The Tandberg-GStreamer Connection

Or how we stopped worrying, and started using GStreamer

TANDBERG is now part of Cisco.

Us:

- Håvard Graff (hgr)
- Ole André Vadla Ravnås (oleavr)

The beginning

- TANDBERGs first PC-client
  - A bought software-project
  - Mainly for driving sales of infrastructure
  - Based on ActiveX and JavaScript (!)
  - Limited video quality, H.263 only
  - Hopeless to maintain, impossible to advance

The birth of Movi 2.0

- Proprietary code versus Open Source
- A few people knew of GStreamer, and showed it off
- Quickly getting popular with developers, management no-so-much
- Lots of fighting!

We won!

- Decided to use GStreamer
- GObject for application code
  - PIDL
  - (Vala)

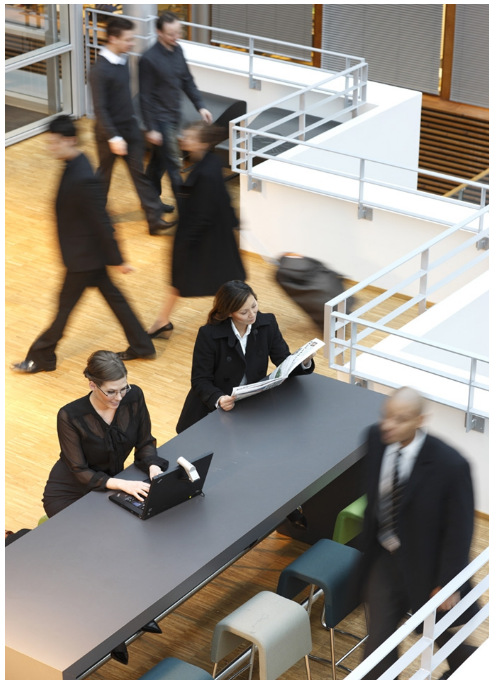
What is Movi?

HINTS OF CINNAMON
TED'S HOT CHOCOLATE
A RICH HOMEMADE
DRINKING CHOCOLATE
COFFEE BAR
HOUSE BLEND $1.5/1.75/2
RED EYE $2/2.5/3
AMERICANO $2.25/2.75
ESPRESSO $2.25/2.75
CAFE LATE $3/3.50/4.00
CAPPUCINO $3/3.50/4.00
MACHIATO $2.25/2.75
CAFE AU LAIT $2.00
ADD VANILLA $.50
ADD ESPRESSO $1.00

Movi

Business quality HD Video on your PC or Mac…

Anywhere.
Anytime.



Superior user experience

- Superior business quality video and audio communications.
- Connect to any standards based video system from anywhere

Scalable & Interoperable

- PC or Mac video capable of scaling to the tens of thousands.
- Seamless functionality with the Total TelePresence Solution

Easy, Secure, Managed

- Easy to use with little or no training or direction.
- Secure and standards based communications
- Centrally managed solution.

Movi Key Features and Benefits

Superior Quality

- Business quality HD video (up to 720p 30 frames-per -second) on the desktop
- PC or Mac Video Industry's best audio performance (MPEG4 AAC-LD, G.722.1/G.711)
- Acoustic echo cancellation
- Rich presence awareness so you know who's available

Scalable & Interoperable

- Capability to deploy to thousands of users
- SIP Registrar (VCS) and Provisioning (TMS)
- Interoperability with any standards-based SIP endpoint and H.323 systems

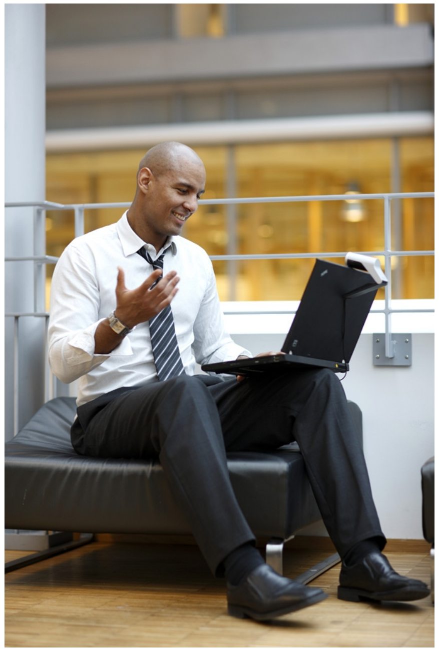
Easy, Secure, Managed

- Enterprise Ready
- Simple installation and management via TMS
- Firewall friendly
- AES and TLS Encryption

Mobile Collaboration with Movi

Extends video communications to the PC or Mac supporting work/life balance programs and green initiatives

Remain connected when traveling with high quality and dependable communications



Build more effective teams across corporate boundaries

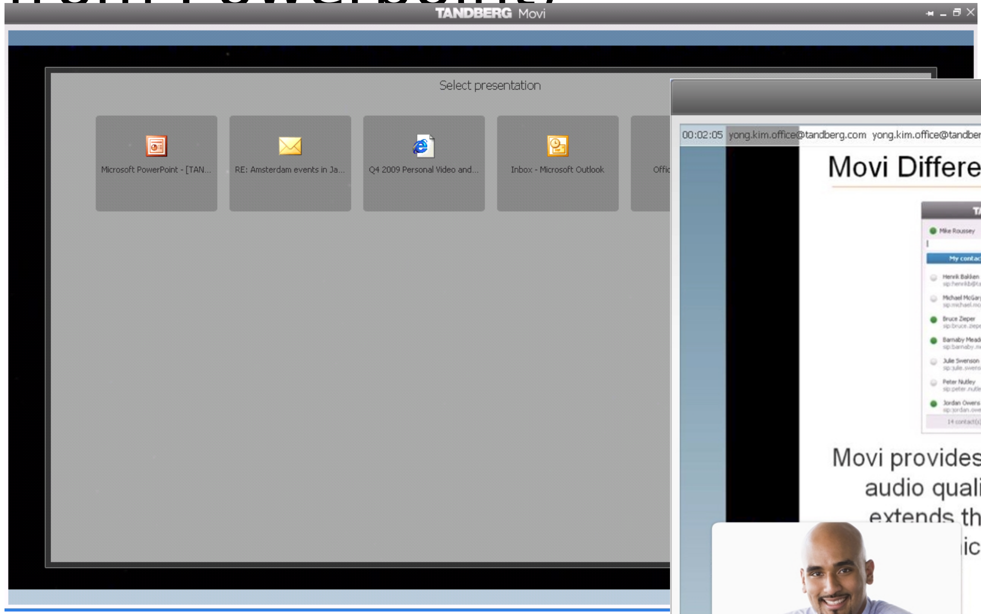
Brings speed and precision to any organization

