Making videoconferencing easy, also in your application

Olivier Crête

Origins of Farsight

- aMSN
- Free IM had no VoIP
- Each proprietary IM protocol had its own thing
- Philippe's end of studies project
- Hacked into aMSN, gaim

Collabora

Original goals of Farsight

- Enable Free Software IM to do audio/video like other platforms
- Abstract the streaming from different IM protocols
- Hide the complexities of media streaming (of GStreamer)

Collabora

Farsight 2: Goals

- High level objects
- Core: interface, helper libraries
- RTP is reference, most standard, most capable
- Also, MSN, Yahoo, etc
- One GStreamer element per protocol
- Elegance
- Complete automated test coverage
- Good documentation

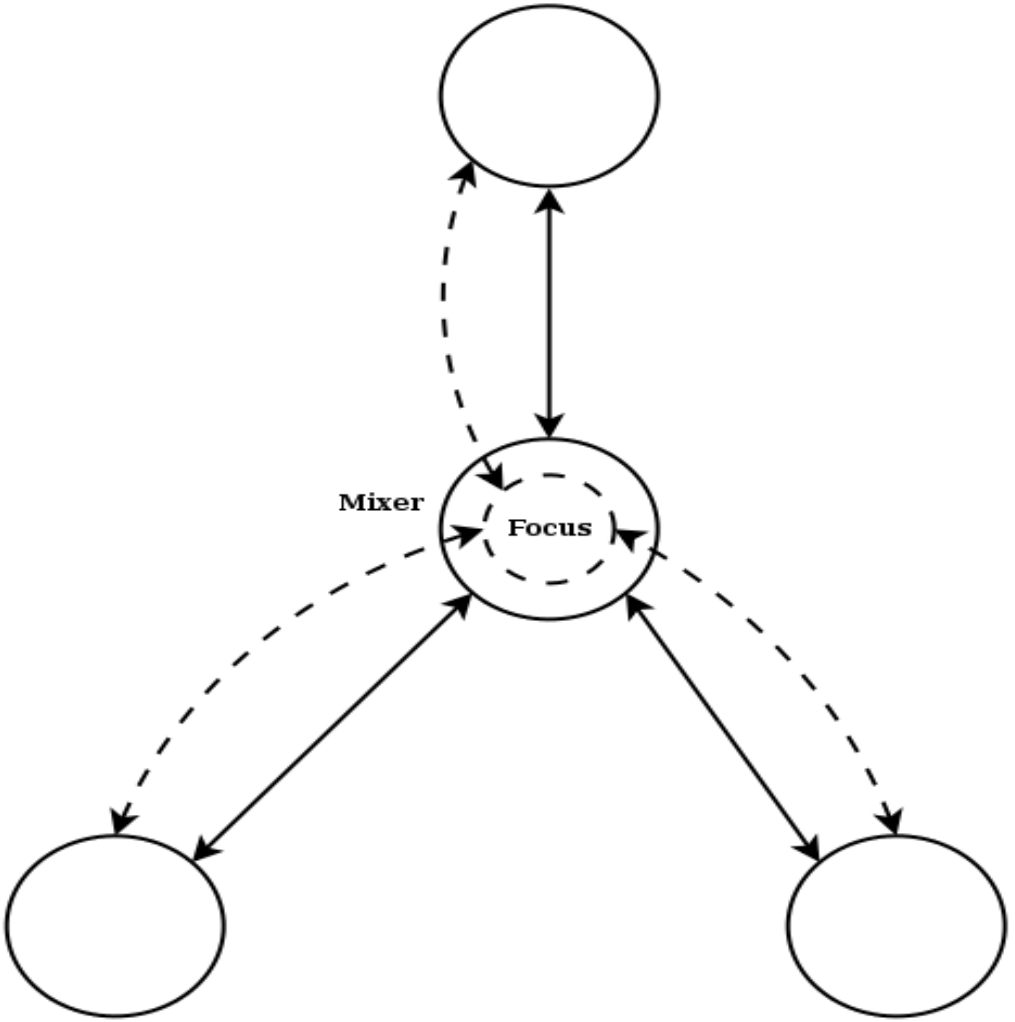
Collabora

Different conferencing models

- Media
  - Decentralised
    - Unicast
    - Multicast
  - Centralized (mixing server)
- Signalling
  - Centralized
  - Decentralized (crazy!)

