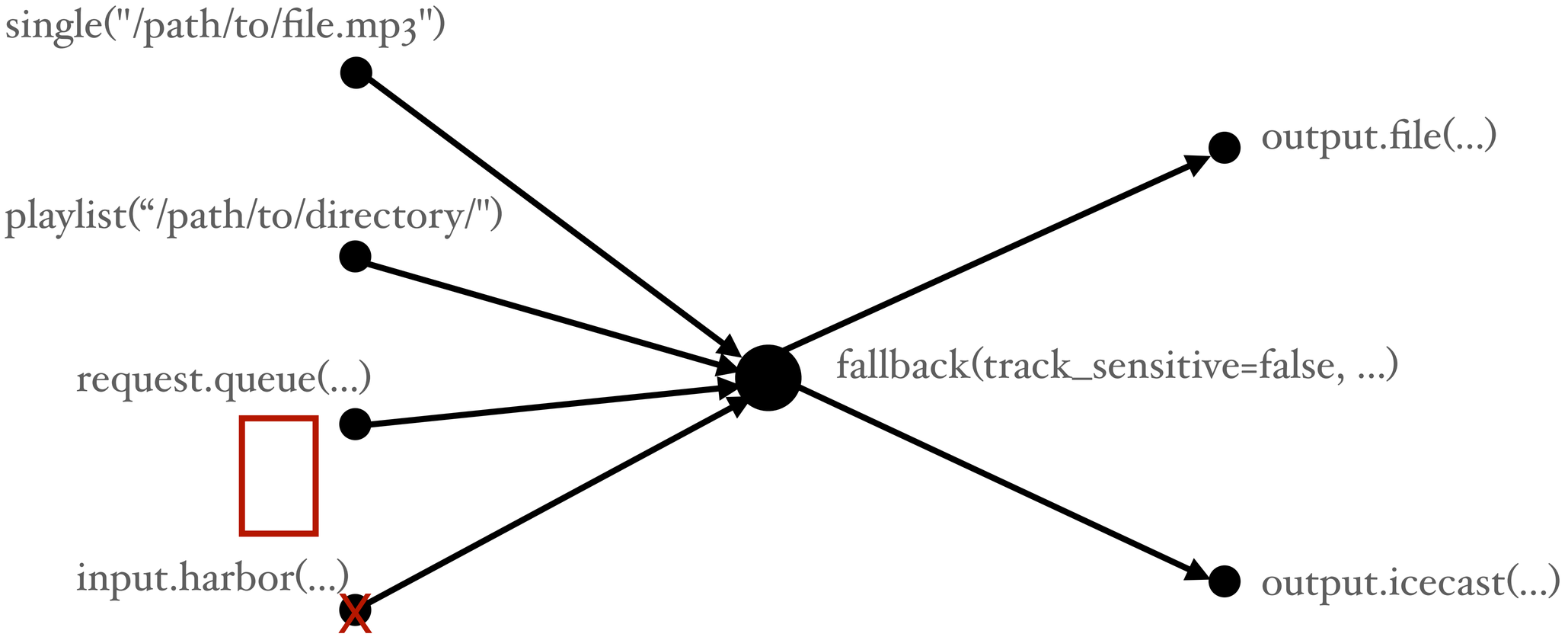
FFmpeg integration

Anything is possible!



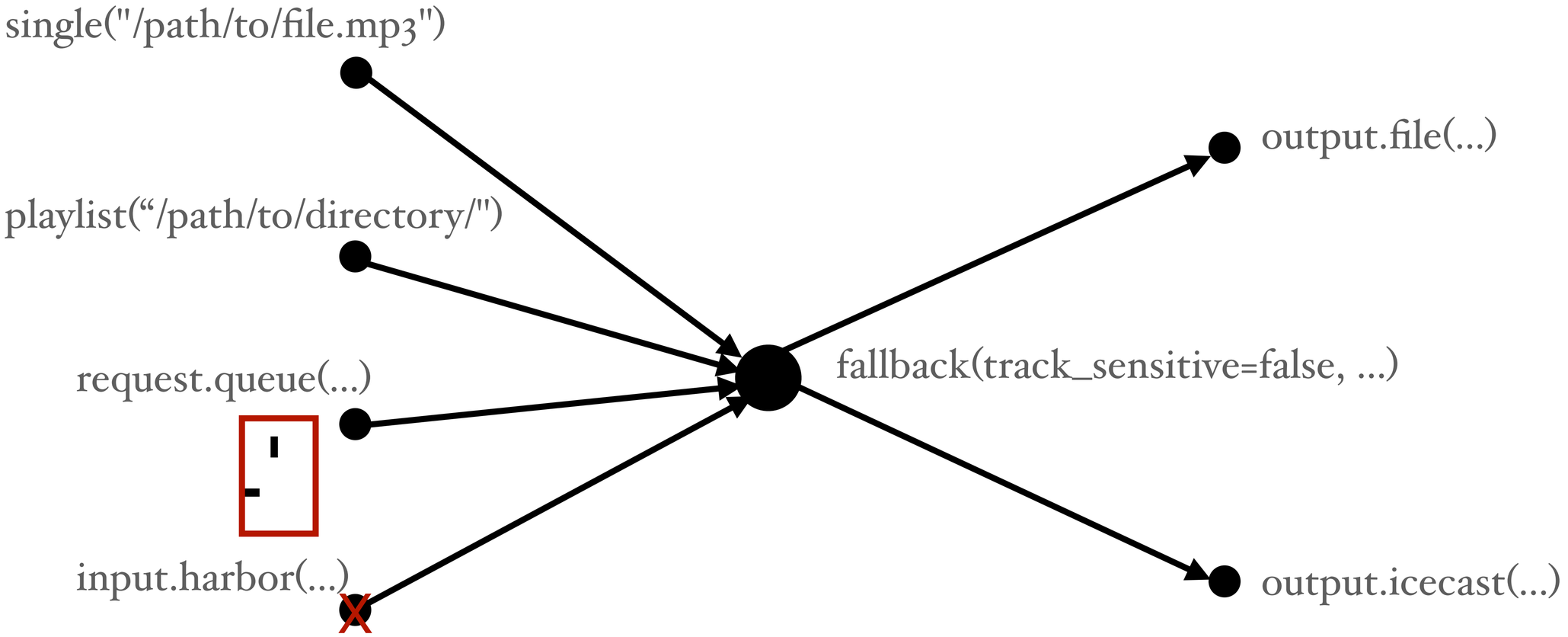
FFmpeg integration

Anything is possible!