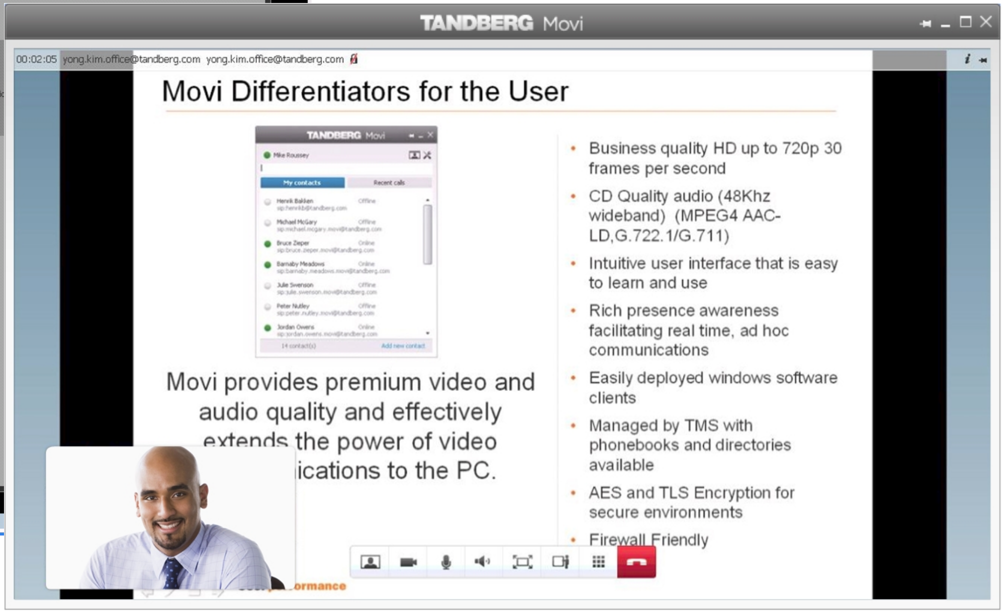
Increase productivity

Collaborate and share

Select application and share content from the PC or Mac with any other standards compliant devices

Optimized for the most common use case (share from Powerpoint)


TANDBERG Movi
Select presentation
Microsoft PowerPoint - [TAN...
RE: Amsterdam events in Ja...
Q4 2009 Personal Video and...
Inbox - Microsoft Outlook
Offic


TANDBERG Movi
00:02:05 yong.kim.office@tandberg.com yong.kim.office@tandberg.com

Movi Differentiators for the User

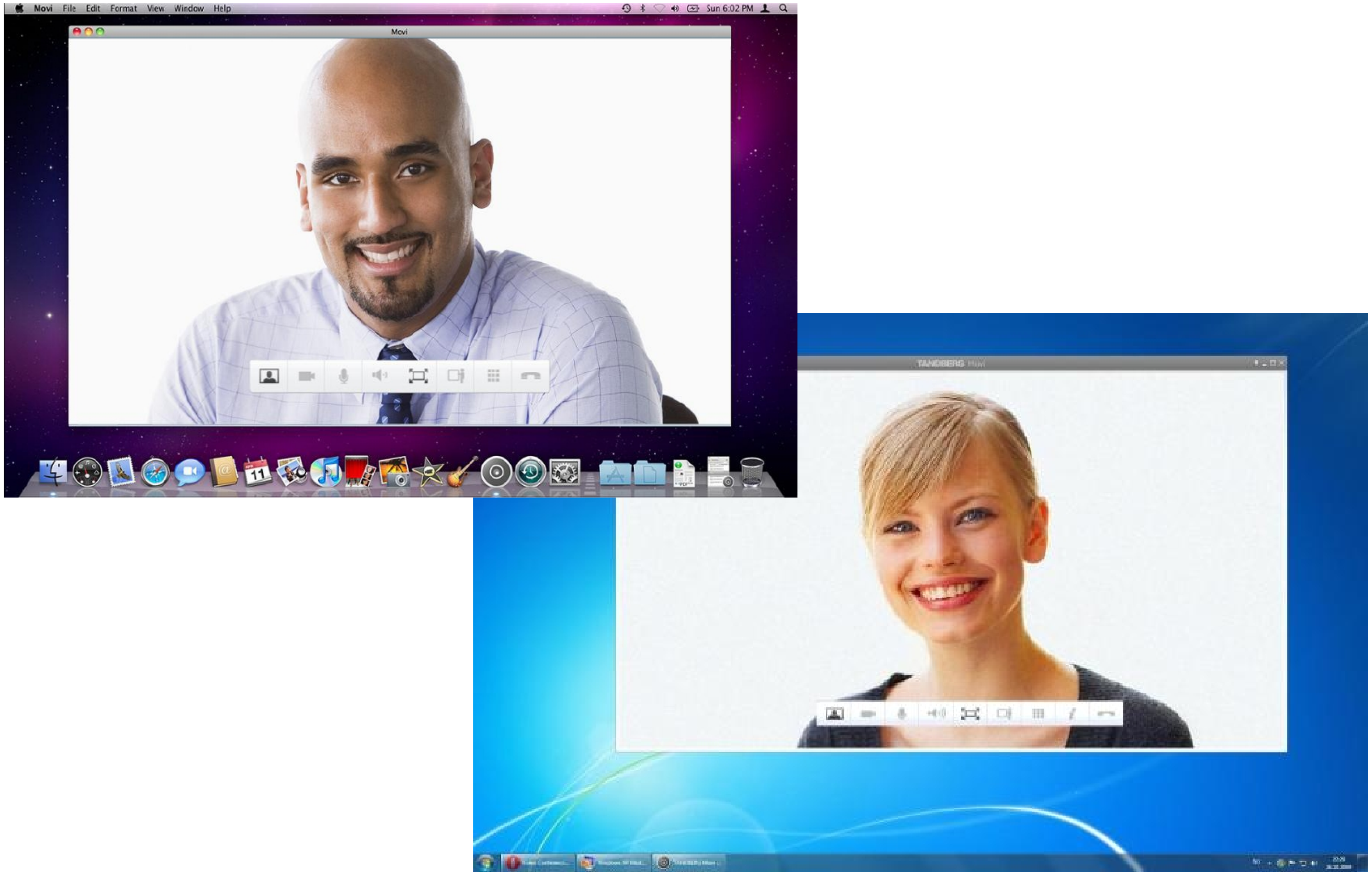
- Business quality HD up to 720p 30 frames per second
- CD Quality audio (48Khz wideband) (MPEG4 AAC-LD,G.722.1/G.711)
- Intuitive user interface that is easy to learn and use
- Rich presence awareness facilitating real time, ad hoc communications
- Easily deployed windows software clients
- Managed by TMS with phonebooks and directories available
- AES and TLS Encryption for secure environments
- Firewall Friendly

Movi provides premium video and audio quality and effectively extends the power of video ications to the PC.

Cisco PrecisionHD USB

- High quality optics
- Glass-only lenses
- Large aperture (F1.7)
- Large sensor → Low noise
- Capable of 720p 30f
- Wideband omni-directional microphone

Movi for the PC or Mac



Details about Movi

- A lot of Intel Optimization
  - Among the fastest MJPEG sw decoders in the world
  - Fractal – in-house developed and optimized H.264 codec
  - Does HD videoconferencing on a standard dual-core PC (competitors require quad core)
  - Can run 720p ~25fps on a single-core Atom!