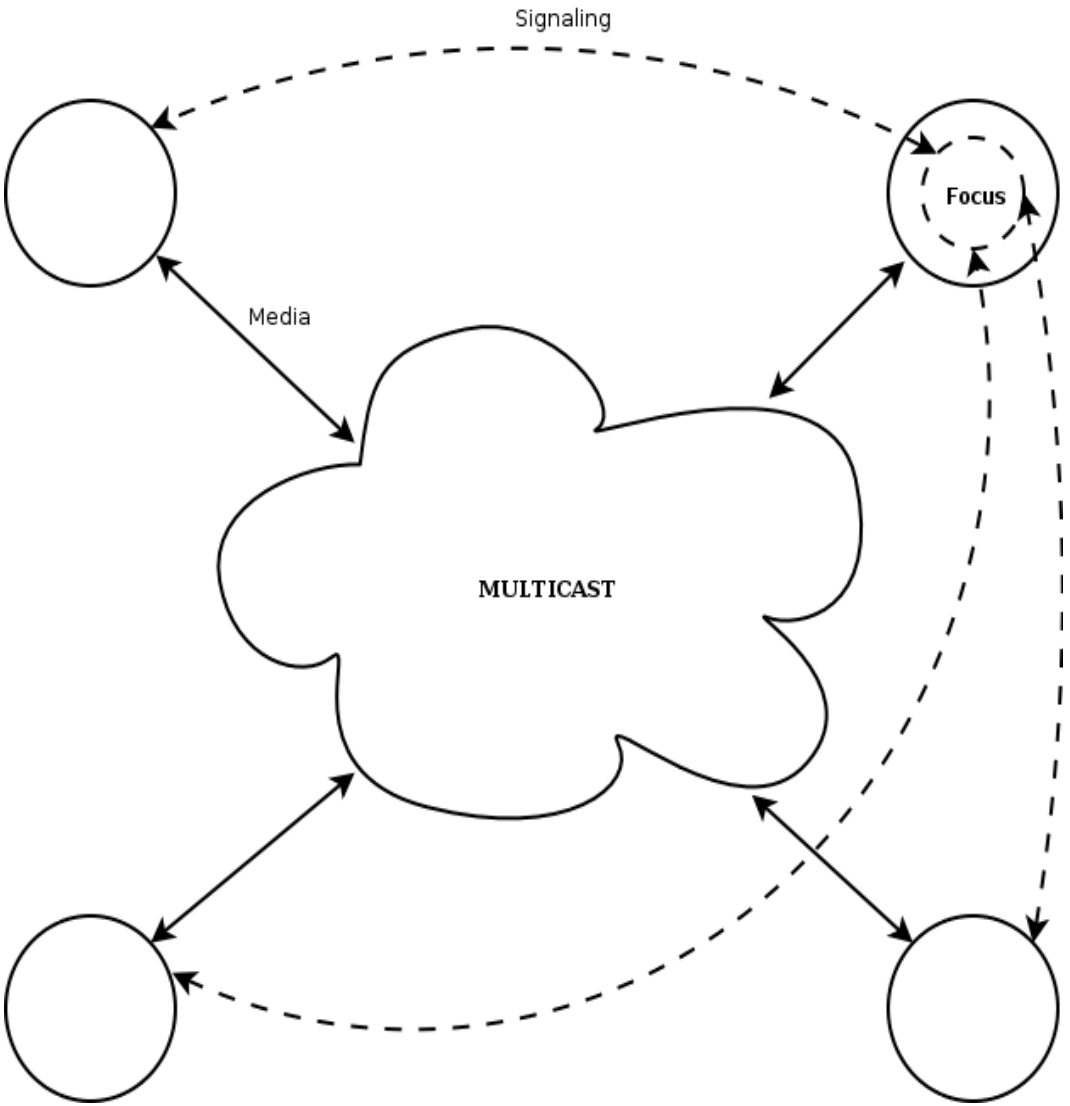
Collabora

Focused signalling
Centralised media



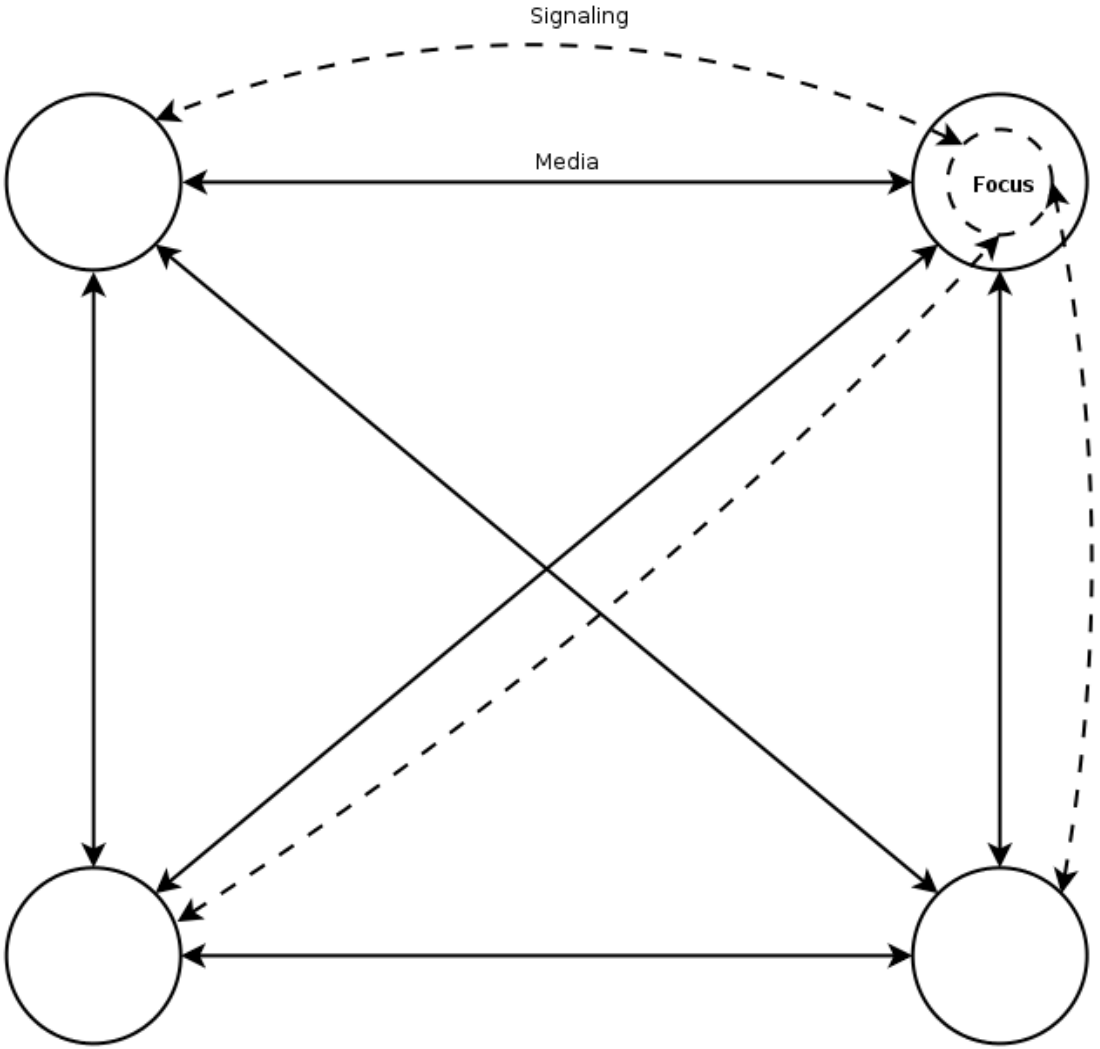
Collabora

Focused signalling
Distributed media (multicast)



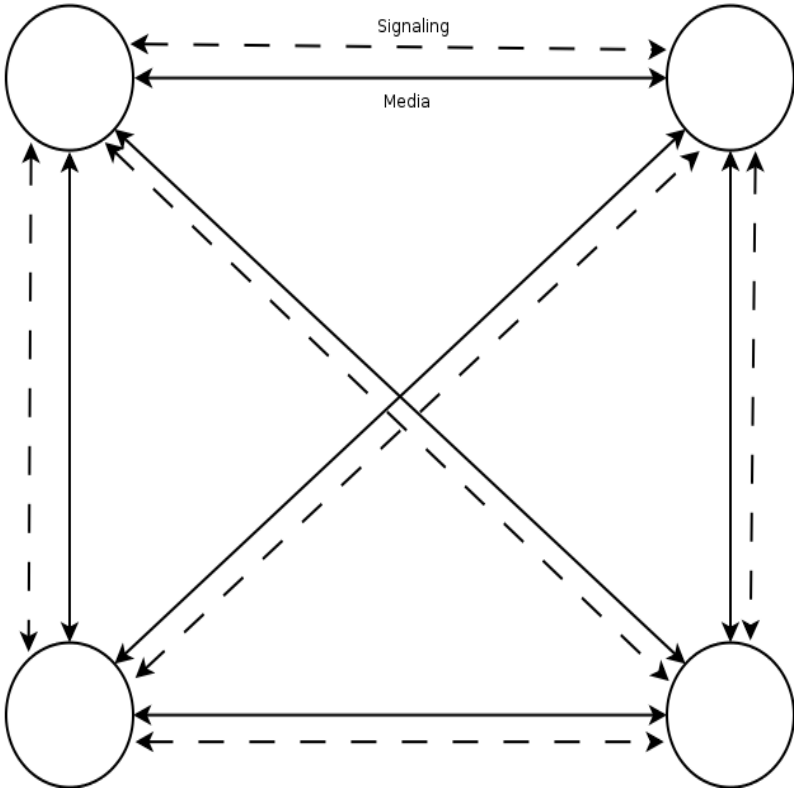
Collabora

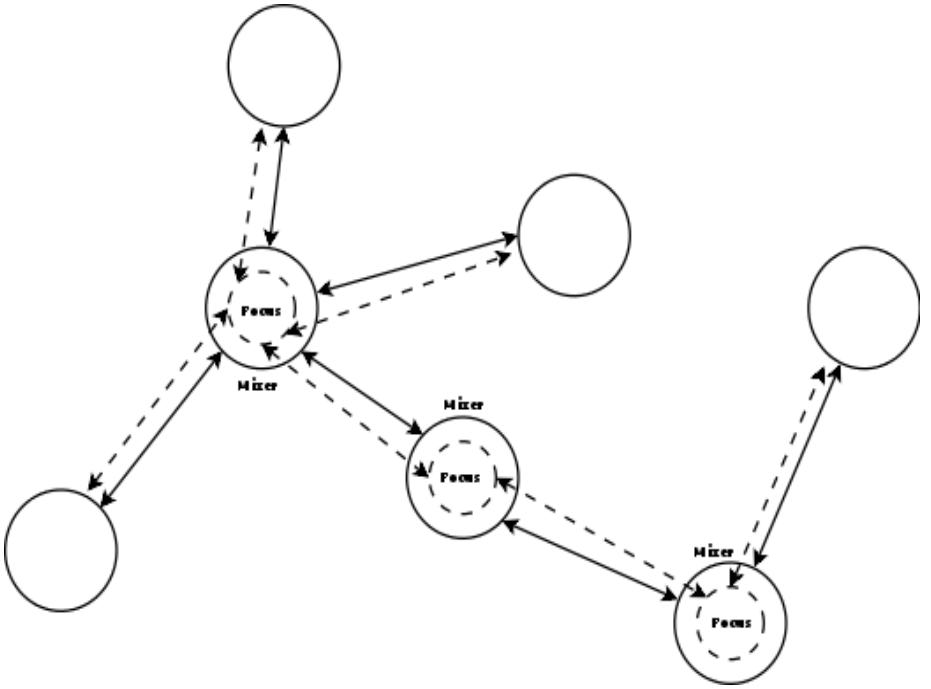
Focused signalling
Distributed media (multi-unicast)



Collabora

Distributed signalling
Crazy ideas!