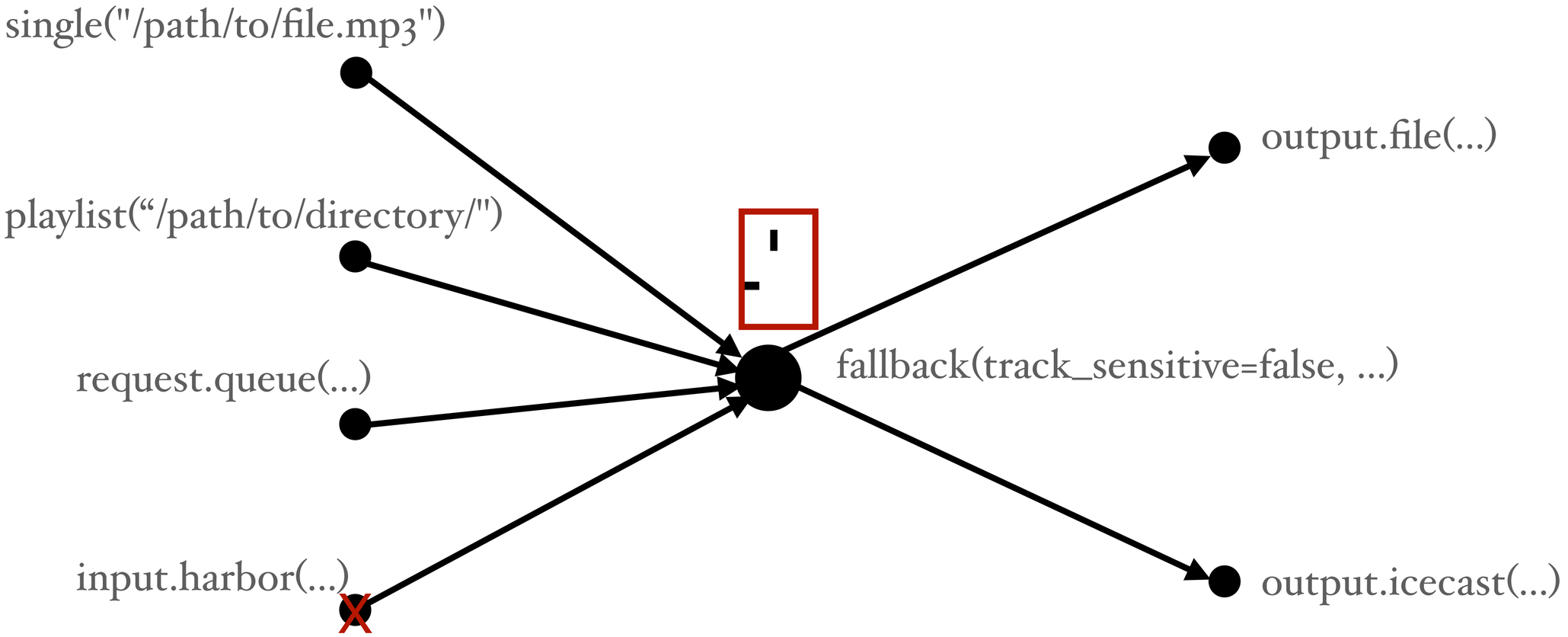
FFmpeg integration

Anything is possible!



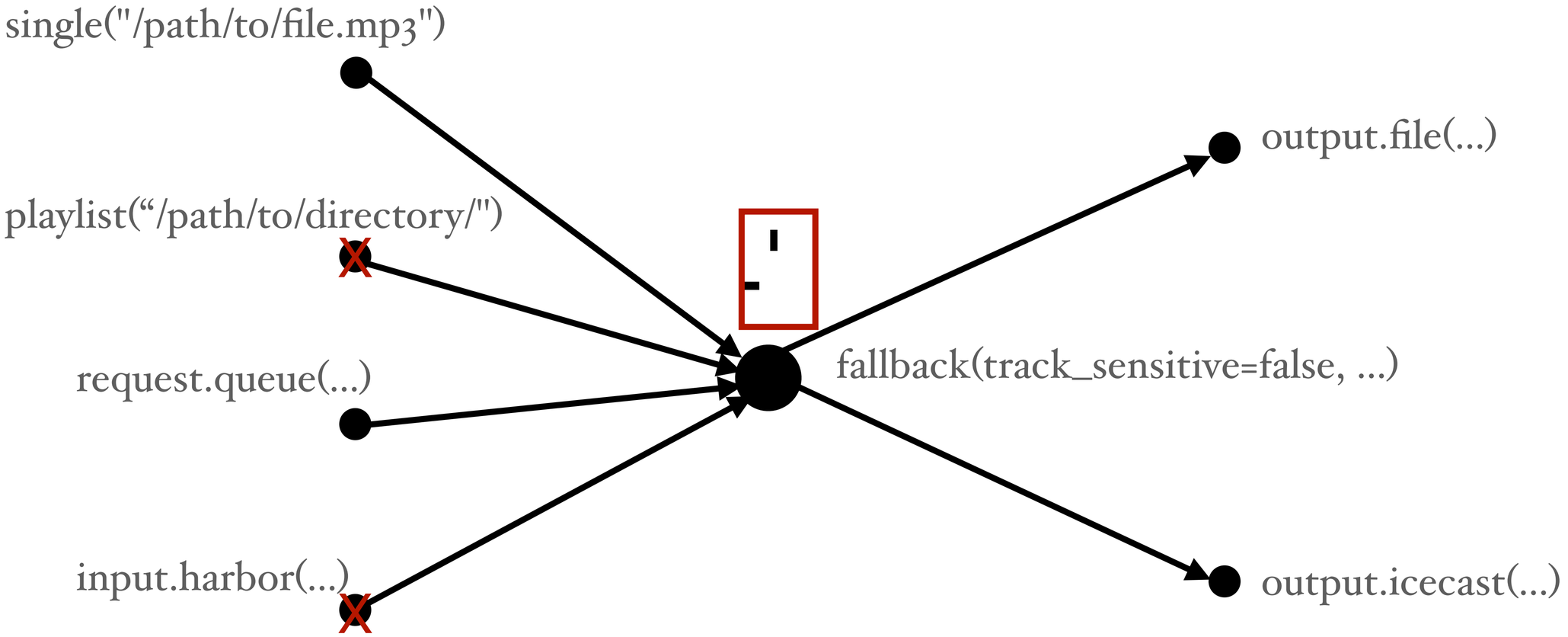
FFmpeg integration

Anything is possible!