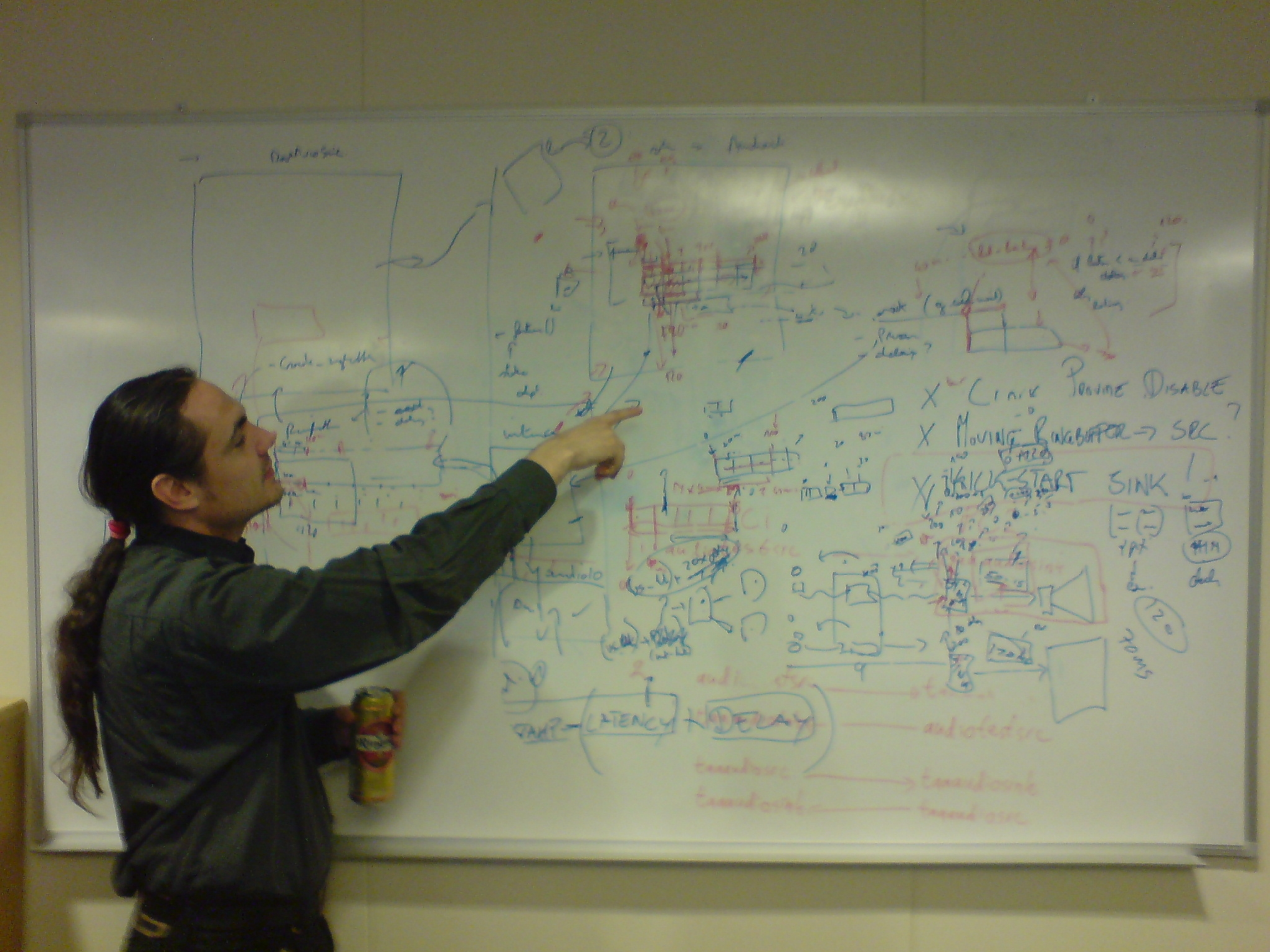
- On the Business Side
  - Movi is used to sell infrastructure (call control, conferencing etc)
  - Shipped about 150 000 licenses
  - Selling more Movi licenses than appliance-based endpoints, effectively doubling demand for network infrastructure products

The future…

- CPVE in CSF
  - Cisco Precision Video Engine – based on the Movi media engine
  - CSF – Client Services Framework – enabling soft clients with common services for network, call control and media
- Potentially GStreamer running on millions of Cisco-clients

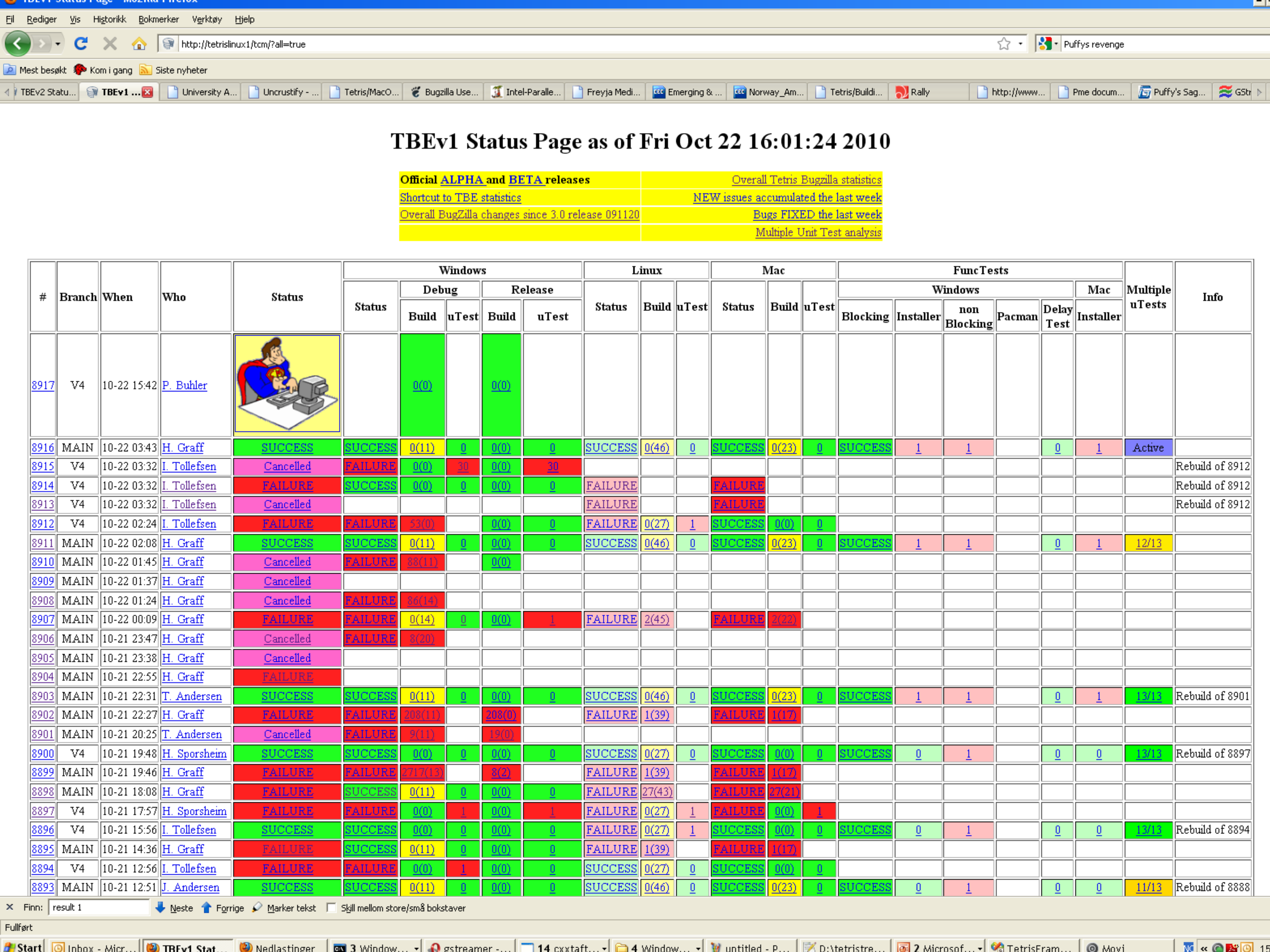
The GStreamer symbiosis

- Quite high threshold to go past simple playback-cases
- Live pipelines poses many challenges
- We have invested a lot of time and effort into GStreamer
  - Directly related to how successful it has been
- Give and take
  - Communication (#IRC)
  - Patches…
    - Memory-leaks
    - Tricky race-conditions
    - New functionality
    - Windows-stuff
  - Sponsored development
    - Dynamic BaseTransform
    - Thread priority in tasks
- Belgian Support ™


X CLINK PROFILE DISABLE
X MOVING RINGBUFFER → SRC ?
X KICK-START SINK !
LATENCY + DELAY
audiotestsrc
audiofedsrc
tcpserversrc
tcpserversink
120
20 MS

TBE

- Our buildsystem
- Windows, Linux and OSX
- Continuous integration
- Runs roughly 12.000 tests related to GStreamer for every build
  - Reveals race-conditions
- Memory-leaks not allowed!