Collabora

High level objects

- Codec
- Candidate
- Participant
  - One person with synchronized streams
- Session
- Stream
- Conference

Collabora

Session

- One type of media (audio, video, etc)
- One local media source
  - One microphone
  - One camera
  - File
  - etc
- Multiple stream from other participants
- RTP session

Collabora

Stream

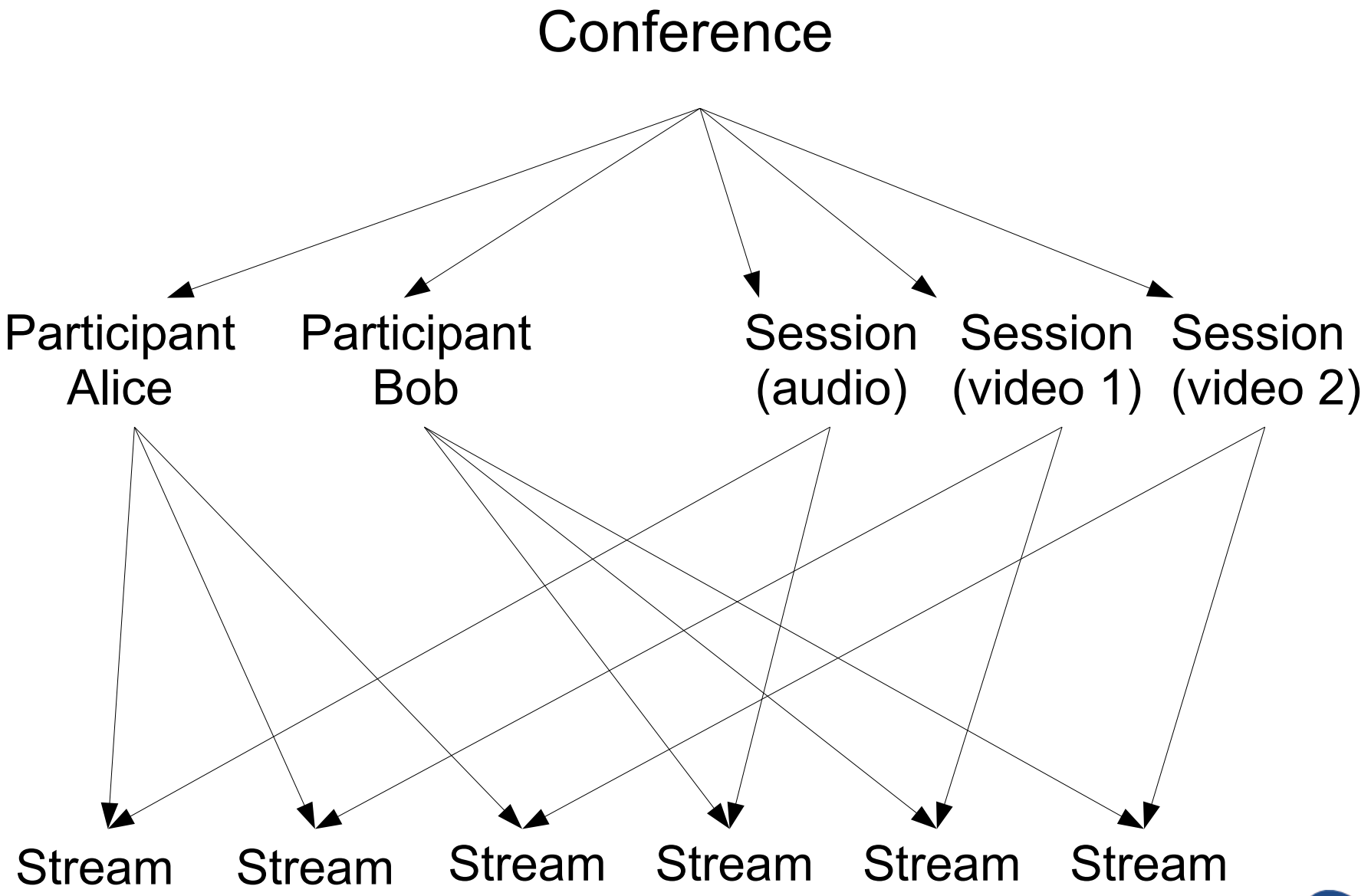
- One participant in one session
- Use for communication with participant
  - Codecs
  - Candidates
- Remote media comes out of here

Collabora

Conference

- The GStreamer element
- Multiple synchronized sessions
- Contains everything else

Collabora


Conference
Participant Alice
Participant Bob
Session (audio)
Session (video 1)
Session (video 2)
Stream
Stream
Stream
Stream
Stream
Stream

Collabora

New RTP plugin

- Keep good things from older versions
  - Codec detection
  - SDP Offer/Answer codec negotiation
  - GStreamer elements: DTMF, RTP payloaders, etc
- Use new GStreamer rtpmanager
  - Multi-party
  - Lip-sync
  - Complete RTP feature set
    - Including full RTCP, SSRC collision detection, etc

Transmitters

- Multi Unicast UDP (with STUN and UPnP)
- Multicast UDP
- Interactive Connection Establishment (ICE)
  - Standard (draft 19)
  - Google Talk
  - Windows Live Messenger 8.5 & 2009
  - Including TURN relay support
  - Proxies support (HTTP & SOCKS5)

Collabora

Current status

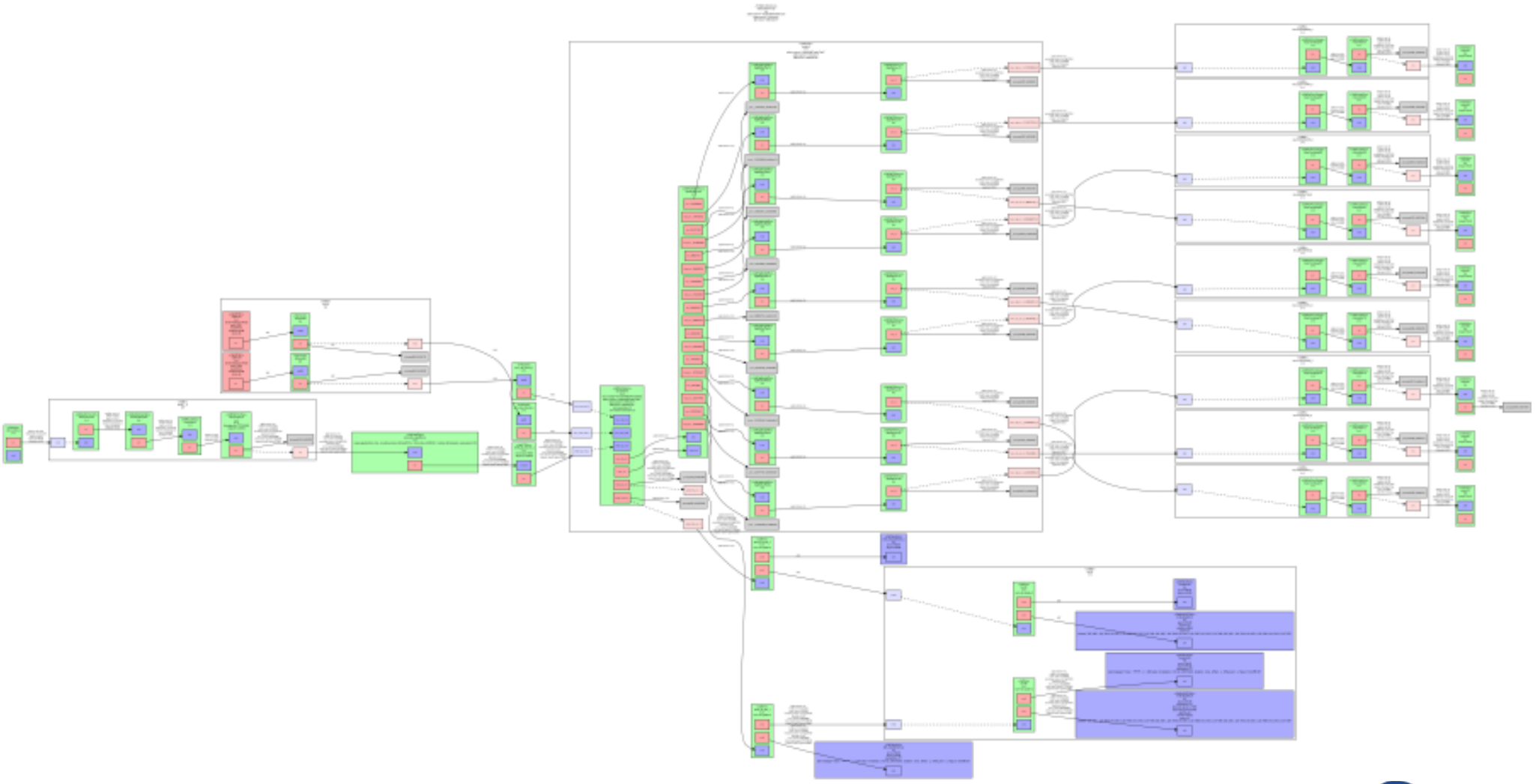
- Complete RTP implementation
  - Full DTMF (RTP events and sound)
  - Multi-party
  - Lip-sync
  - Python bindings
  - Automated tests
  - Unicast, Multicast, libnice (ICE) transmitters
  - Complex encoding pipelines profiles
- MSN Webcam plugin (almost ready)