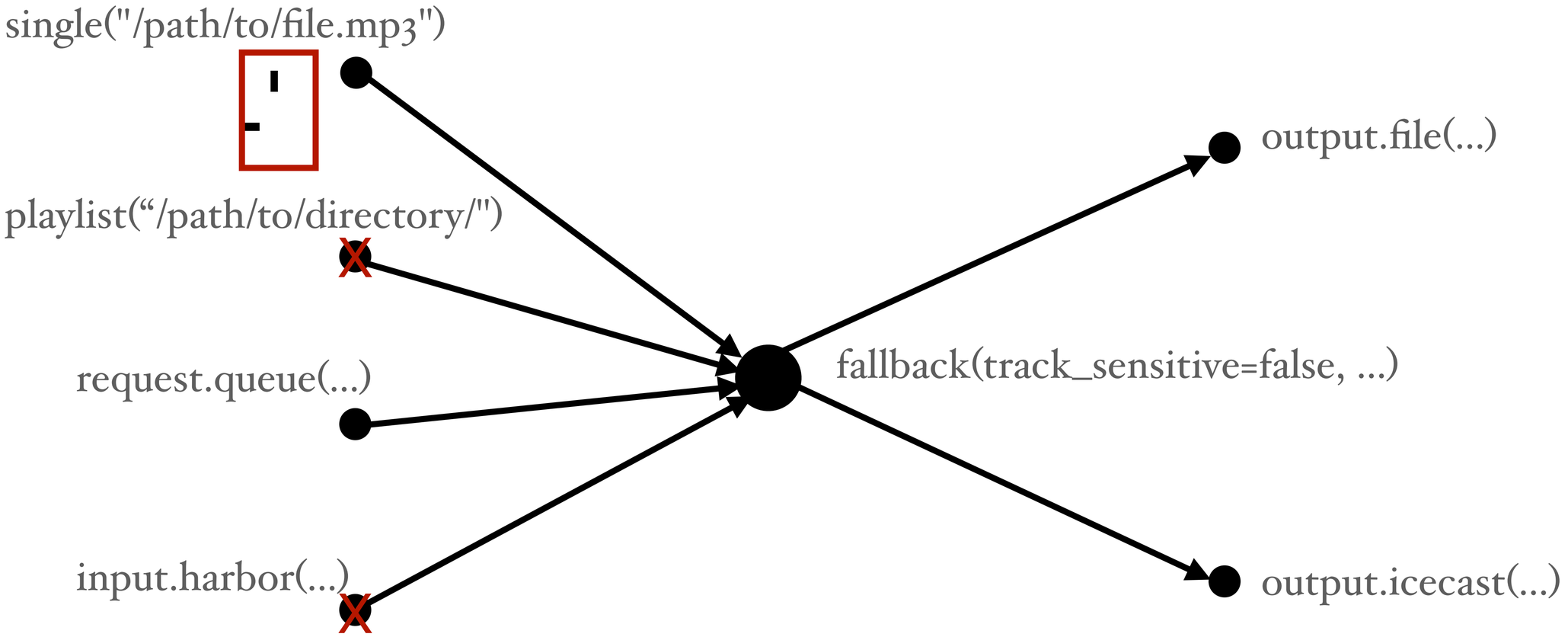
FFmpeg integration

Anything is possible!



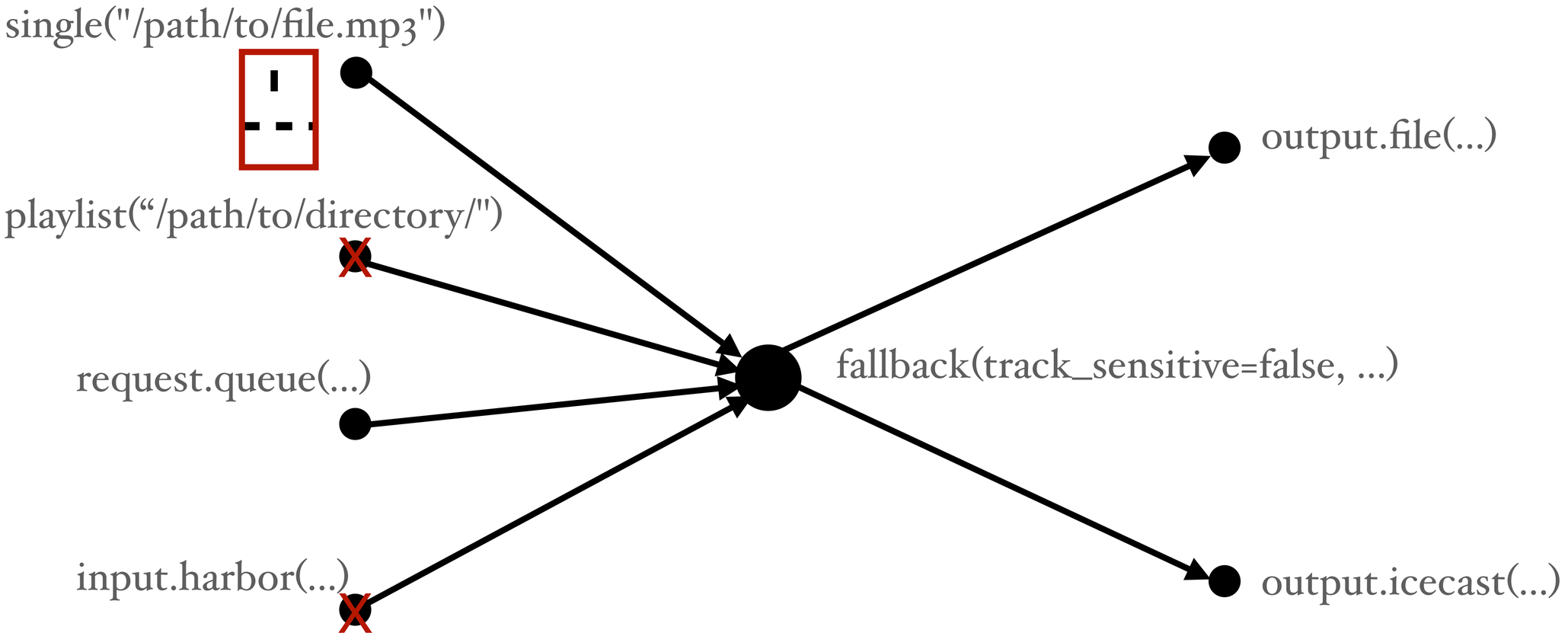
FFmpeg integration

Anything is possible!