Fil Rediger Vis Historikk Bokmerker Verktøy Hjelp
http://tetrislinux1/tcm/?all=true
Puffys revenge
Mest besøkt Kom i gang Siste nyheter
TBEv2 Statu... TBEv1 ... University A... Uncrustify - ... Tetris/MacO... Bugzilla Use... Intel-Paralle... Freyja Medi... Emerging & ... Norway_Am... Tetris/Buildi... Rally http://www... Pme docum... Puffy's Sag... GStr
Finn: result 1 Neste Forrige Marker tekst Skill mellom store/små bokstaver
Fullført
Start Inbox - Micr... TBEv1 Stat... Nedlastinger 3 Window... gstreamer -... 14 cxxtaft... 4 Window... untitled - P... D:\tetristre... 2 Microsof... TetrisFram... Movi

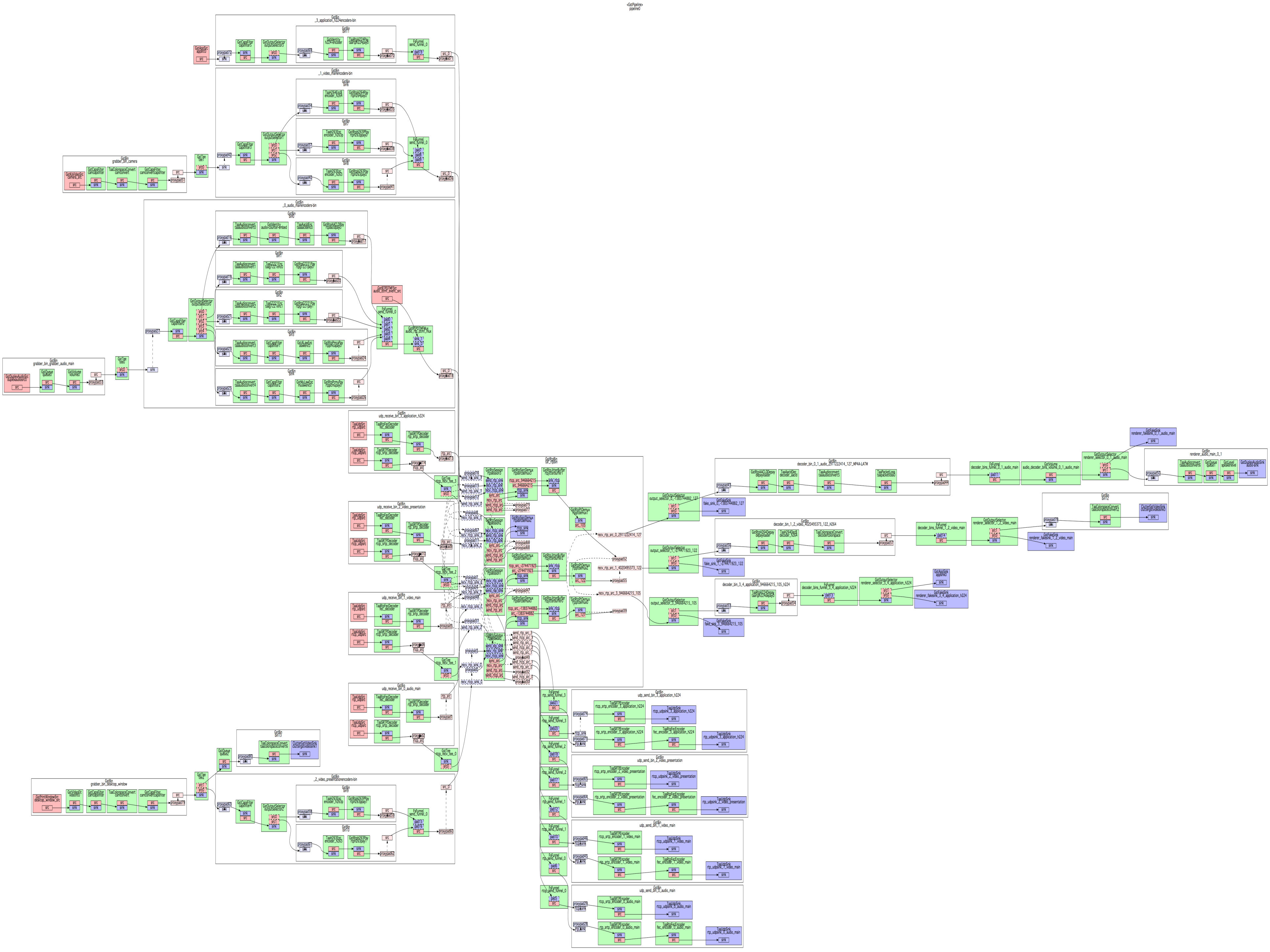
TBEv1 Status Page as of Fri Oct 22 16:01:24 2010

Official ALPHA and BETA releases	Overall Tetris Bugzilla statistics
Shortcut to TBE statistics	NEW issues accumulated the last week
Overall BugZilla changes since 3.0 release 091120	Bugs FIXED the last week
	Multiple Unit Test analysis

