

# Farsight 2

## Video conferencing made easy

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

# 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)

# State of Farsight 1

- Current used in Telepathy Stream Engine
- On Nokia Internet Tablets
- Very stable
- Thin core, complex plugins
- Complex RTP plugin
- In maintenance-only mode
- Unmaintained plugins for MSN, Yahoo

# RTP plugin

- 1 to 1 audio & video calls
- Codec detection, simple negotiation
- Transmitters plugins
  - Unicast UDP
  - Google's pseudo-ICE (libjingle)
- Full DTMF (RTP events and sound)
- Comfort Noise (on Nokia Tablets)

# Limitations

- Only one to one calls
- No lip-sync
- Video support broke abstraction
- Hard to use with non-trivial GStreamer pipelines
- No sRTP
- Hacks for Nokia Tablets (DSP codecs, CN)
- No RTCP (No statistics)

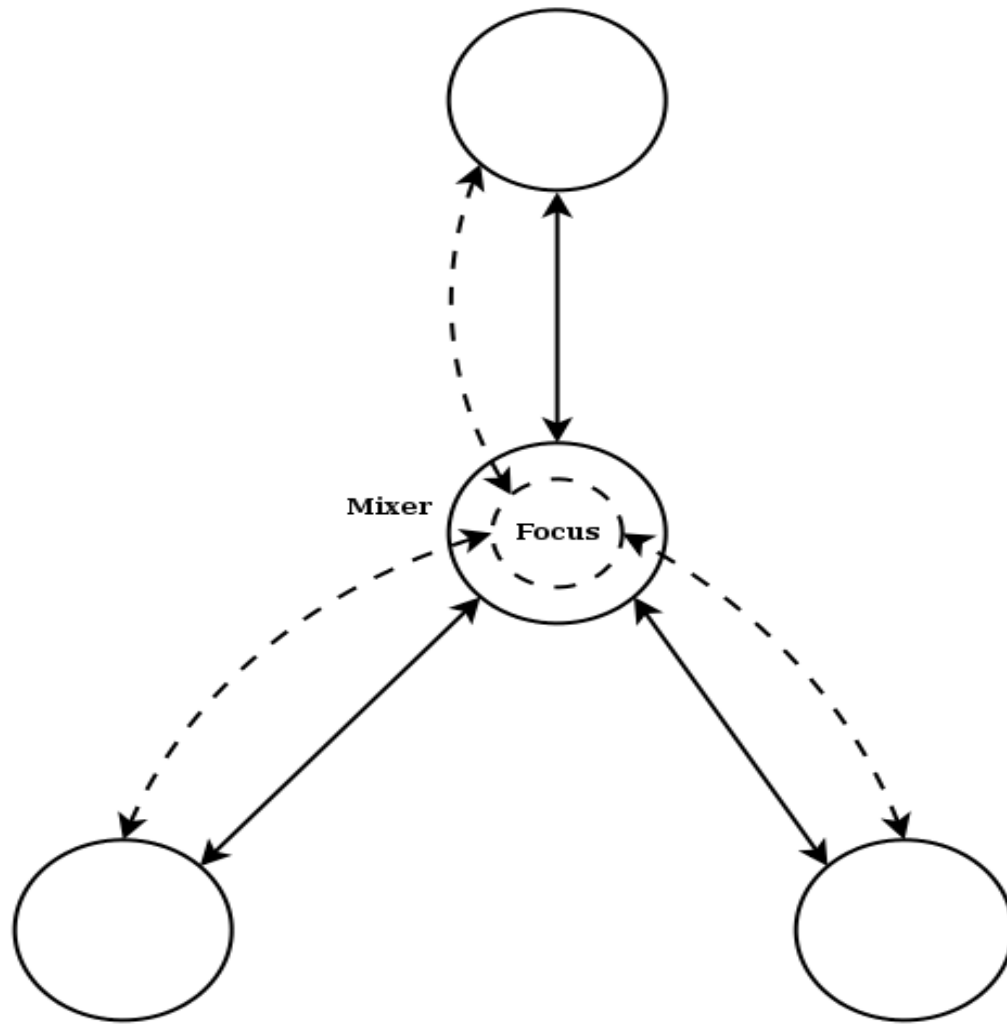
# Farsight 2: Goals

- 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

# Different conferencing models

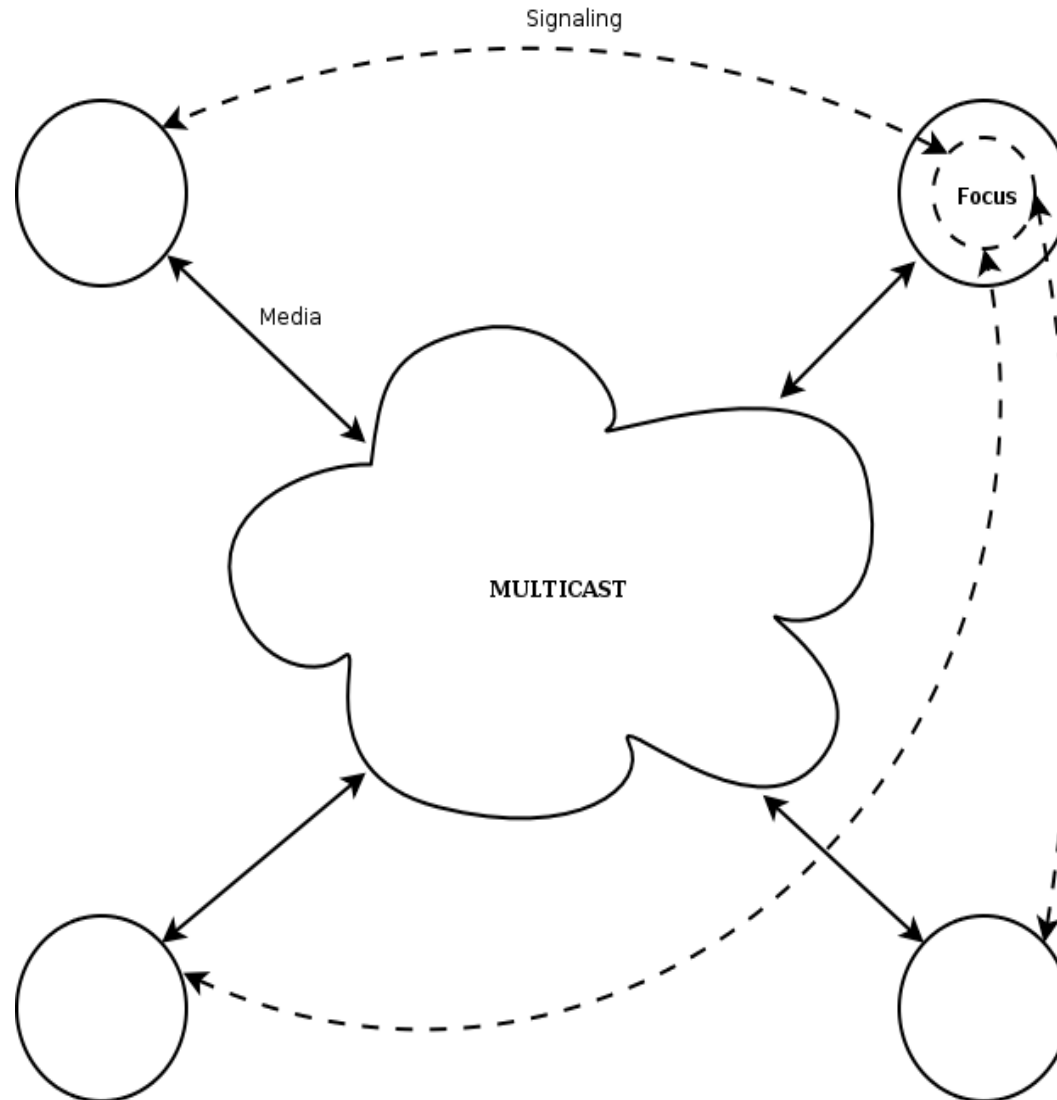
- Media
  - Decentralised
    - Unicast
    - Multicast
  - Centralized (mixing server)
- Signalling
  - Centralized
  - Decentralized (crazy!)