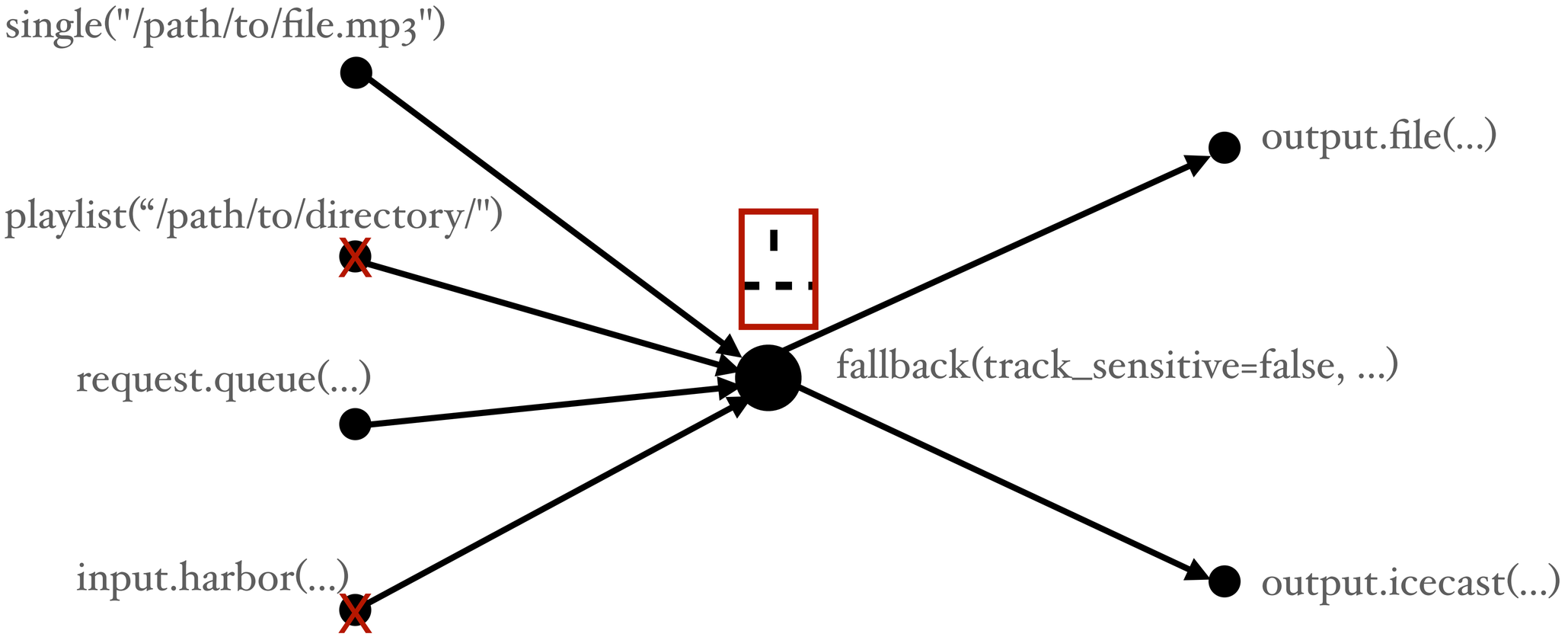
FFmpeg integration

Anything is possible!



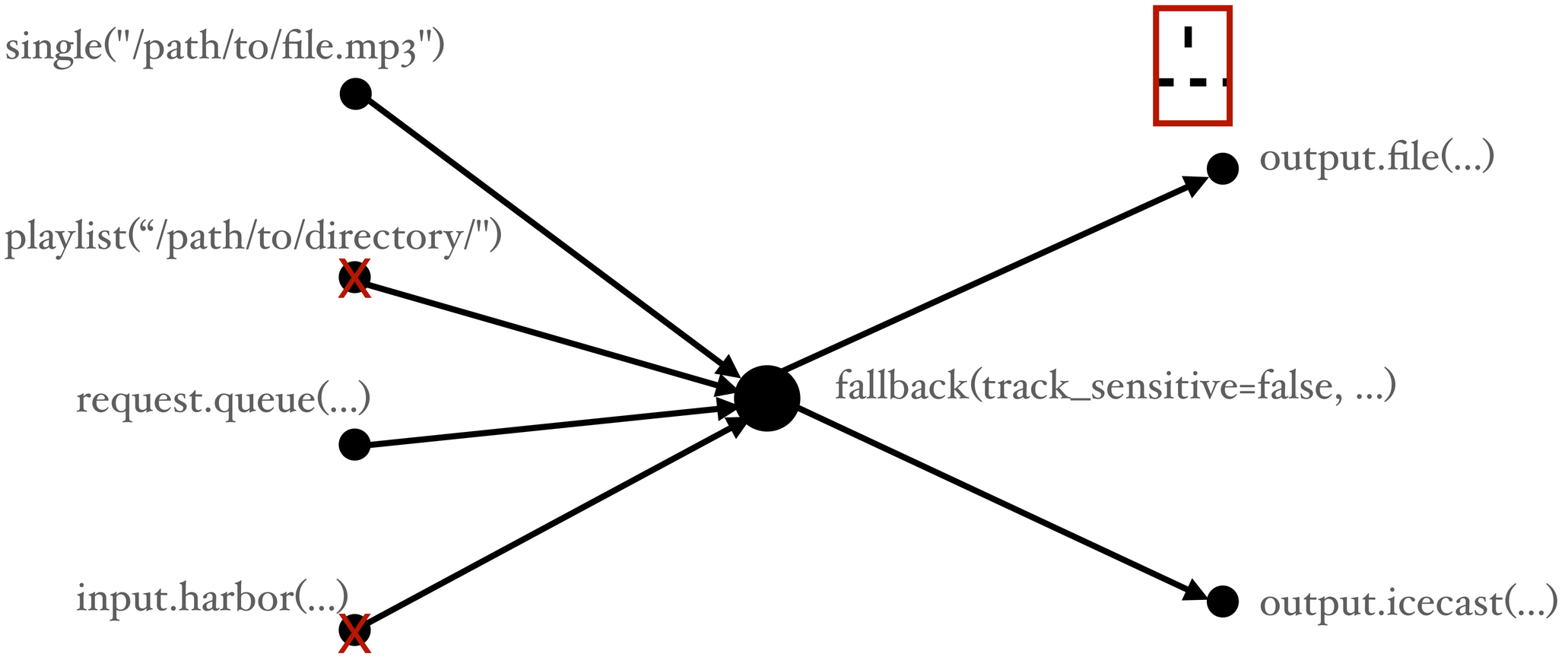
FFmpeg integration

Anything is possible!