#	Branch	When	Who	Status	Windows					Linux			Mac			FuncTests						Multiple uTests	Info
					Status	Debug		Release		Status	Build	uTest	Status	Build	uTest	Windows					Mac		
						Build	uTest	Build	uTest							Blocking	Installer	non Blocking	Pacman	Delay Test	Installer		
8917	V4	10-22 15:42	P. Buhler			0(0)		0(0)															
8916	MAIN	10-22 03:43	H. Graff	SUCCESS	SUCCESS	0(11)	0	0(0)	0	SUCCESS	0(46)	0	SUCCESS	0(23)	0	SUCCESS	1	1		0	1	Active	
8915	V4	10-22 03:32	I. Tollefsen	Cancelled	FAILURE	0(0)	30	0(0)	30														Rebuild of 8912
8914	V4	10-22 03:32	I. Tollefsen	FAILURE	SUCCESS	0(0)	0	0(0)	0	FAILURE			FAILURE										Rebuild of 8912
8913	V4	10-22 03:32	I. Tollefsen	Cancelled						FAILURE			FAILURE										Rebuild of 8912
8912	V4	10-22 02:24	I. Tollefsen	FAILURE	FAILURE	53(0)		0(0)	0	FAILURE	0(27)	1	SUCCESS	0(0)	0								
8911	MAIN	10-22 02:08	H. Graff	SUCCESS	SUCCESS	0(11)	0	0(0)	0	SUCCESS	0(46)	0	SUCCESS	0(23)	0	SUCCESS	1	1		0	1	12/13	
8910	MAIN	10-22 01:45	H. Graff	Cancelled	FAILURE	88(11)		0(0)															
8909	MAIN	10-22 01:37	H. Graff	Cancelled																			
8908	MAIN	10-22 01:24	H. Graff	Cancelled	FAILURE	86(14)																	
8907	MAIN	10-22 00:09	H. Graff	FAILURE	FAILURE	0(14)	0	0(0)	1	FAILURE	2(45)		FAILURE	2(22)									
8906	MAIN	10-21 23:47	H. Graff	Cancelled	FAILURE	8(20)																	
8905	MAIN	10-21 23:38	H. Graff	Cancelled																			
8904	MAIN	10-21 22:55	H. Graff	FAILURE																			
8903	MAIN	10-21 22:31	T. Andersen	SUCCESS	SUCCESS	0(11)	0	0(0)	0	SUCCESS	0(46)	0	SUCCESS	0(23)	0	SUCCESS	1	1		0	1	13/13	Rebuild of 8901
8902	MAIN	10-21 22:27	H. Graff	FAILURE	FAILURE	208(11)		208(0)		FAILURE	1(39)		FAILURE	1(17)									
8901	MAIN	10-21 20:25	T. Andersen	Cancelled	FAILURE	9(11)		19(0)															
8900	V4	10-21 19:48	H. Sporsheim	SUCCESS	SUCCESS	0(0)	0	0(0)	0	SUCCESS	0(27)	0	SUCCESS	0(0)	0	SUCCESS	0	1		0	0	13/13	Rebuild of 8897
8899	MAIN	10-21 19:46	H. Graff	FAILURE	FAILURE	2717(13)		8(2)		FAILURE	1(39)		FAILURE	1(17)									
8898	MAIN	10-21 18:08	H. Graff	FAILURE	SUCCESS	0(11)	0	0(0)	0	FAILURE	27(43)		FAILURE	27(21)									
8897	V4	10-21 17:57	H. Sporsheim	FAILURE	FAILURE	0(0)	1	0(0)	1	FAILURE	0(27)	1	FAILURE	0(0)	1								
8896	V4	10-21 15:56	I. Tollefsen	SUCCESS	SUCCESS	0(0)	0	0(0)	0	FAILURE	0(27)	1	SUCCESS	0(0)	0	SUCCESS	0	1		0	0	13/13	Rebuild of 8894
8895	MAIN	10-21 14:36	H. Graff	FAILURE	SUCCESS	0(11)	0	0(0)	0	FAILURE	1(39)		FAILURE	1(17)									
8894	V4	10-21 12:56	I. Tollefsen	FAILURE	FAILURE	0(0)	1	0(0)	0	SUCCESS	0(27)	0	SUCCESS	0(0)	0								
8893	MAIN	10-21 12:51	J. Andersen	SUCCESS	SUCCESS	0(11)	0	0(0)	0	SUCCESS	0(46)	0	SUCCESS	0(23)	0	SUCCESS	0	1		0	0	11/13	Rebuild of 8888

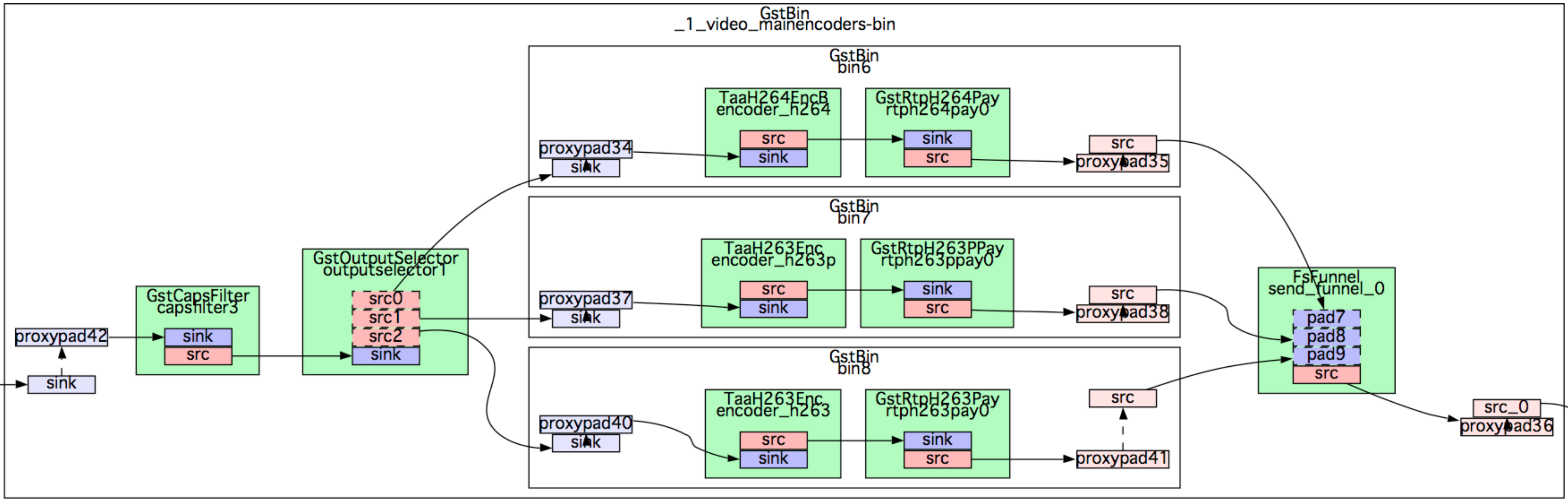
Our pipeline
– a mix of Open Source and proprietary

- Proprietary:
  - H.264, H.263, H.263+
  - AAC-LD
  - G.722.1
  - JPEG decoder
  - SRTP (Secure RTP)
  - Colorspace converter
  - Forward Error Correction
  - Far End Camera Control (H.224)
  - Audio packetloss concealer
  - Audio sink and src
- Open source
  - RTP (payloading, rtpbin)
  - Pipe-fittings: Tee / OutputSelector / Funnel (farsight)
  - Capsfilter
  - Level, Volume
  - Queue
  - Fakesink, Appsink, Cluttersink
  - Videosources for Windows and OS X (made by us!)



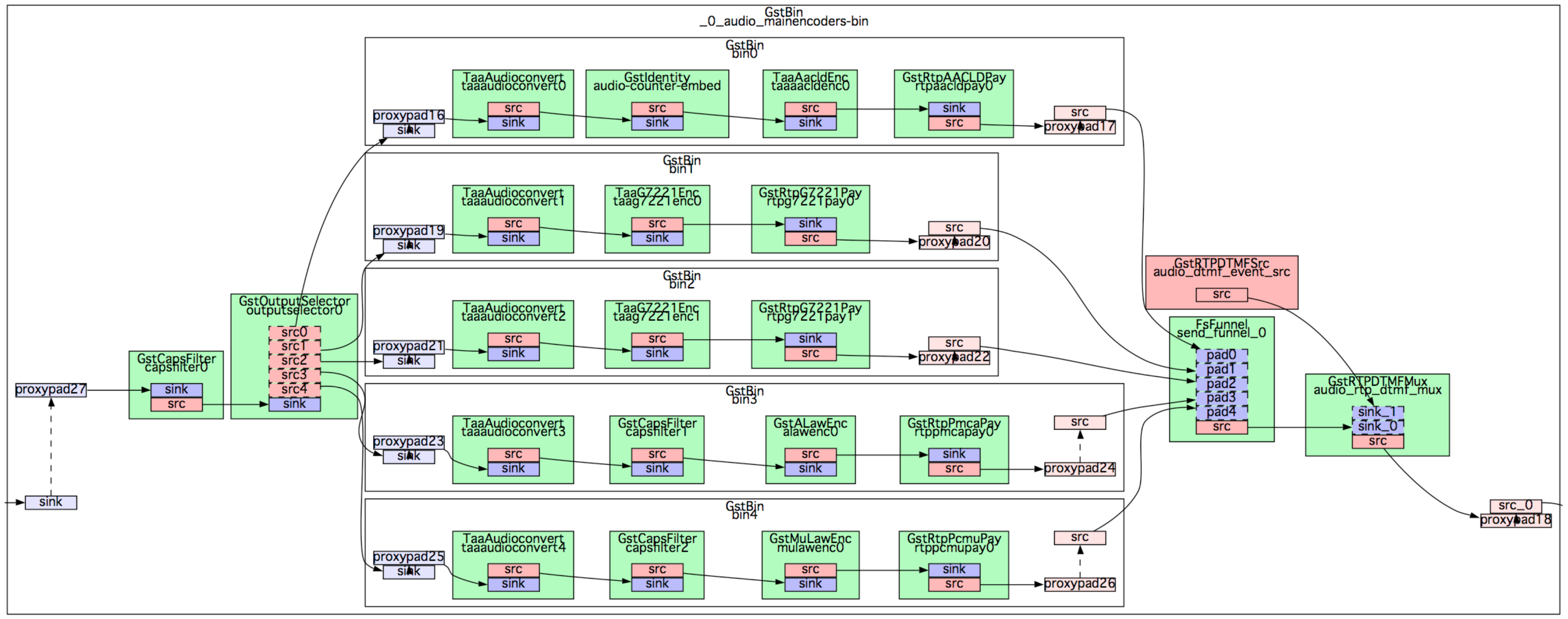
<GstPipeline>
pipeline0
GstBin _3_application_h224encoders-bin
GstBin _1_video_mainencoders-bin
GstBin grabber_bin_camera
GstBin _0_audio_mainencoders-bin
GstBin grabber_bin_grabber_audio_main
GstBin udp_receive_bin_3_application_h224
GstBin udp_receive_bin_2_video_presentation
GstBin udp_receive_bin_1_video_main
GstBin udp_receive_bin_0_audio_main
GstRtpBin rtpbin
GstBin decoder_bin_0_1_audio_2911222414_127_MP4A-LATM
GstBin decoder_bin_1_2_video_4020495373_122_H264
GstBin decoder_bin_3_4_application_946664215_105_h224
GstBin renderer_audio_main_0_1
GstBin grabber_bin_desktop_window
GstBin _2_video_presentationencoders-bin
GstBin udp_send_bin_3_application_h224
GstBin udp_send_bin_2_video_presentation
GstBin udp_send_bin_1_video_main
GstBin udp_send_bin_0_audio_main