# Focused signalling Centralised media



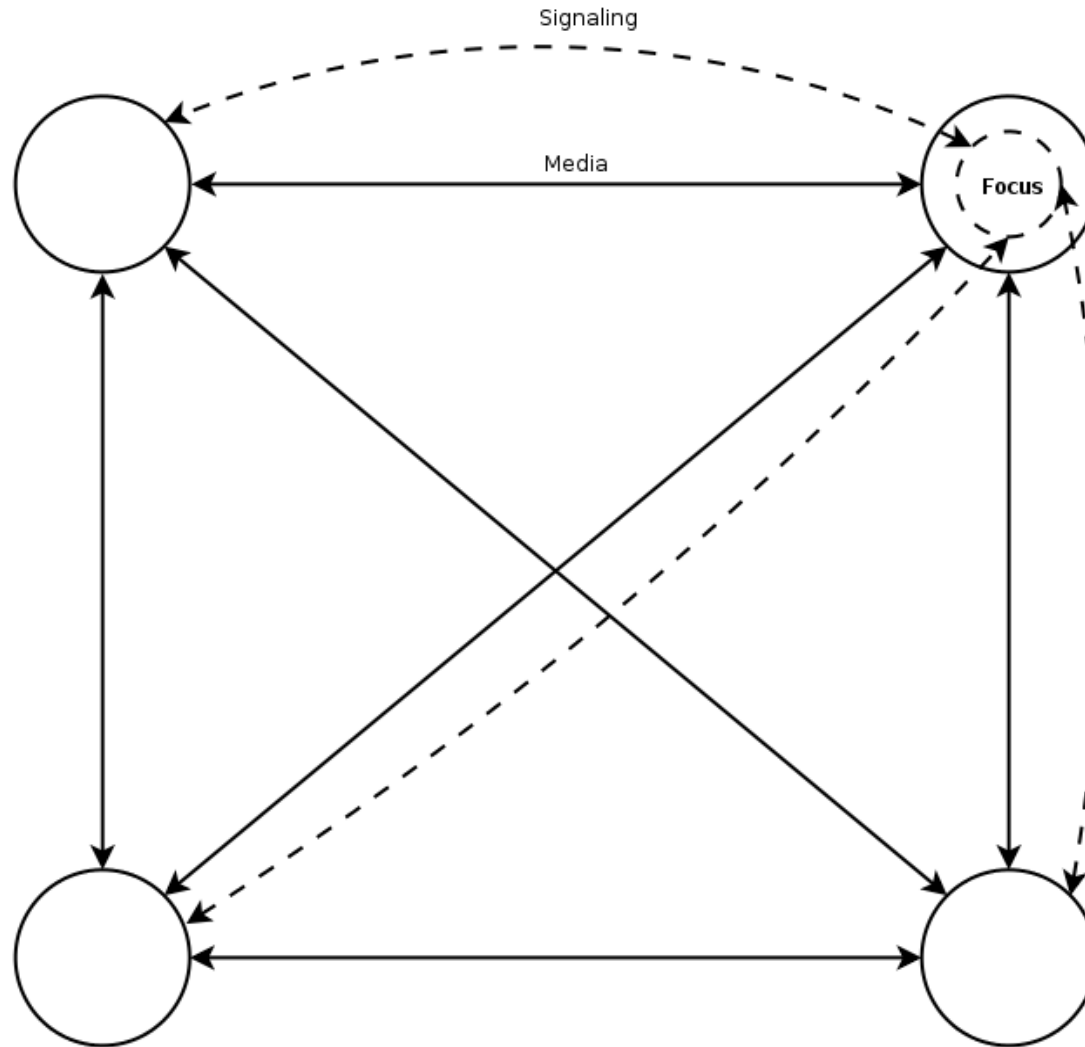
# Focused signalling

## Distributed