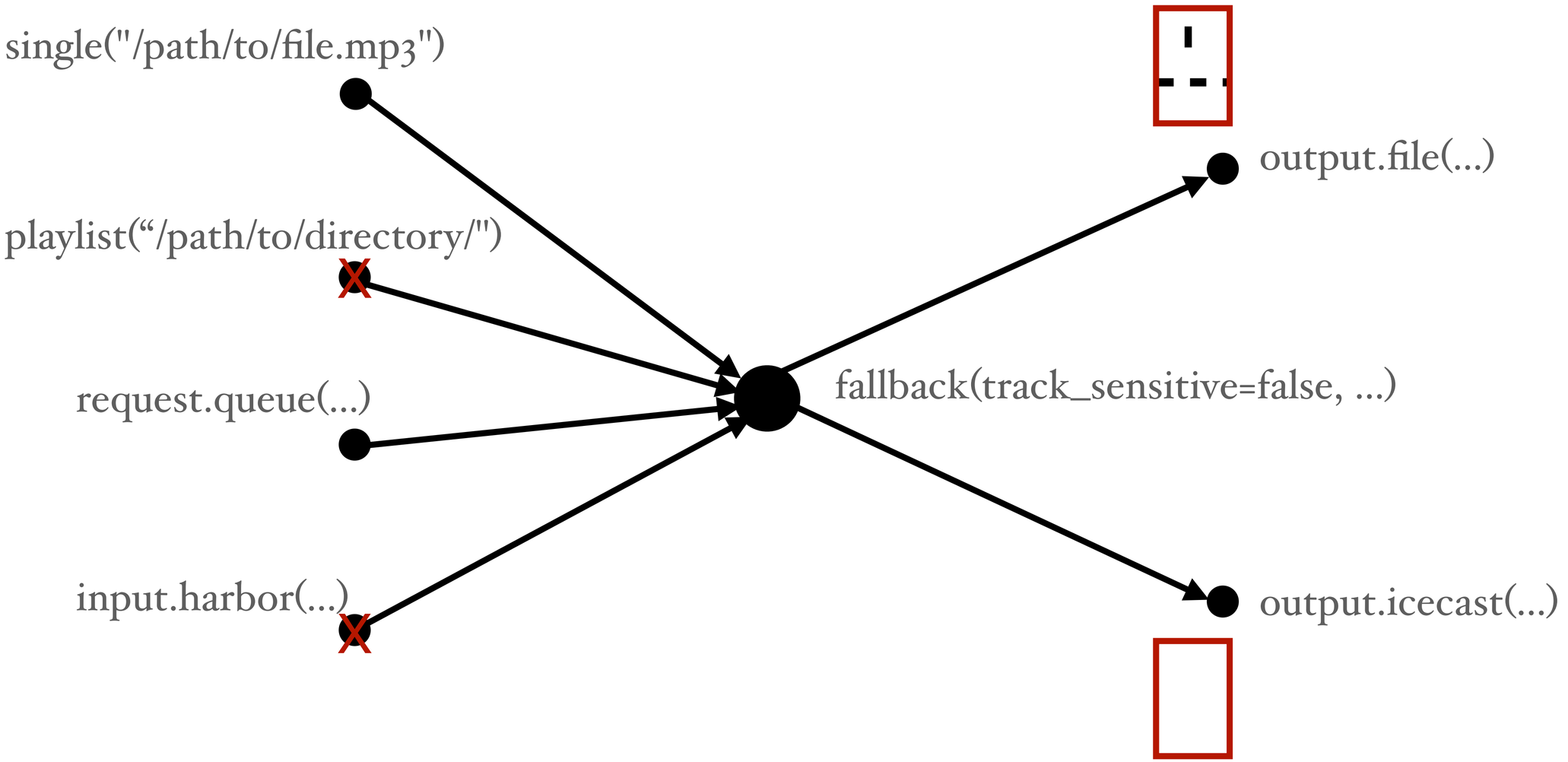
FFmpeg integration

Anything is possible!



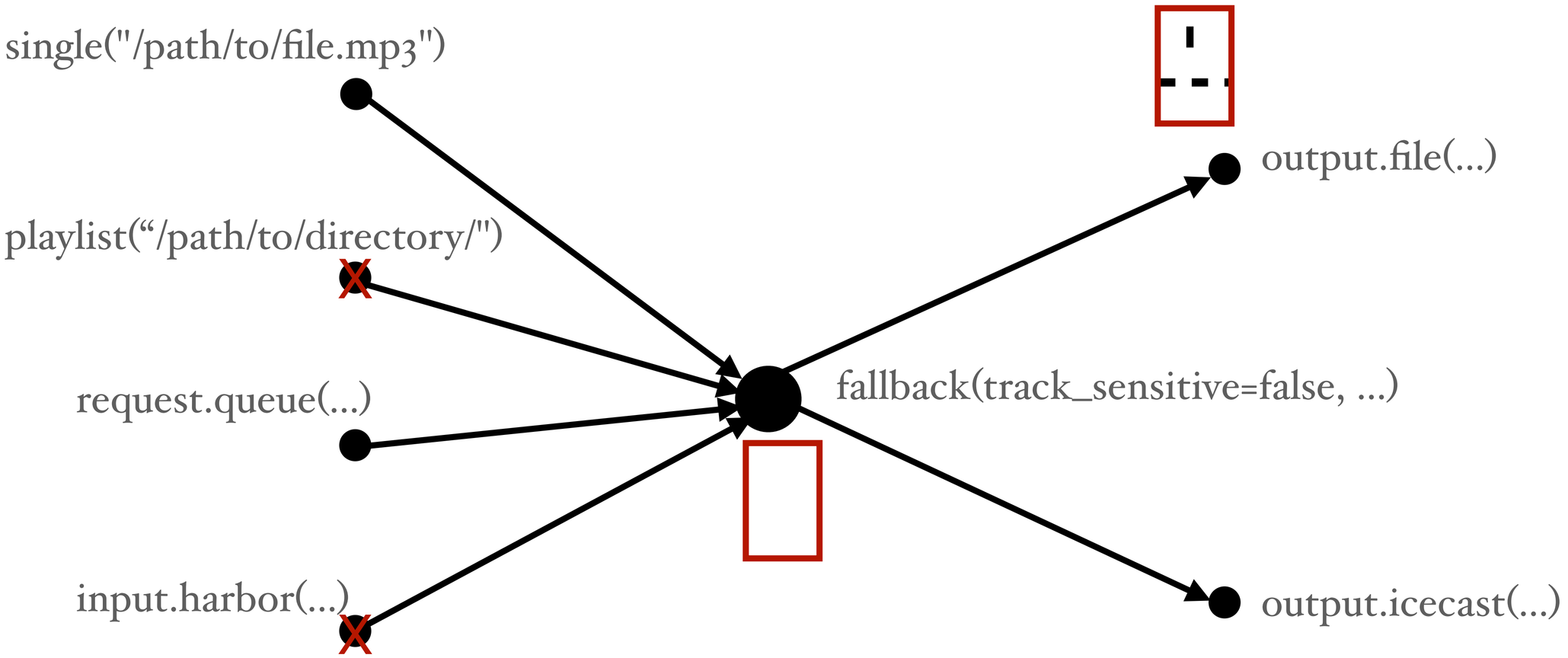
FFmpeg integration

Anything is possible!