Free codecs available

- Audio
  - Speex
  - AMR Narrowband & Wideband (patented)
  - PCMA / PCMU
  - Vorbis, MP3 & AAC (not great choices for VoIP)
- Video
  - Theora
  - Dirac (for HD)
  - H.263, H.264 (patented)

Collabora

10 way conference



Collabora

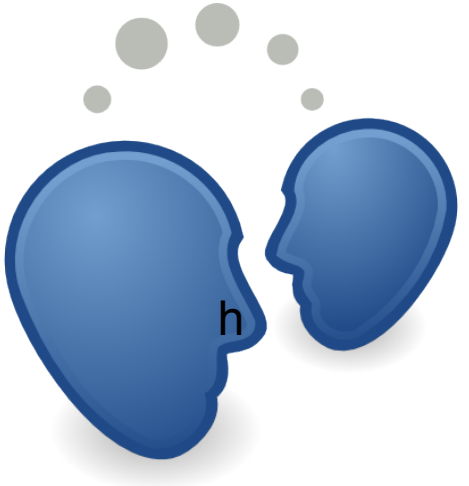
Example

```
import pygst; pygst.require('0.10'); import farsight, gst, gobject, sys


loop = gobject.MainLoop()
pipeline = gst.Pipeline()


conference = gst.element_factory_make ("fsrtpconference")
pipeline.add (conference)


session = conference.new_session (farsight.MEDIA_TYPE_VIDEO)
participant = conference.new_participant (“”)
stream = session.new_stream (participant, farsight.DIRECTION_BOTH, "multicast")
```

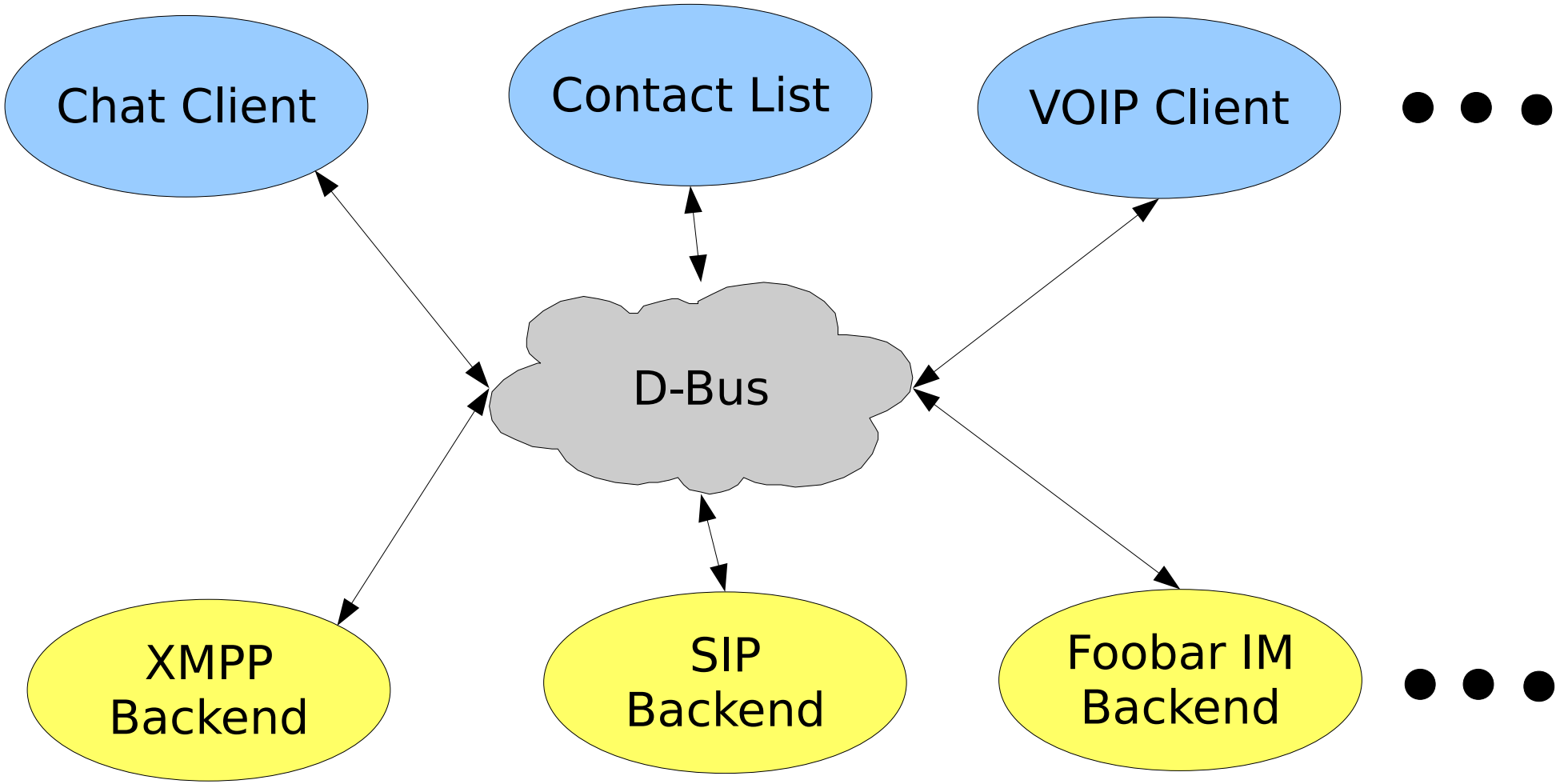
```
stream.set_remote_codecs([farsight.Codec(96, "H263-1998",
                                         farsight.MEDIA_TYPE_VIDEO,
                                         90000)])

candidate = farsight.Candidate()
candidate.ip = "224.0.0.110"
candidate.port = 3442
candidate.component_id = farsight.COMPONENT_RTP
candidate.proto = farsight.NETWORK_PROTOCOL_UDP
candidate.type = farsight.CANDIDATE_TYPE_MULTICAST
candidate.ttl = 1

Candidate2 = candidate.copy()
candidate.port = 3443
candidate.component_id = farsight.COMPONENT_RTCP
stream.set_remote_candidates ([candidate, candidate2])
```

```
 gst.parse_bin_from_description(sys.argv[3] + ' ! videoscale', True)
pipeline.add (videosource)
videosource.get_pad ("src").link(session.get_property ("sink-pad"))

def _src_pad_added (stream, pad, codec, pipeline):
    videosink = gst.element_factory_make ("xvimagesink")
    pipeline.add (videosink)
    videosink.set_state (gst.STATE_PLAYING)
    pad.link (videosink.get_pad ("sink"))

stream.connect ("src-pad-added", _src_pad_added, pipeline)
pipeline.set_state(gst.STATE_PLAYING)

loop.run()
```