Encoder Bin (Video)



- Capsfilter forces a standard videoformat
  - YUV I420
    - Used by all our video encoders
- OutputSelector used for selecting encoder
- Funnel in the opposite end

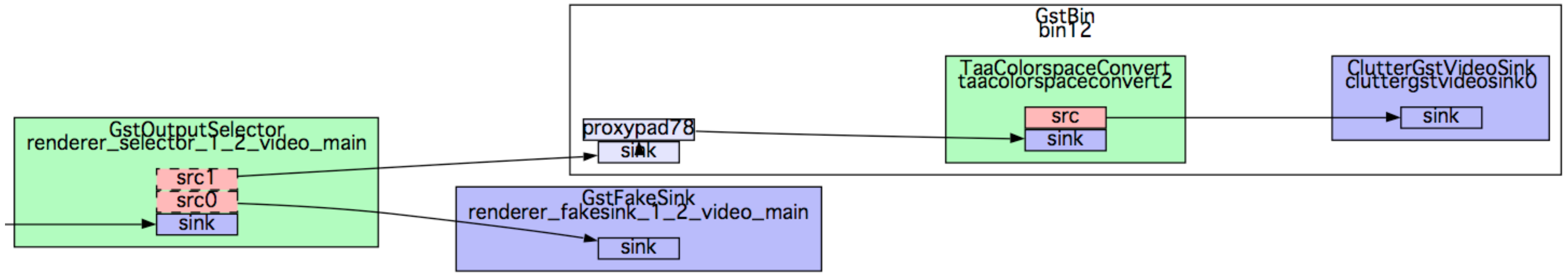
Encoder Bin (Audio)



Dynamic Linking Sinks

- Several Problems
  - The ClutterSink needs to be created inside the GLContext
    - Dangerous to do this synchronously
      - Deadlocks
    - Asynchronously
      - Problem with flow-not-linked
  - The sink needs to be removed dynamically as well
    - Avoiding flow-not-linked
    - Blocking the pad is problematic:
      - Unsafe to unlink until the callback comes
      - No buffer – no callback

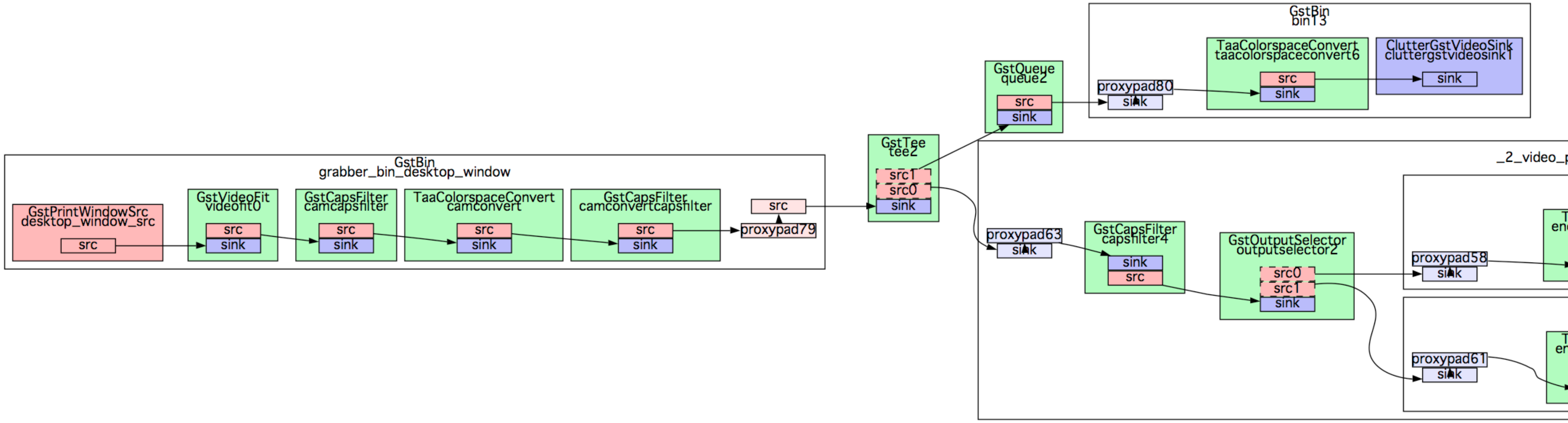
Our solution



- Linking
  - The OutputSelector start being linked to the fakesink
  - Once the sink arrives, it is linked up, and swapped in
- Unlinking
  - A probe is attached to the src-pad of the selector
  - A custom event is sent to the sink-pad of the selector
  - In the callback of that probe, the sink is unlinked and the probe removed

Sources in live pipelines

- Sources send events:
  - When starting
    - Newsegment
  - When stopping
    - End Of Stream
- We don´t want either!
  - Newsegment
    - A source being started in a running pipeline will emit a newsegment with start 0. (Baseclass functionality, not configurable (yet…))
    - However, the buffer timestamps needs to be starting from running_time, not 0, due to lipsync with other media. (the horrors of ts_offset…)
    - The newsegment will be appended in the sink, effectively delaying all buffers with running_time.
    - Our problem: After starting self-view, application would lock up for the length of the conversation up to that point!
  - End Of Stream
    - Would effectively stop all elements upstream. (rtp send…)
    - Muting video, we unlink the src (free up the resources)
    - Unmuting would not work…