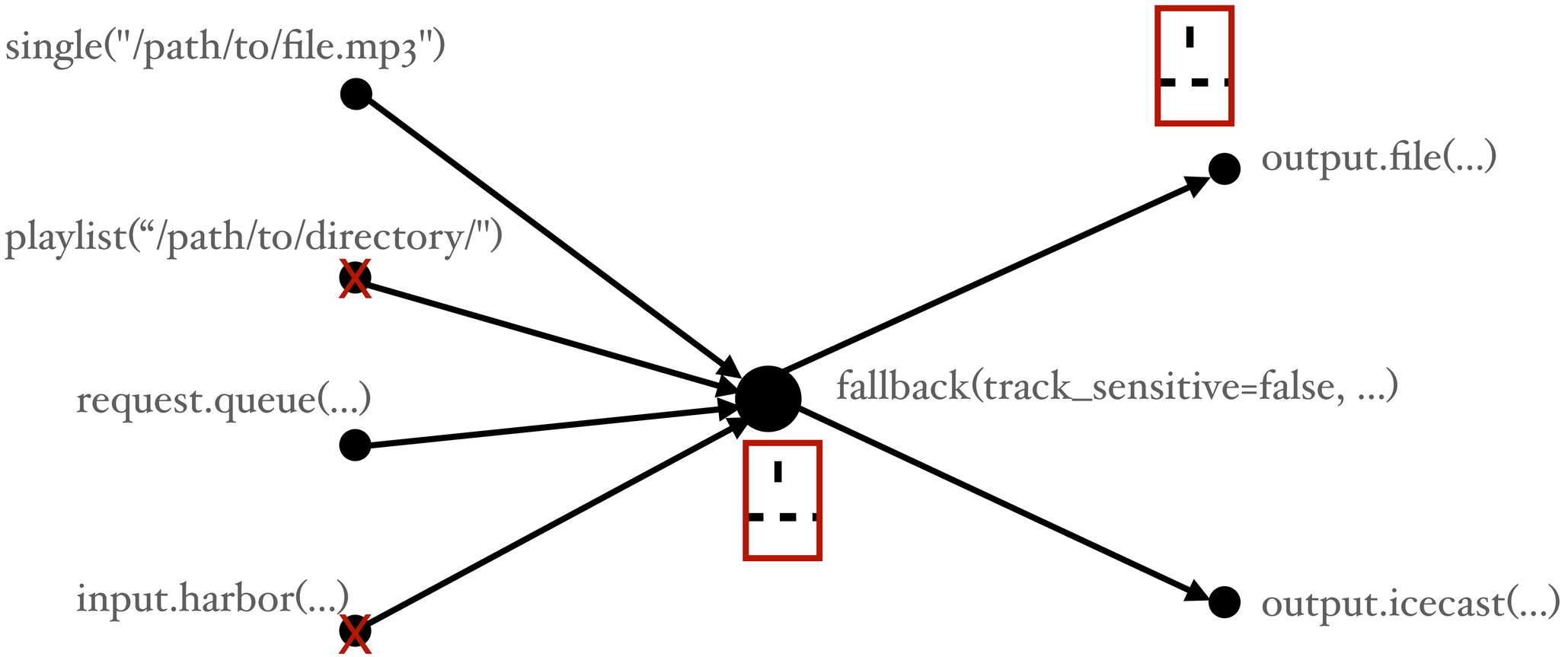
FFmpeg integration

Anything is possible!



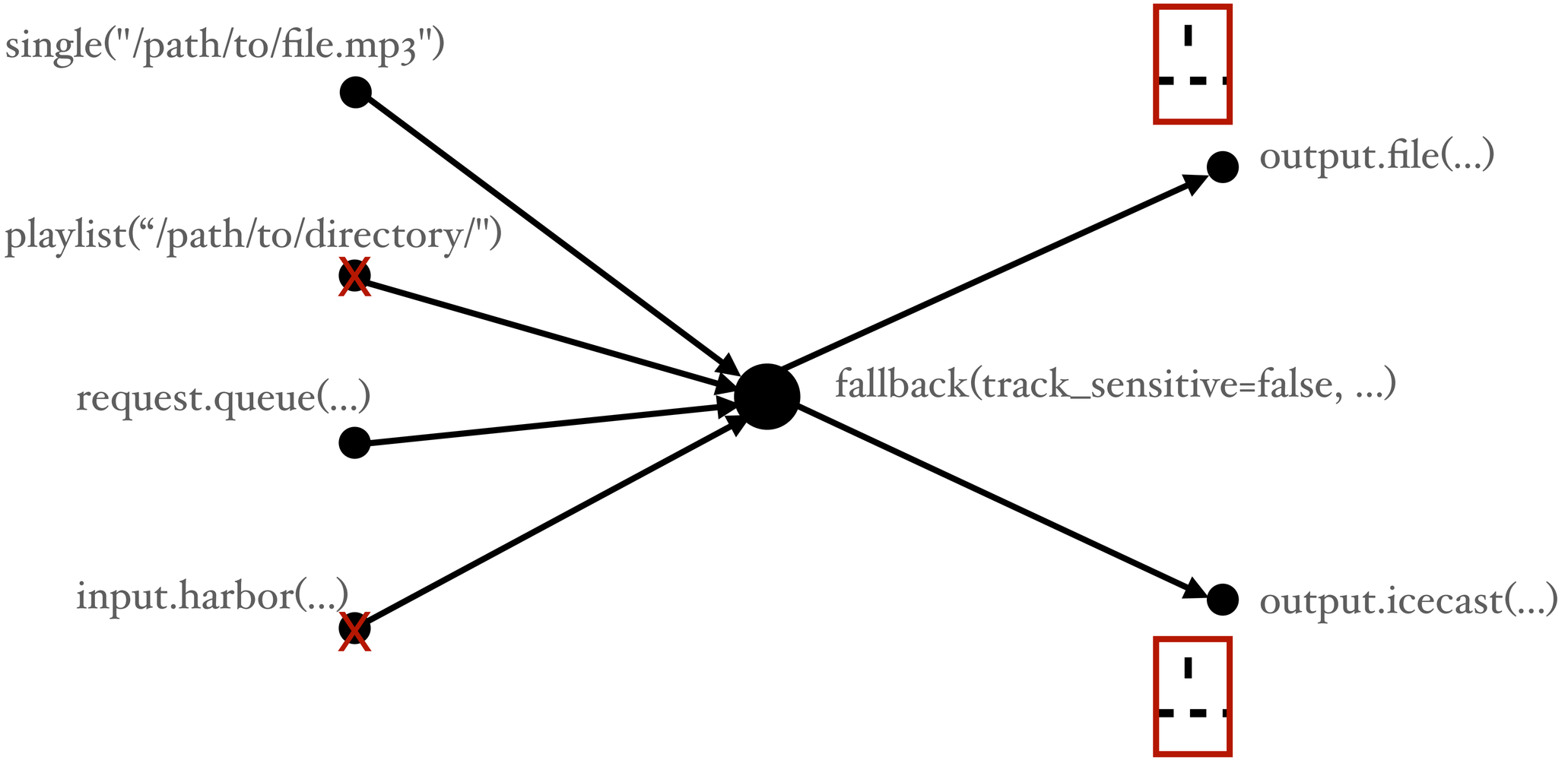
FFmpeg integration

Anything is possible!