media (multicast)



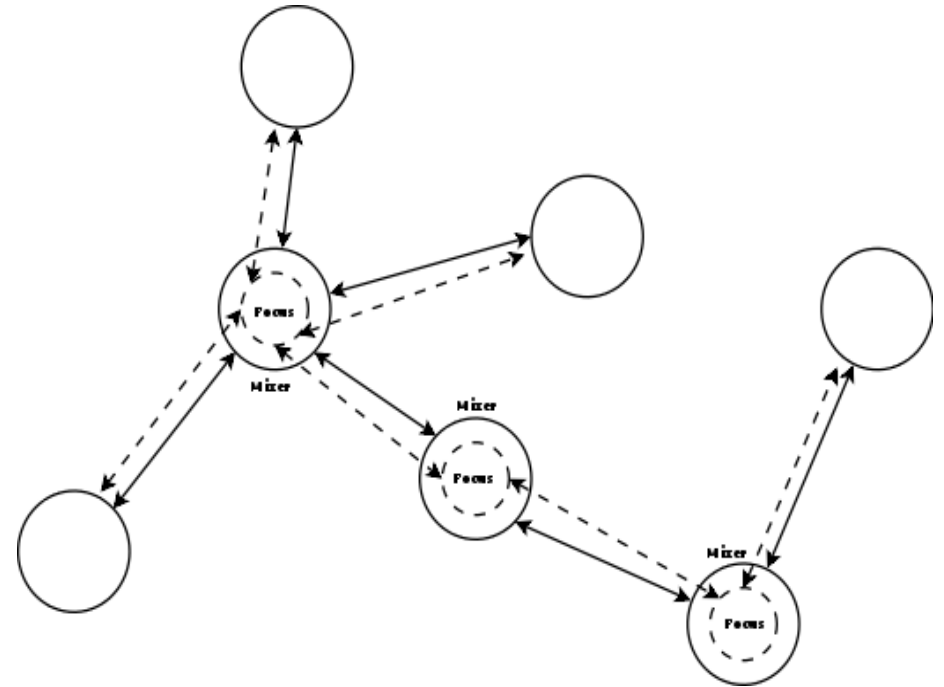
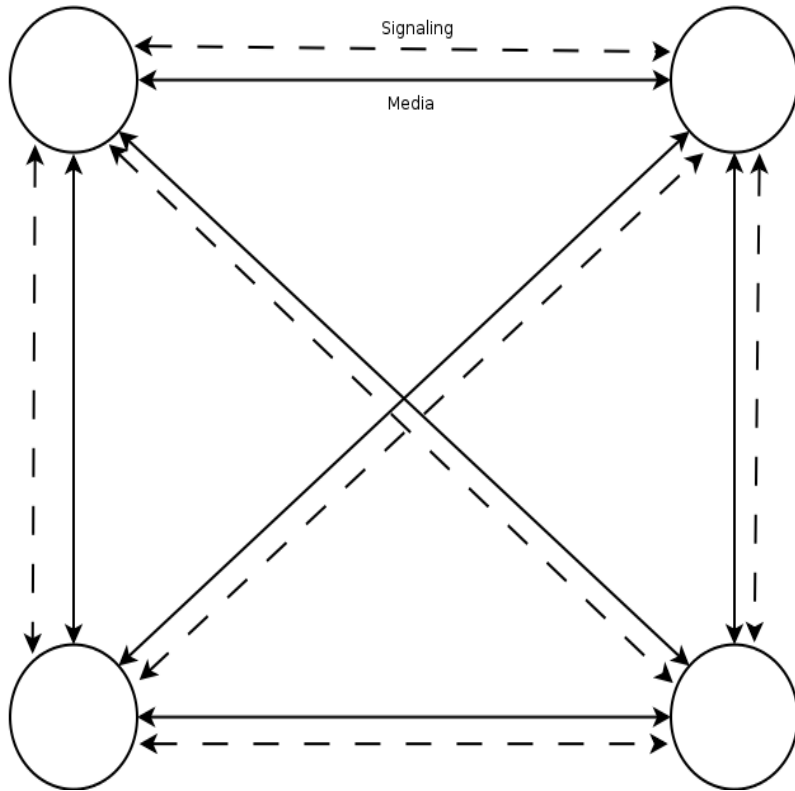
# Focused signalling