- Our solution
  - Attaching a probe, removing all Newsegments and EOS!
  - We have yet to miss them…

```
static void
attach_event_dropper_on_tee (GstElement * camera_tee)
{
  GstPad * pad_for_event_dropper = gst_element_get_pad (camera_tee, "sink");
  gst_pad_add_event_probe (pad_for_event_dropper, (GCallback)event_dropper, NULL);
  gst_object_unref (pad_for_event_dropper);
}

static gboolean
event_dropper (GstPad * pad, GstMiniObject * mini_obj, gpointer user_data)
{
  GstEvent * event = GST_EVENT (mini_obj);
  int event_type = GST_EVENT_TYPE (event);

  switch (event_type)
  {
    case GST_EVENT_EOS:
    case GST_EVENT_NEWSEGMENT:
      // drop
      return FALSE;
  }
  return TRUE;
}
```

Renderdelay

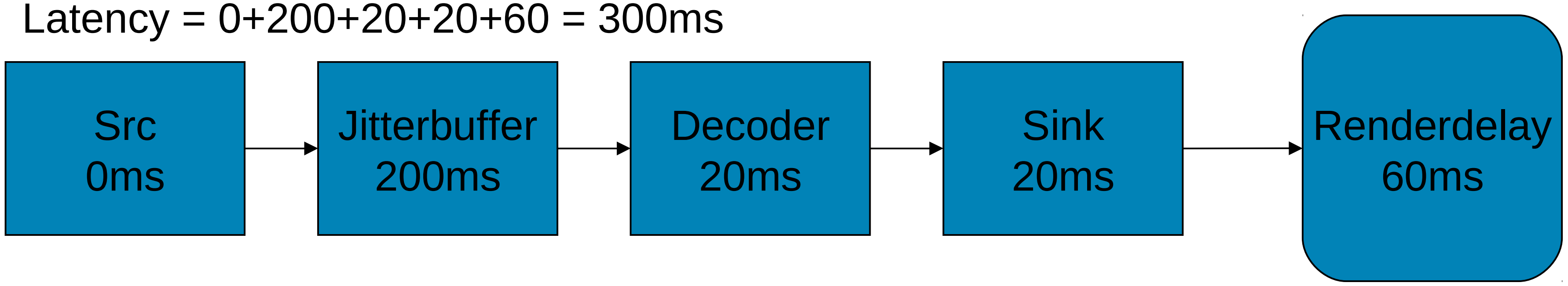
- To be able to compensate for known delays after the buffer has left the sink.
- In our case, ex. DirectSound buffer headroom and processing delay.
- Description:
    - Set on the sink. (basesink)
    - Adds the configured renderdelay as extra reported latency.
    - Then subtracts its own renderdelay after latency has been added.
- ?!?!?
    - What is the point then?
        - No point if there is only one stream.
        - For 2 or more streams, you could have something like this:

$$\begin{tikzcd}
\boxed{\text{Src } 0\text{ms}} \arrow[r] & \boxed{\text{Jitterbuffer } 200\text{ms}} \arrow[r] & \boxed{\text{Decoder } 20\text{ms}} \arrow[r] & \boxed{\text{Sink } 20\text{ms}} \arrow[r] & \boxed{\text{Renderdelay } 60\text{ms}}
\end{tikzcd}$$

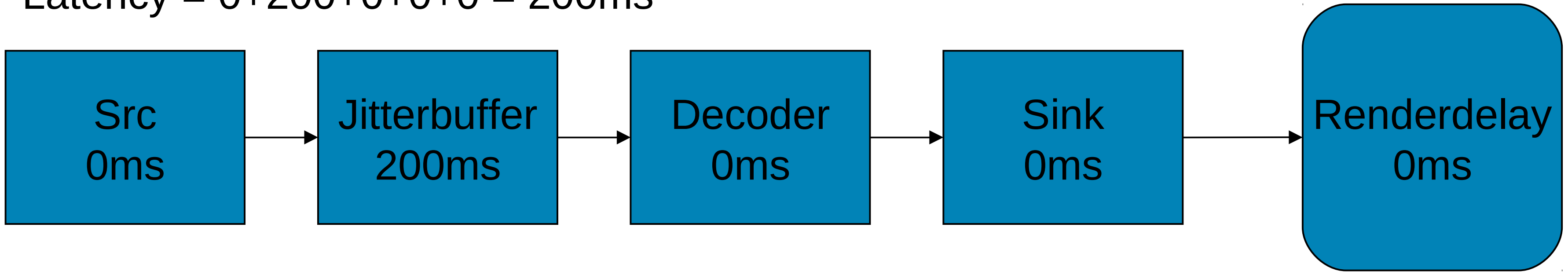
$$\begin{tikzcd}
\boxed{\text{Src } 0\text{ms}} \arrow[r] & \boxed{\text{Jitterbuffer } 200\text{ms}} \arrow[r] & \boxed{\text{Decoder } 0\text{ms}} \arrow[r] & \boxed{\text{Sink } 0\text{ms}} \arrow[r] & \boxed{\text{Renderdelay } 0\text{ms}}
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{\text{Src } 0\text{ms}} \arrow[r] & \boxed{\text{Jitterbuffer } 200\text{ms}} \arrow[r] & \boxed{\text{Decoder } 30\text{ms}} \arrow[r] & \boxed{\text{Sink } 0\text{ms}} \arrow[r] & \boxed{\text{Renderdelay } 20\text{ms}}
\end{tikzcd}$$