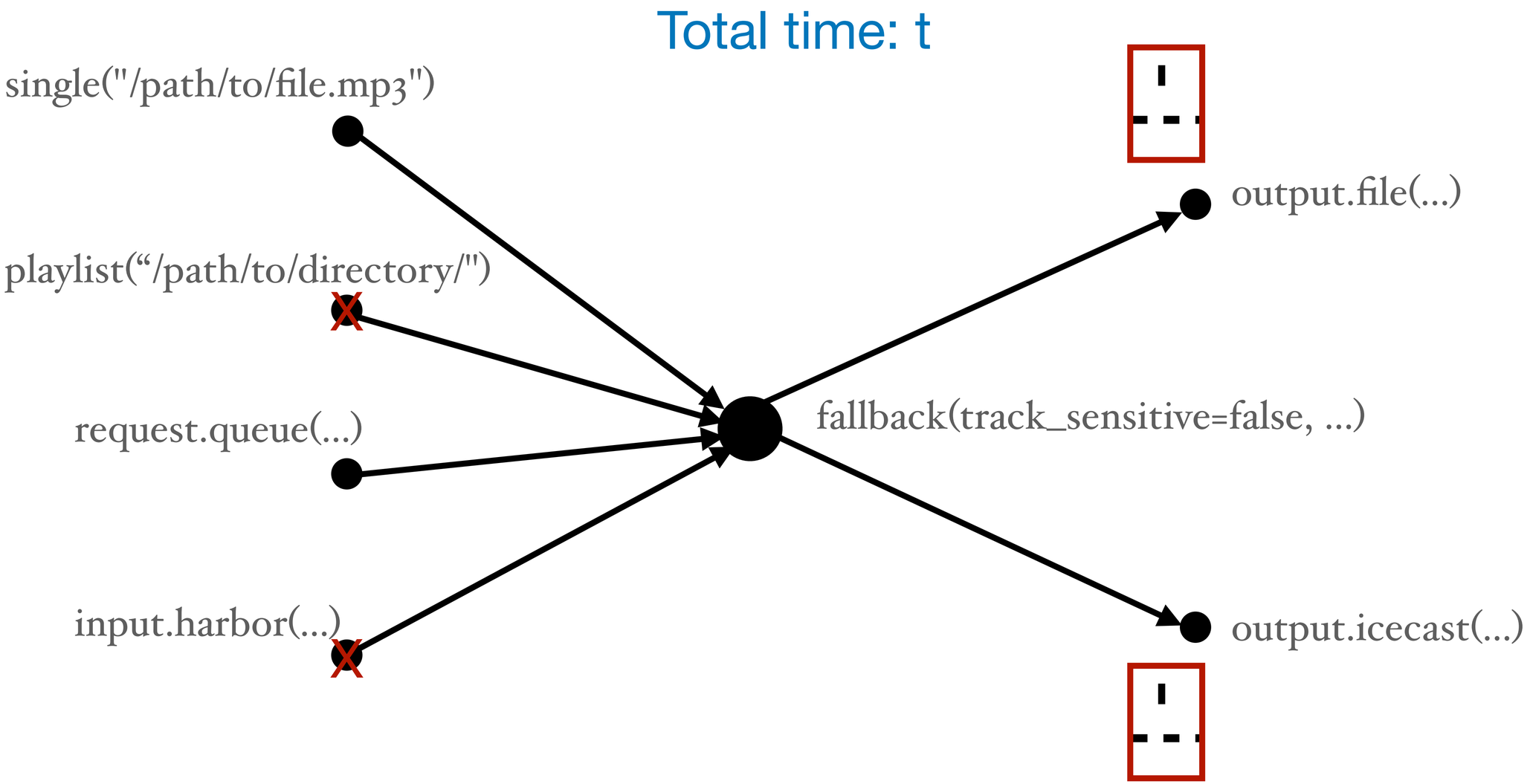
FFmpeg integration

Anything is possible!



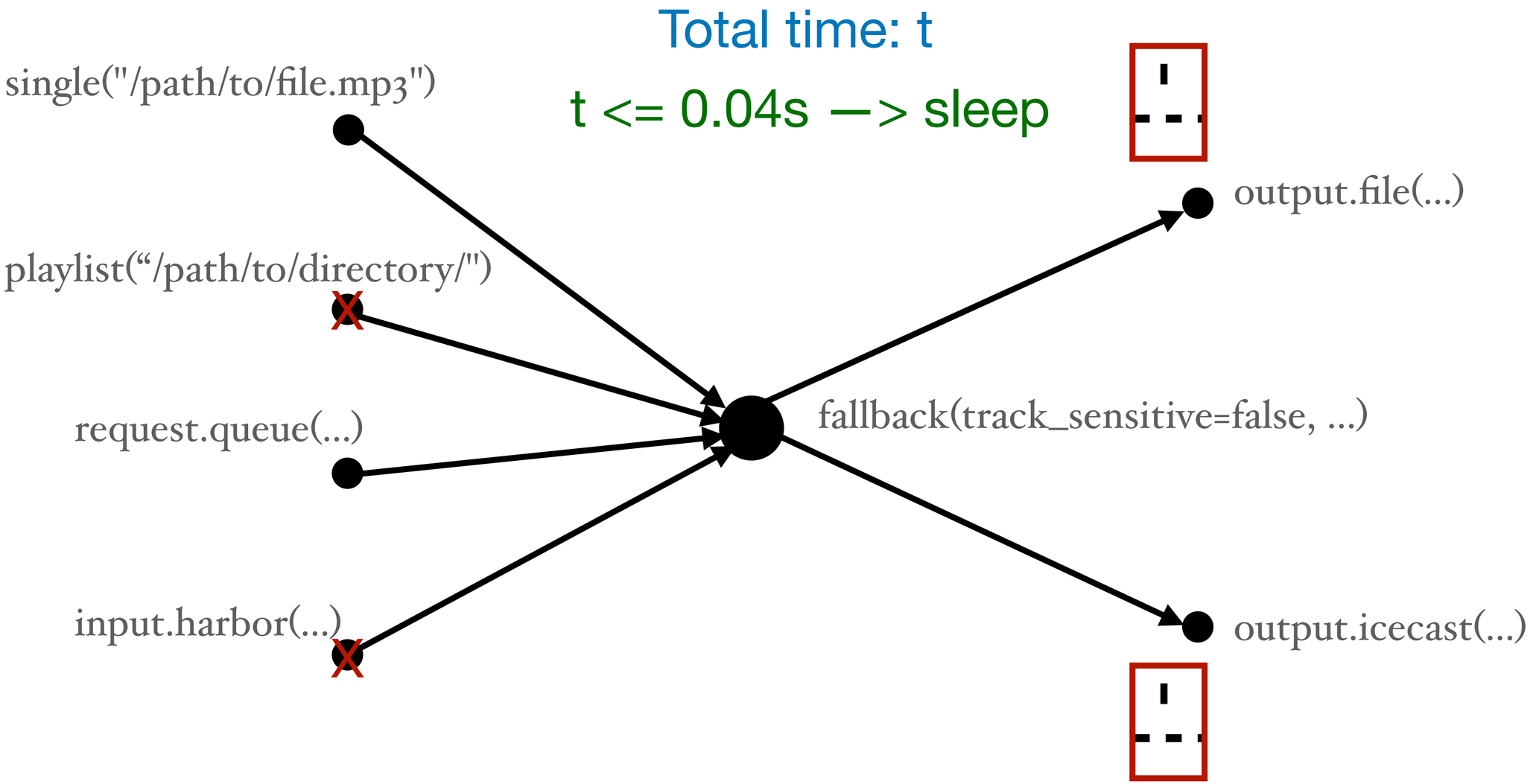
FFmpeg integration

Anything is possible!