## Distributed media (multi-unicast)



# Distributed signalling

## Crazy ideas!



# High level objects

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

# 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

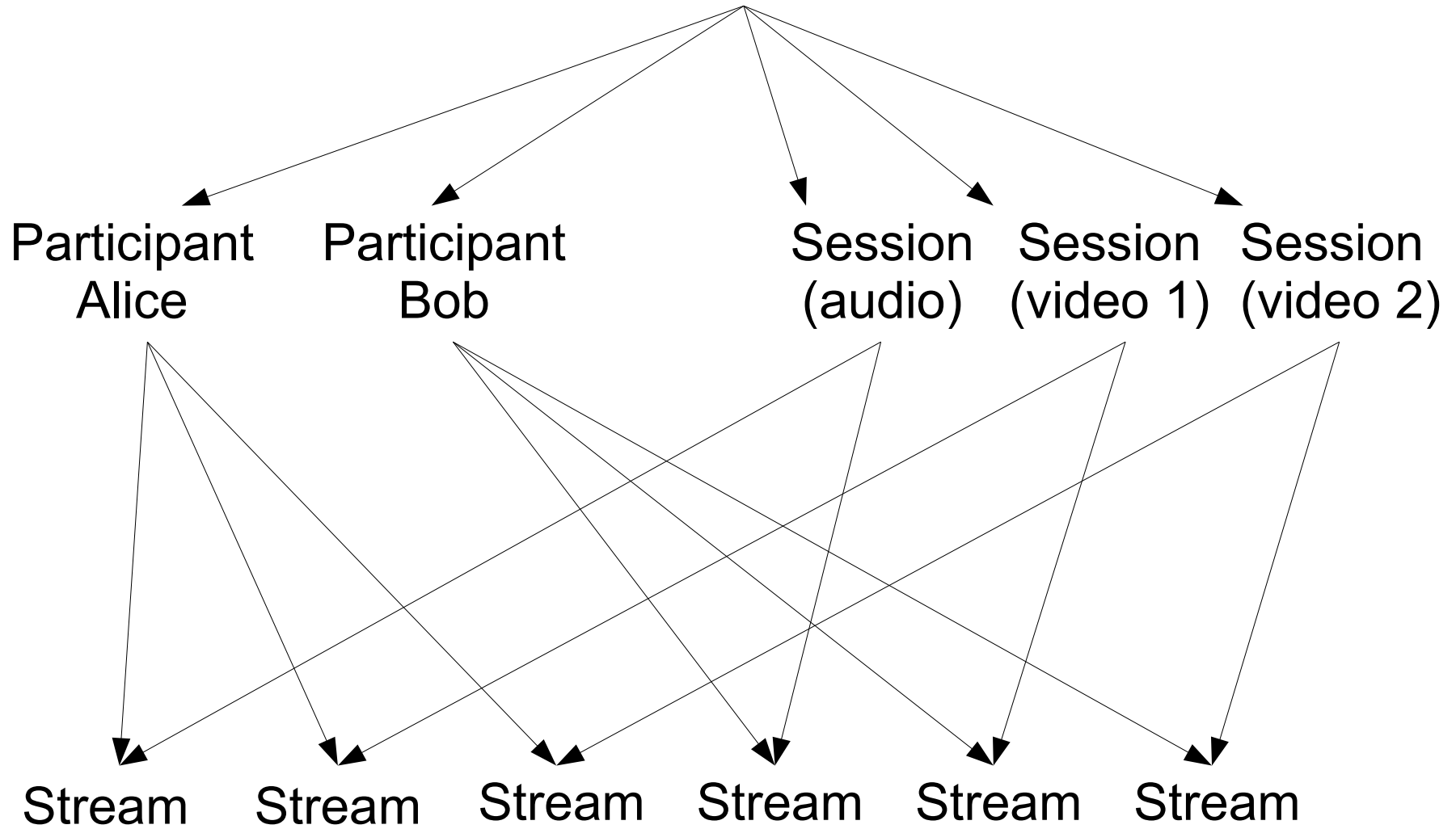
# Stream

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

# Conference

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

# Conference



# New RTP plugin

- Keep good things from older versions
  - Codec detection
  - 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 compatible
  - Windows Live Messenger compatible
  - Including TURN relay support
- Pidgeons, etc