PiTiVi integration

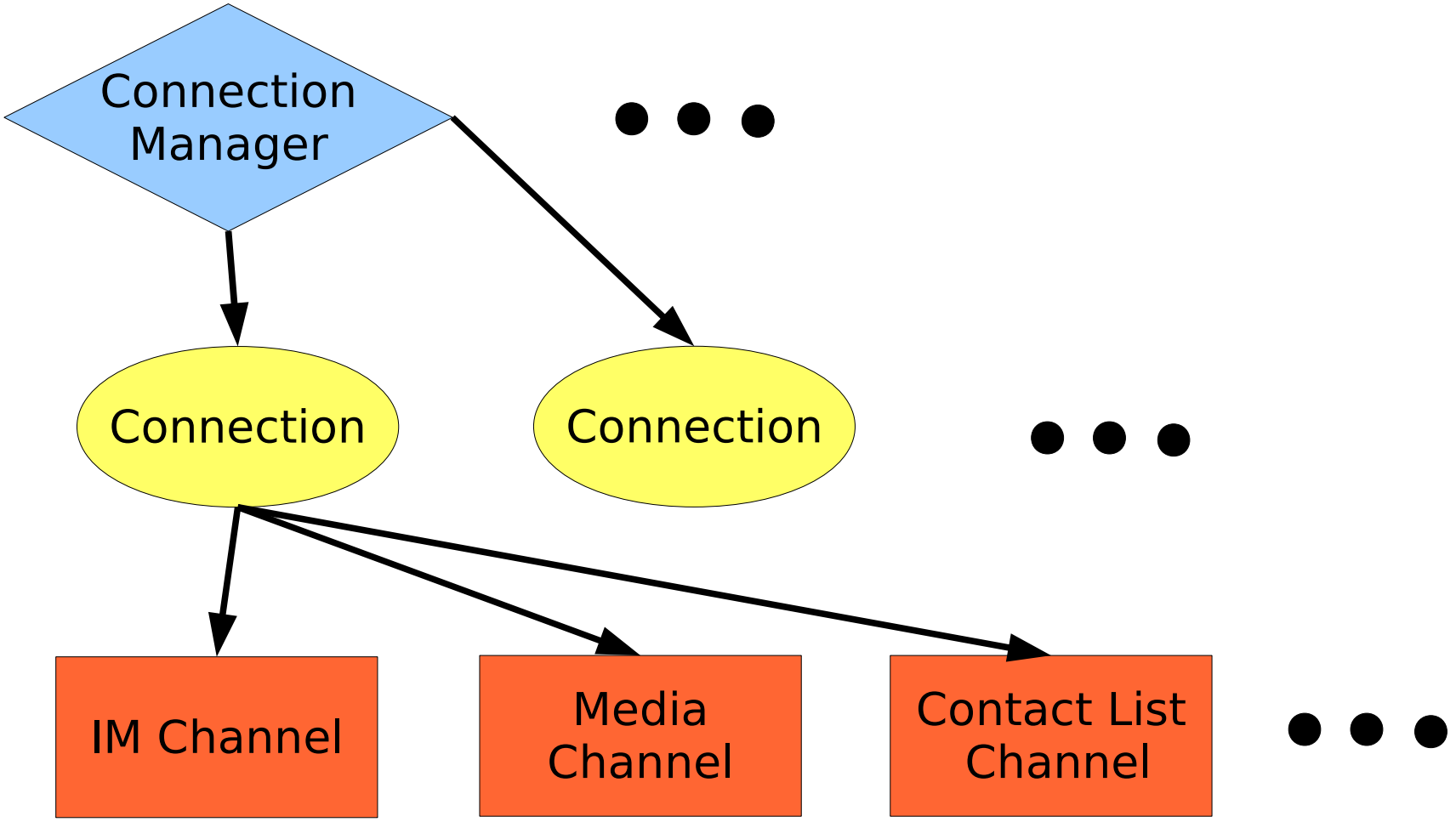


Telepathy: Overview



Collabora

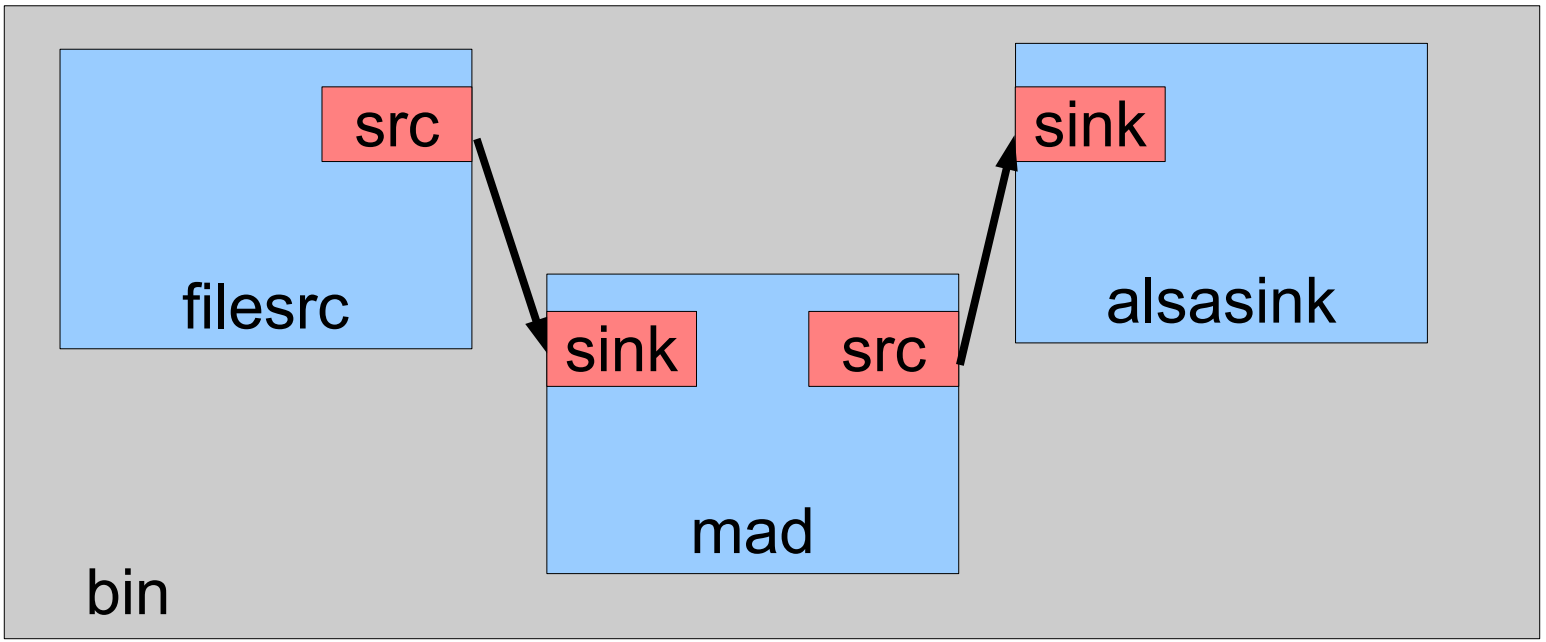
Telepathy: Concepts



Collabora

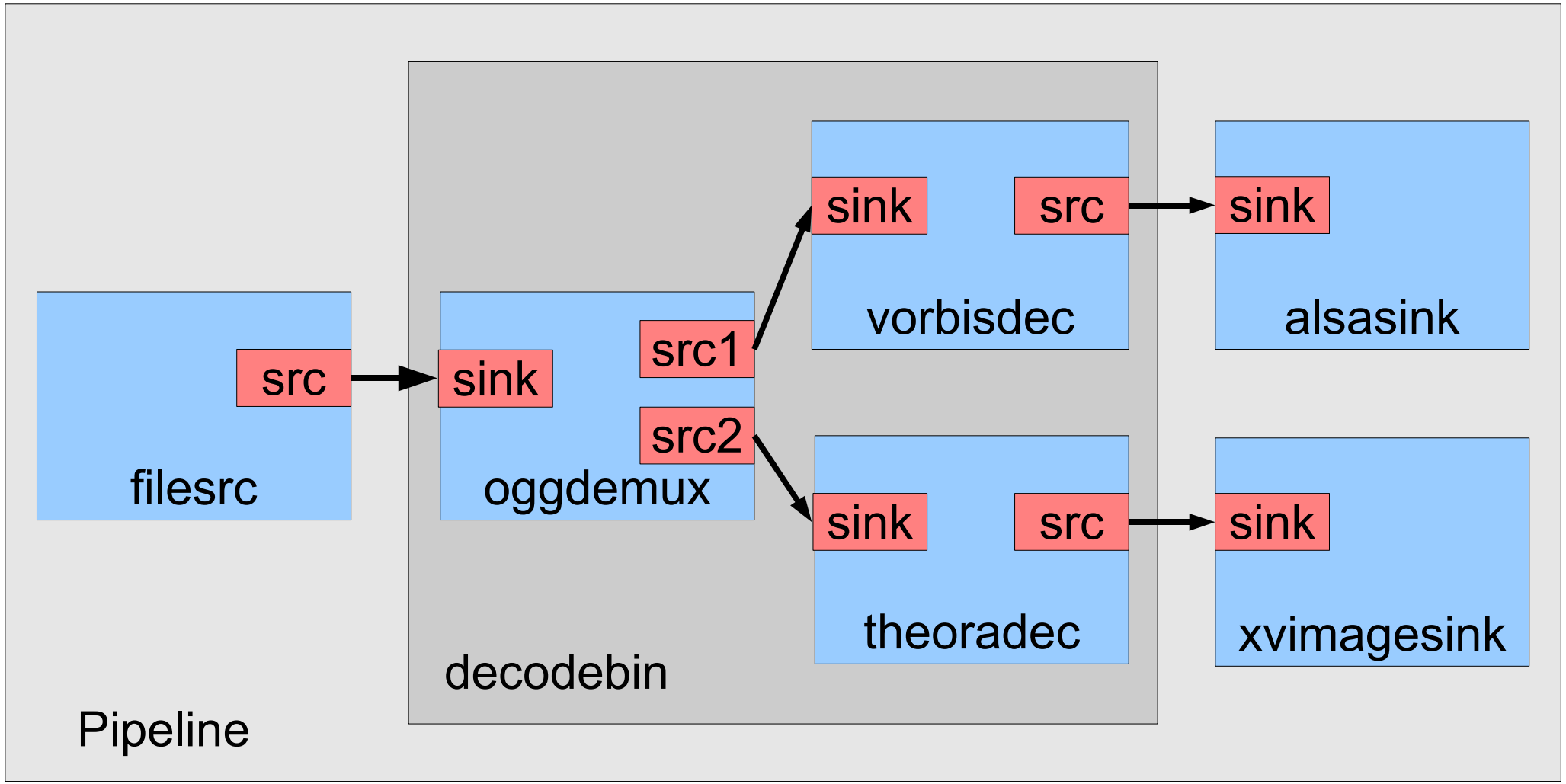
Basic GStreamer

- Elements
- Bins
- Pads



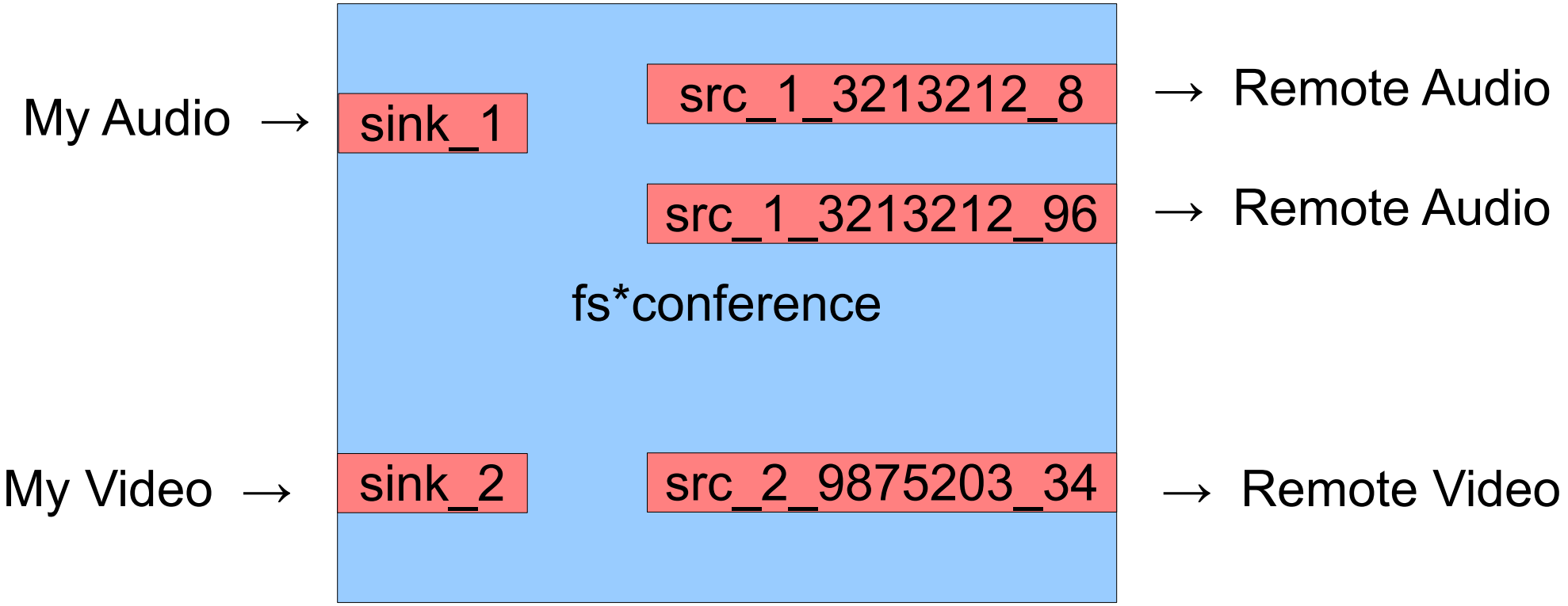
Collabora

More Complex Pipelines



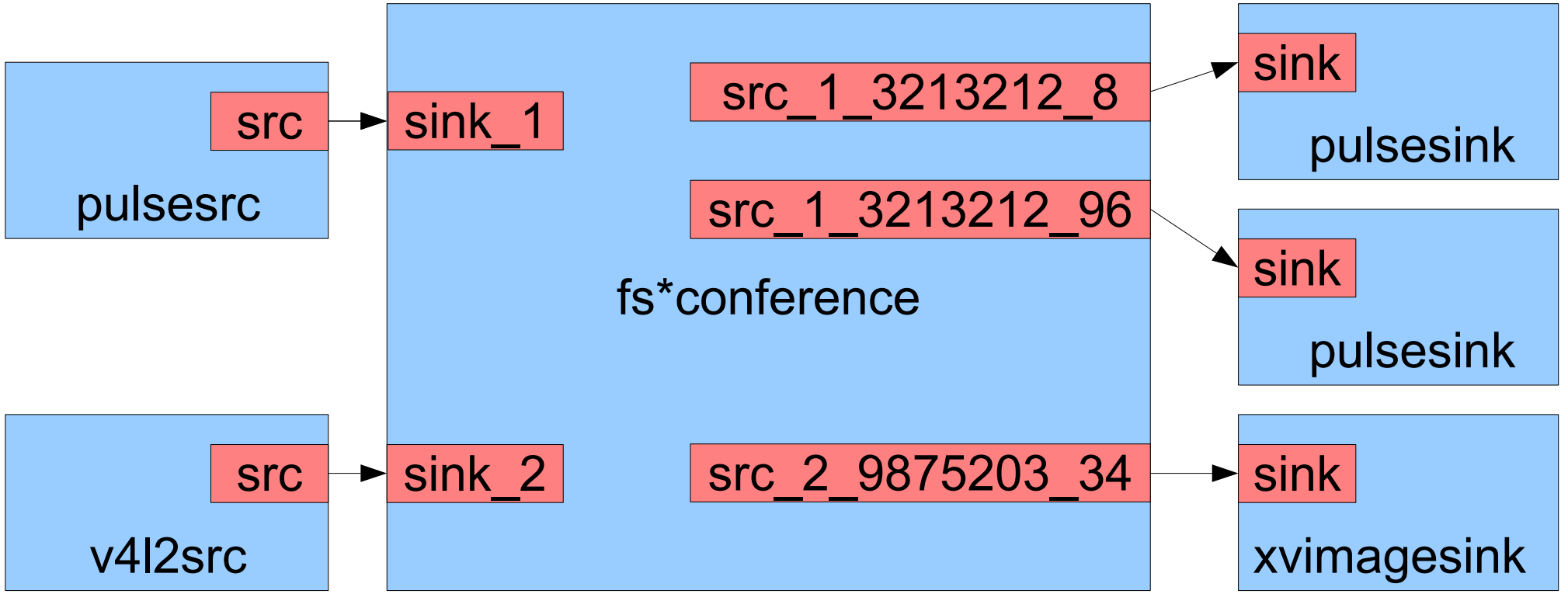
Collabora

Farsight 2 element