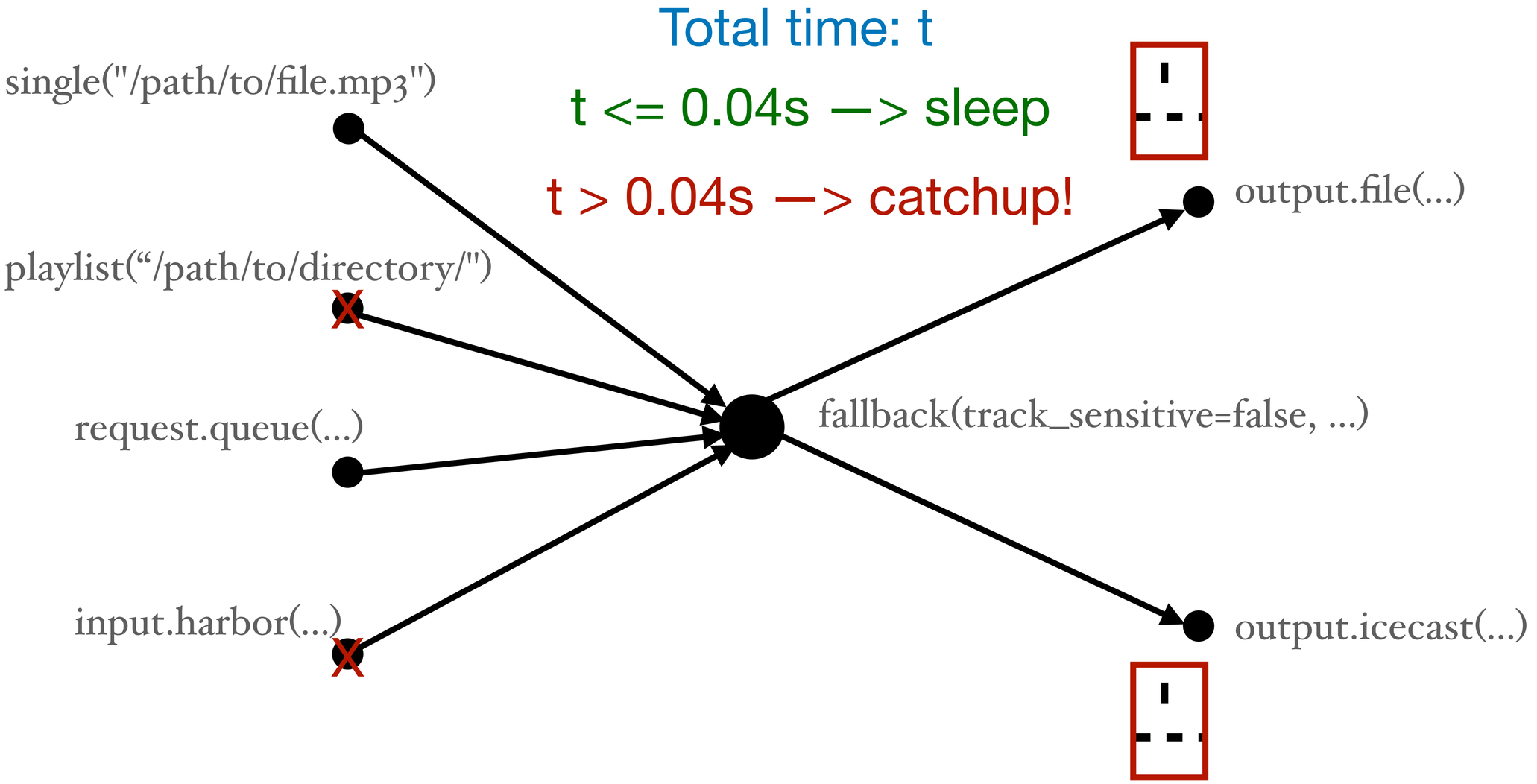
FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!



FFmpeg integration

Anything is possible!

```
live = input.rtmp(listen=true,"rtmp://0.0.0.0:19350/live/test")
live = ffmpeg.filter.bitstream.h264_mp4toannexb(live)

vod = single("/home/vod/sample.mp4")
vod = ffmpeg.filter.bitstream.h264_mp4toannexb(vod)

s = fallback(track_sensitive=false,[live,vod])

output.file(fallible=true,%ffmpeg(format="flv", %audio.copy, %video.copy), "output.mp4", s)
```

Future Development & Challenges

Future Development & Challenges

- 2.2.x release: tracks

Future Development & Challenges

- 2.2.x release: tracks

```
# Encode a source's audio and video content
# @category Source / Conversion
def ffmpeg.encode.audio_video(~id=null("ffmpeg.encode.audio_video"), f, s) =
  let {audio, video} = source.tracks(s)
  audio = track.ffmpeg.encode.audio(f, audio)
  video = track.ffmpeg.encode.video(f, video)
  source(id=id, {
    track_marks = track.track_marks(audio),
    metadata    = track.metadata(audio),
    audio       = audio,
    video       = video
  })
end
```

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)
- Wasm/Javascript

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)
- Wasm/Javascript
- Long-term development?

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)
- Wasm/Javascript
- Long-term development?

Number of tests:
*Liquidsoap 1.4.4: ~25
*Liquidsoap 2.0.0: ~120
*Development branch: ~180

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)
- Wasm/Javascript
- Long-term development?

Number of tests:
*Liquidsoap 1.4.4: ~25
*Liquidsoap 2.0.0: ~120
*Development branch: ~180

CI, Rolling releases

Future Development & Challenges

- 2.2.x release: tracks
- 2.3.x (?) release: streaming & clock model, concurrency (OCaml 5!)
- Wasm/Javascript
- Long-term development?

Number of tests:
*Liquidsoap 1.4.4: ~25
*Liquidsoap 2.0.0: ~120
*Development branch: ~180

CI, Rolling releases

2 developers!

Thanks!

Thanks!

Questions?