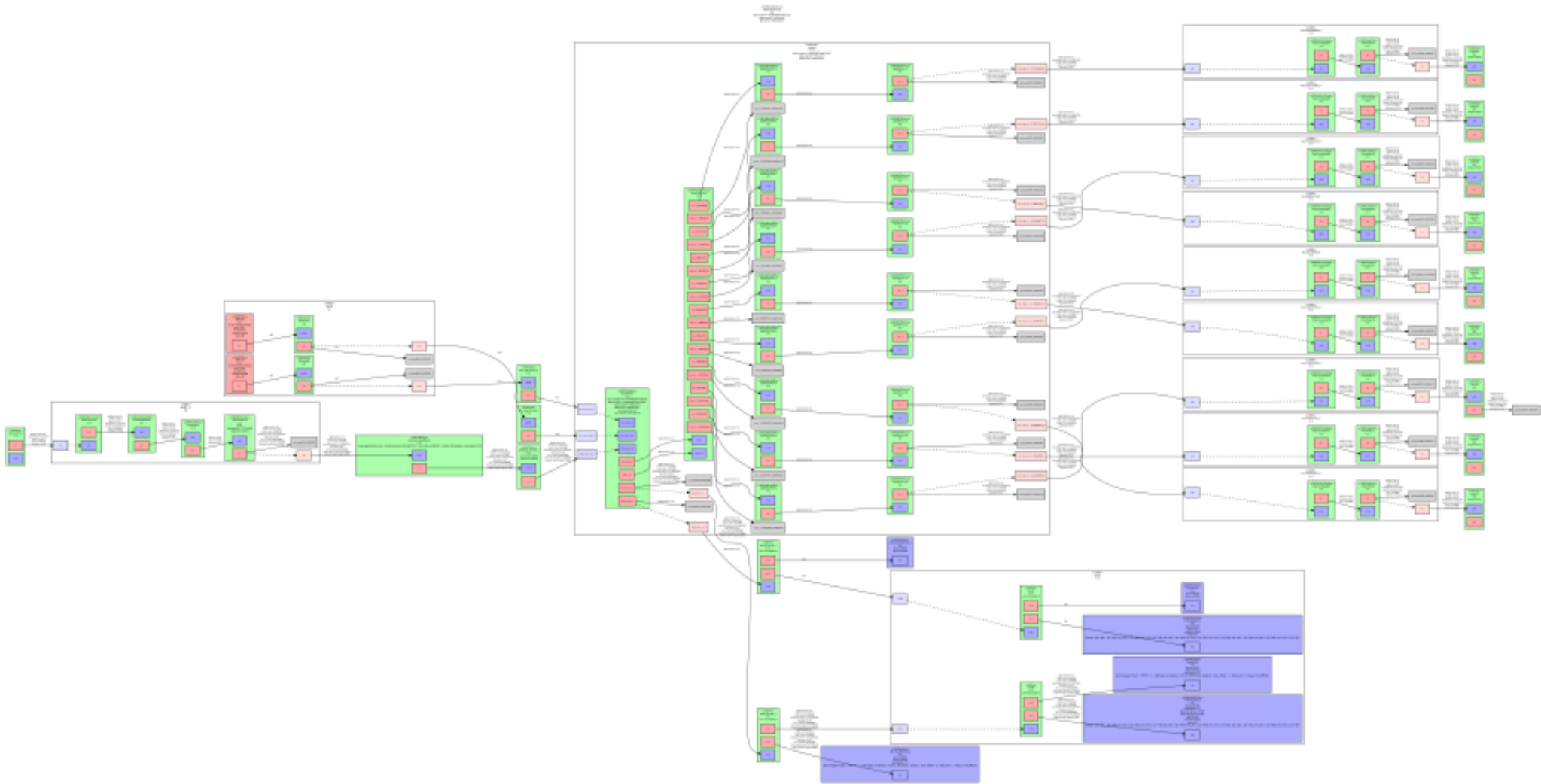
# Current status

- Complete RTP implementation
  - All Farsight 1 features
  - Multi-party
  - Lip-sync
  - Python bindings
  - Automated tests
  - Unicast, Multicast, libnice (ICE) transmitters
  - Complex encoding pipelines (for Comfort Noise)

# Free codecs available

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

# 10 way conference



# Example

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

```
loop = gobject.MainLoop()
```

```
pipeline = gst.Pipeline()
```

```
conference = gst.element_factory_make ("fsrtpconference")
```

```
conference.set_property ("sdes-cname", sys.argv[1] + "@1.2.3.4")
```

```
pipeline.add (conference)
```

```
session = conference.new_session (farsight.MEDIA_TYPE_VIDEO)
```

```
participant = conference.new_participant (sys.argv[2]+"@1.2.3.4")
```

```
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()
```

# Demos

- 3-slide example
- More complex multi-party example
- Will it work ... ?