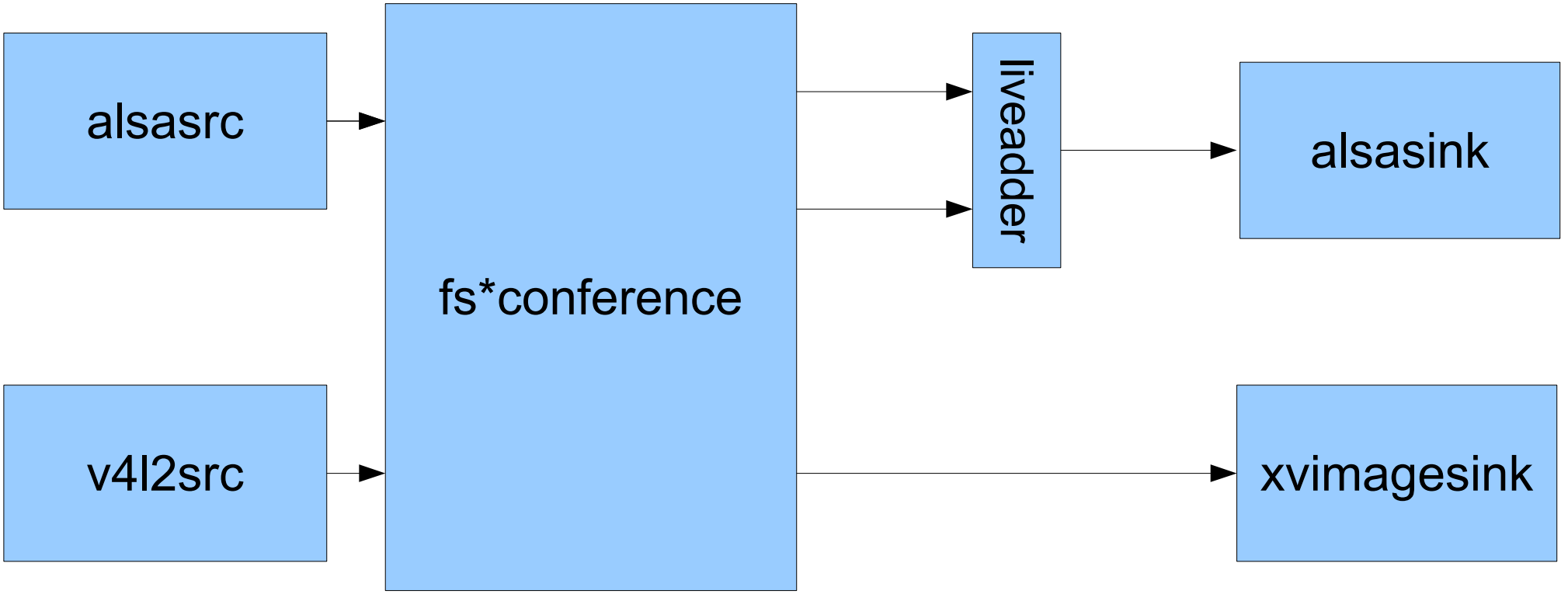
Collabora

Integrated Farsight 2: Simple Case



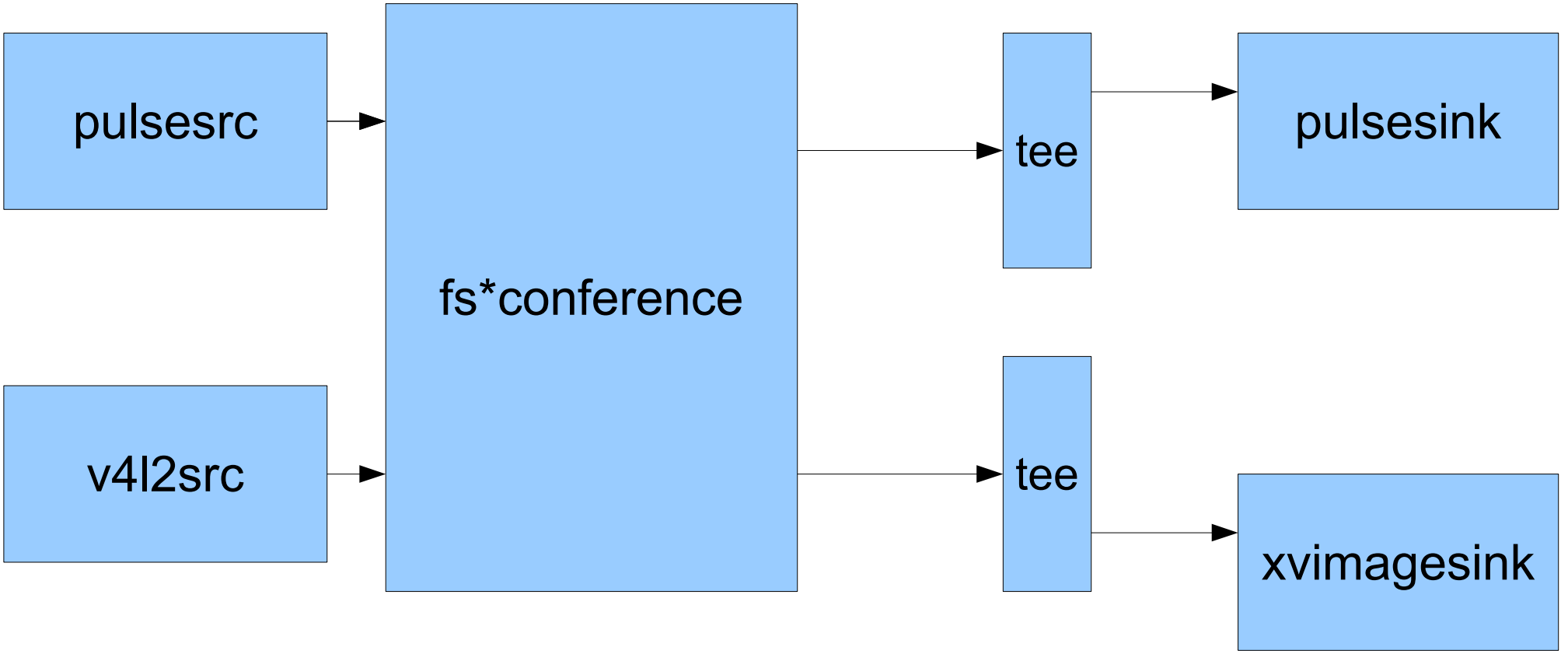
Collabora

Another simple case



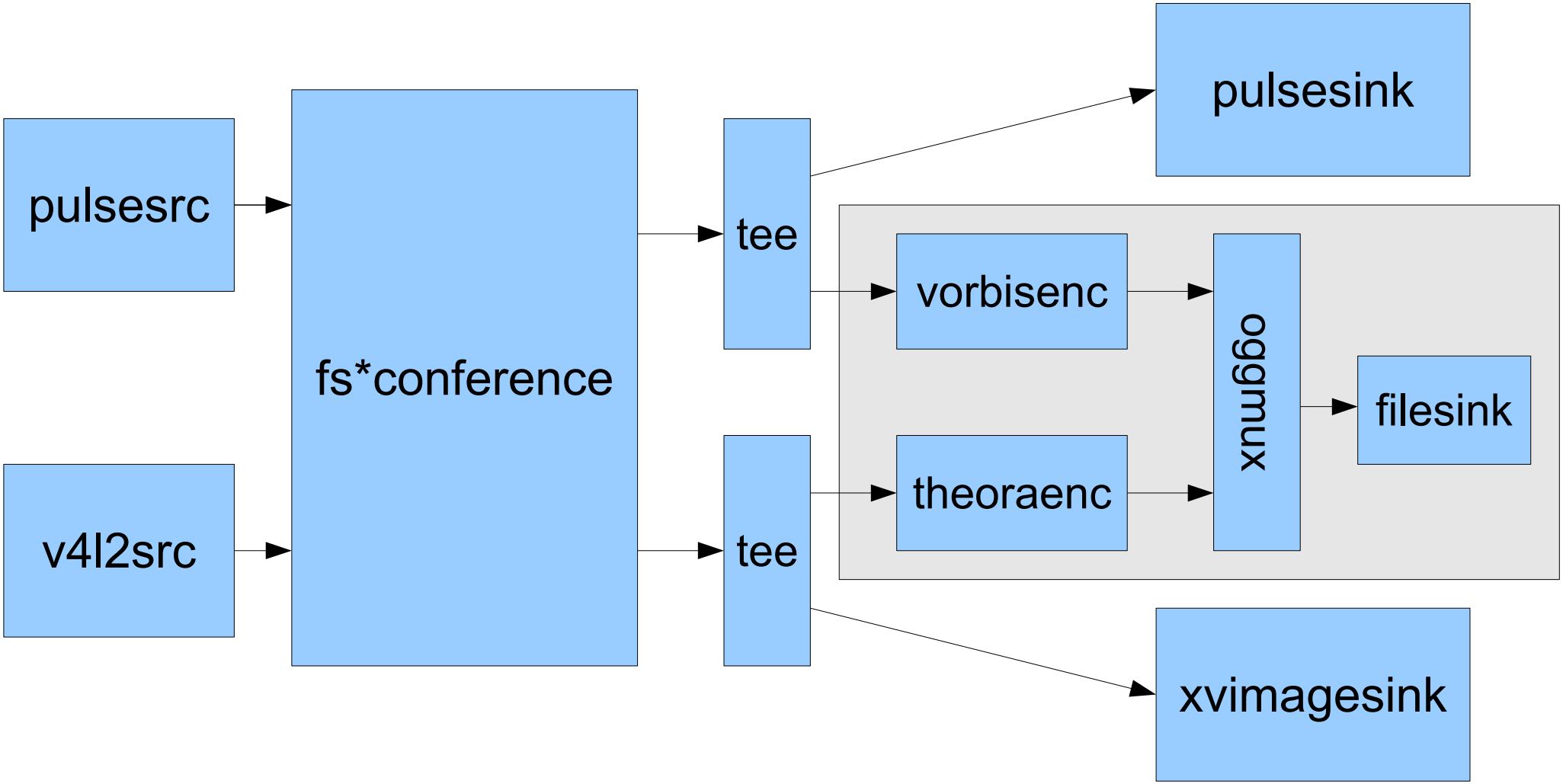
Collabora

Adding More stuff



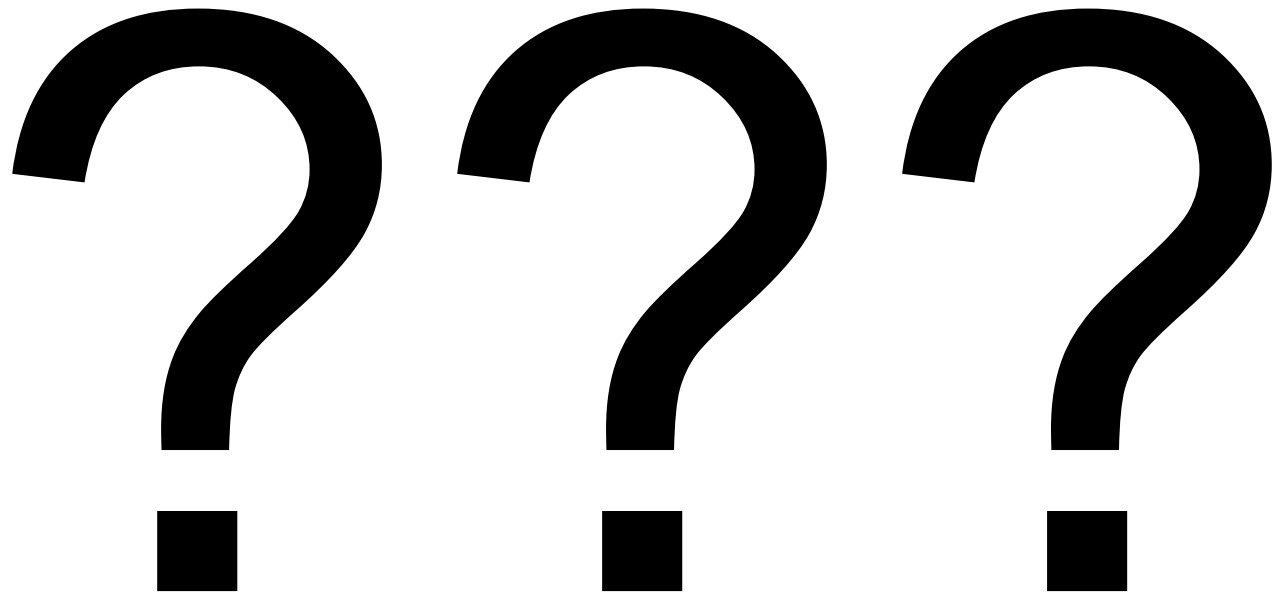
Collabora

Add recording



Collabora

Demo



Collabora

Current Users

- Pidgin-vv branch
- aMSN Voice chats
- Telepathy-Farsight library
  - Empathy 2.26
  - Maemo Stream Engine

Collabora

The Future

- Advanced RTP features
  - Secure RTP
  - RTCP AVPF (more feedback)
  - TCP Friendly Rate Control (TFRC)
- Use it in all Free clients so they can gain AV capabilities
  - Application integration using Telepathy
  - aMSN 2
  - PSI

Collabora

Muji

- Multiparty Jingle
- New XMPP extension to set up multi-party conferences using MUCs
- Works with regular unmodified XMPP servers
- Better than SIP
- Will make it possible to use all of Farsight's capabilities
- R&D prototype written with Twisted and Farsight 2

Collabora

Thank you

- Farsight is brought to you by Collabora

- Questions?

IRC: #farsight, #gstreamer, #telepathy @ FreeNode

http://farsight.freedesktop.org/
http://telepathy.freedesktop.org/
http://gstreamer.net
http://www.collabora.co.uk/