# Current Users

- Pidgin-vv branch
- aMSN Voice chats
- Telepathy-Farsight library
- Next Generation Maemo Stream Engine

# Extending Farsight

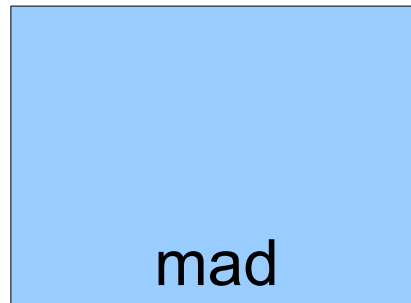
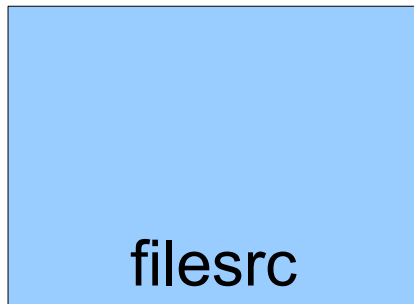
- Codecs are regular GStreamer encoder/decoder elements
- Custom negotiators for codecs that have complex parameter negotiation functions (like H.264)
- Also, RTP payloaders and depayloaders
- Low hanging fruits
  - iLBC
  - G.729
  - H.261

# Introducing GStreamer

- Codecs are regular GStreamer encoder/decoder elements
- Three major types of elements:
  - Sources (camera, microphone, file, etc)
  - Sinks (screen, speakers, file, etc)
  - Transformational elements (encoder, decoders, etc, etc, etc)
  - Also, complex “magical” elements (playbin, farsight, etc)

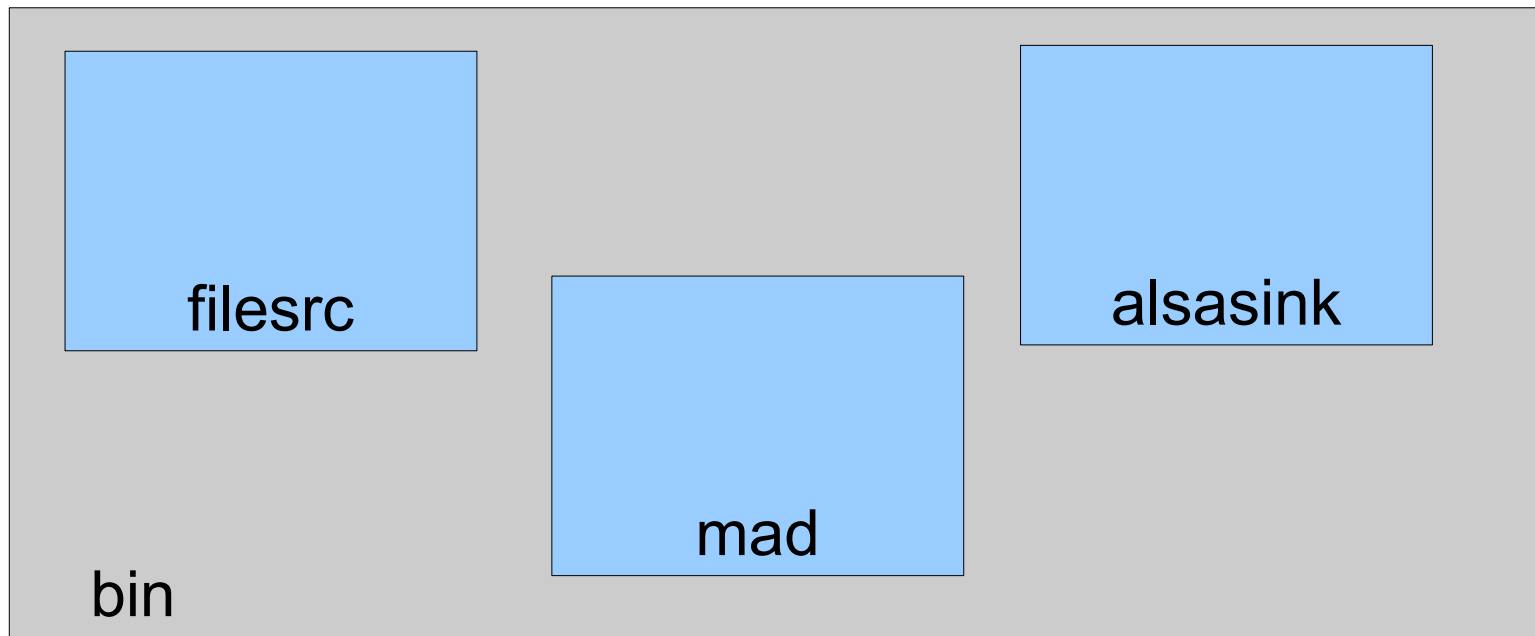
# Basic GStreamer

- Elements



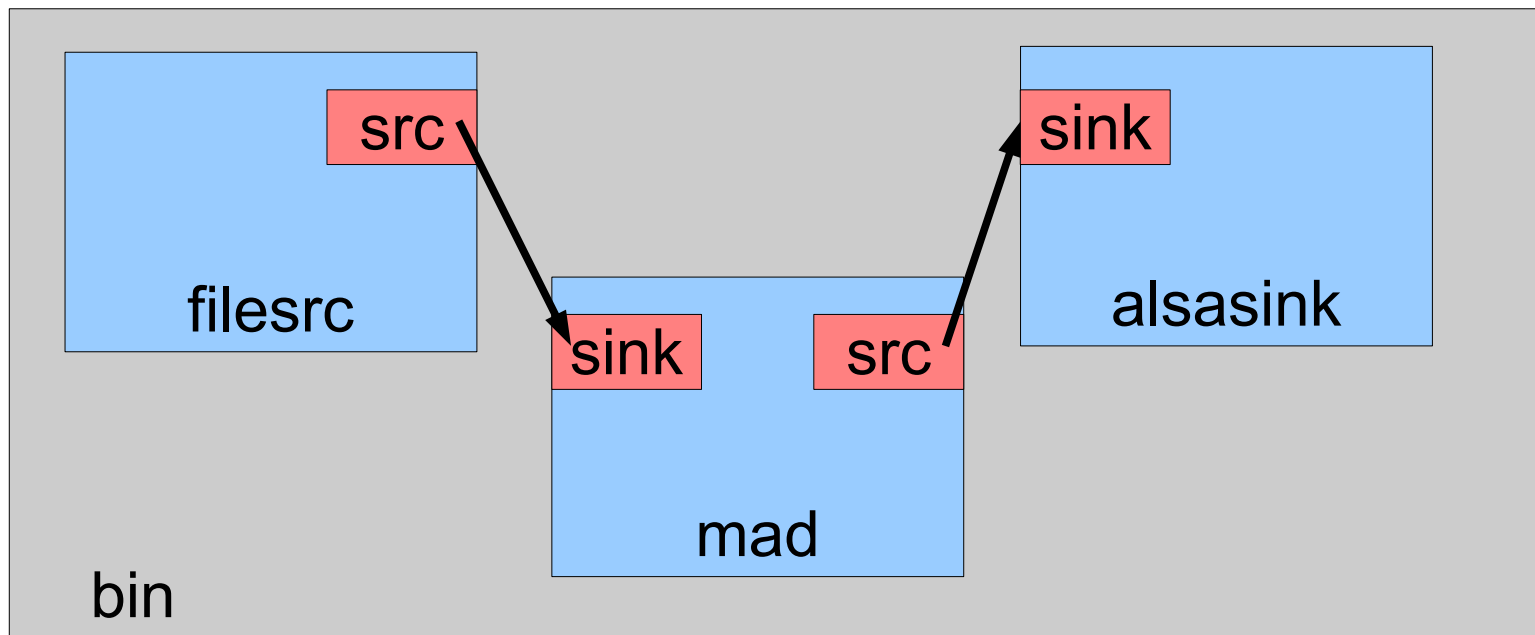
# Basic GStreamer

- Elements
- Bins



# Basic GStreamer

- Elements
- Bins
- Pads



# Extending Farsight

- Non-Free codecs are possible (patents)
- Implement plugins for other protocols
  - MSN Webcam just needs cleanup
  - MSN AV is very similar
  - Others?

Documentation:

<http://www.gstreamer.net/>

# The Future

- Advanced RTP features
  - Secure RTP
  - RTCP AVPF (more feedback)
- Easy library to write simple IM applications
- Use it in all Free clients so they can gain AV capabilities
  - Application integration using Telepathy
  - Empathy
  - aMSN 2
  - PSI

# Mingle

- Multiparty Jingle
- New XMPP extension to set up multi-party conferences using MUCs
- Better than SIP
- Will work with server-based and link-local XMPP
- Will make it possible to use all of Farsight's capabilities
- Will be presented at Linux.Conf.Au 2009 by Sjoerd Simons

# Thank you

- Farsight is brought to you by Collabora
- Questions?

<http://farsight.freedesktop.org/>

<http://www.collabora.co.uk/>

IRC: #farsight @ freenode