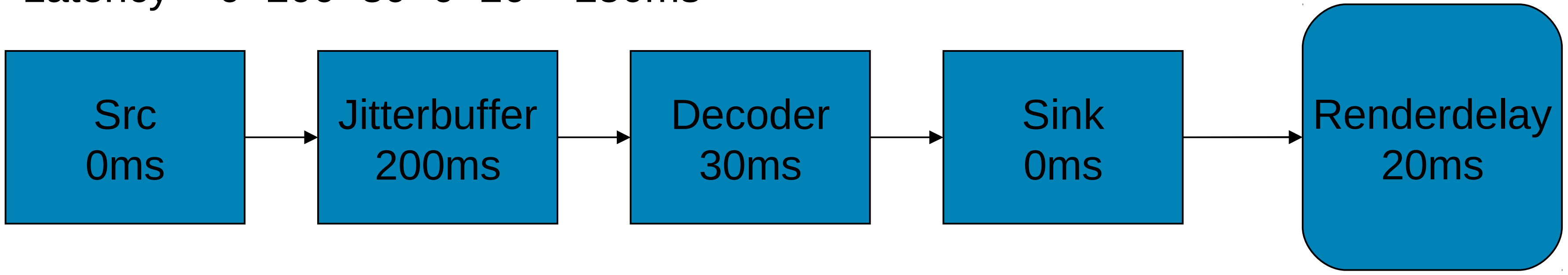
Latency = 0+200+20+20+60 = 300ms



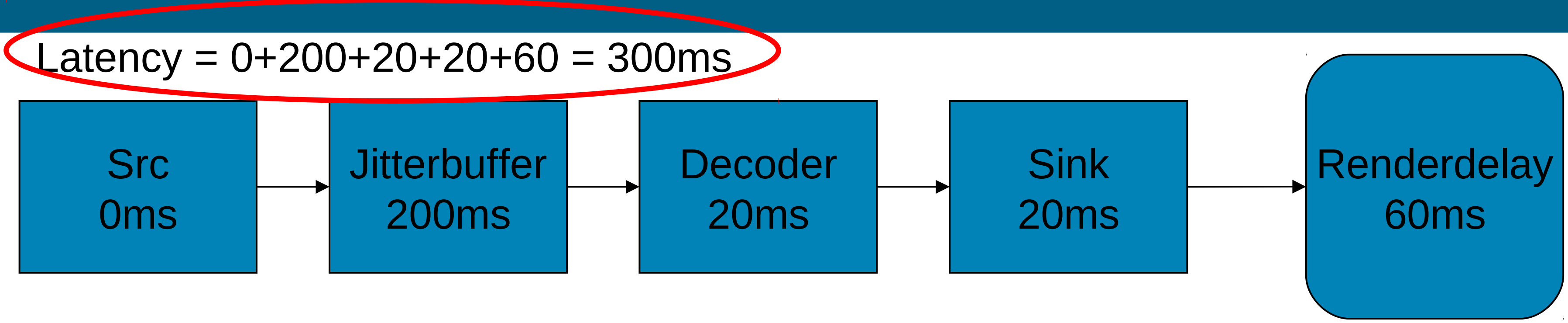
Latency = 0+200+0+0+0 = 200ms



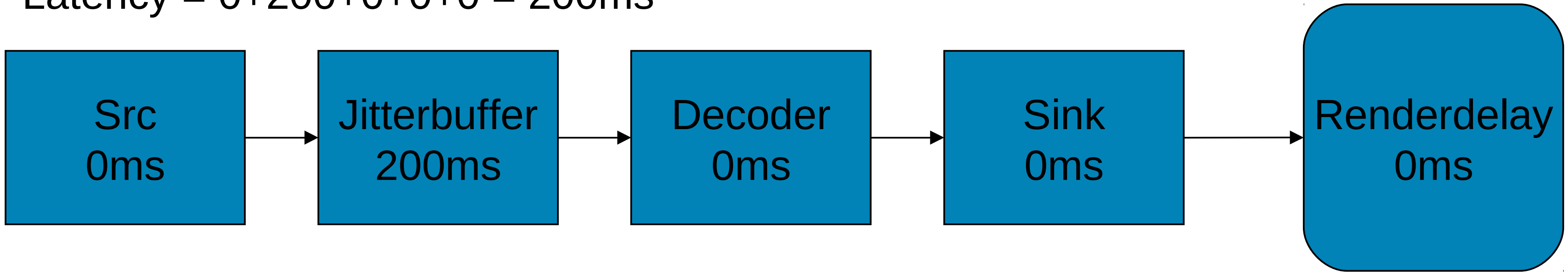
Latency = 0+200+30+0+20 = 250ms



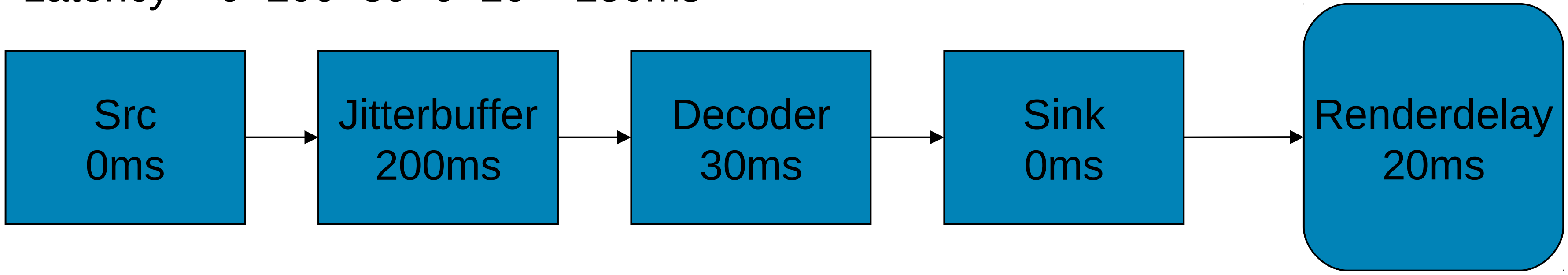
Latency = 0+200+20+20+60 = 300ms

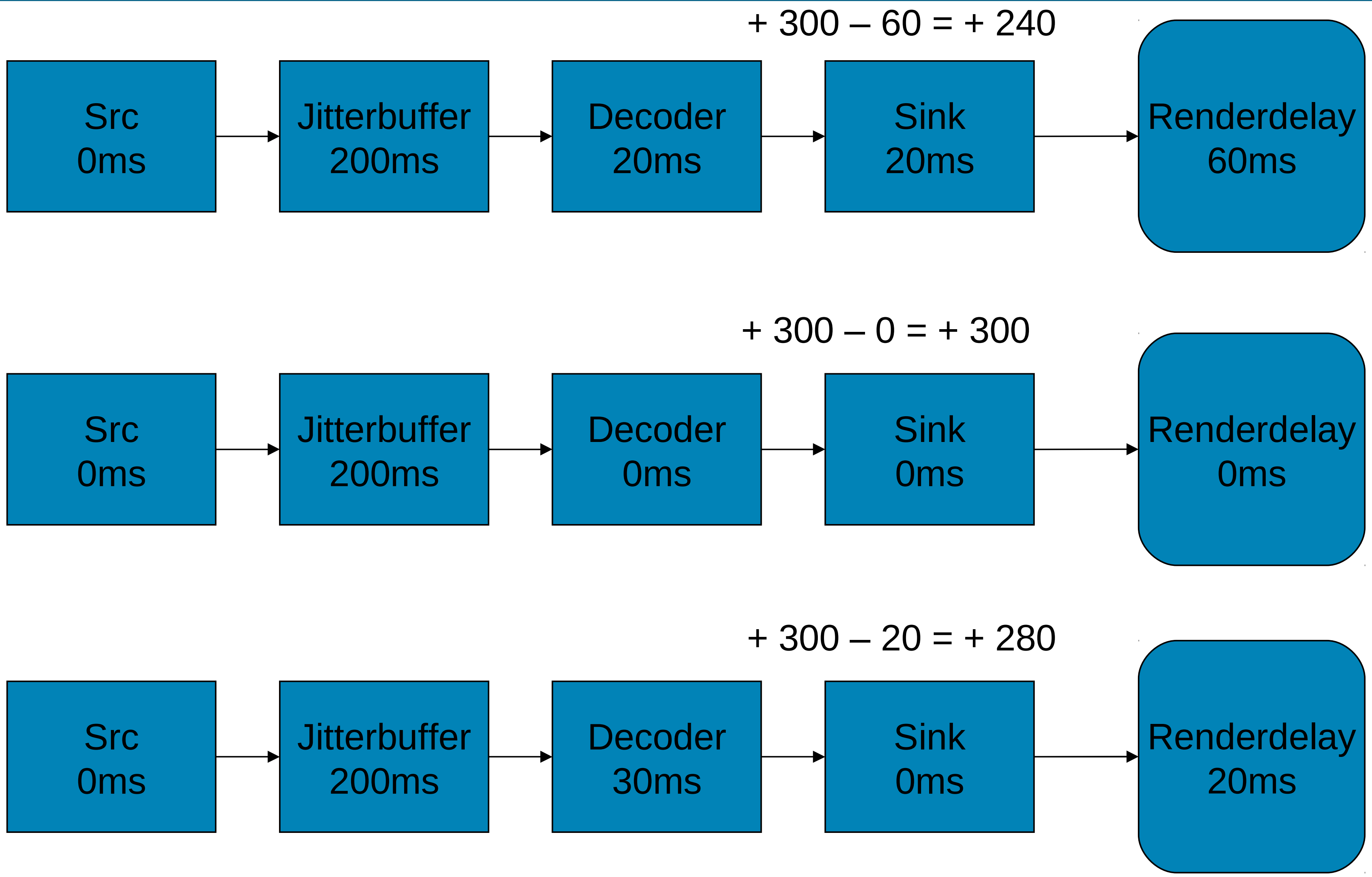


Latency = 0+200+0+0+0 = 200ms



Latency = 0+200+30+0+20 = 250ms

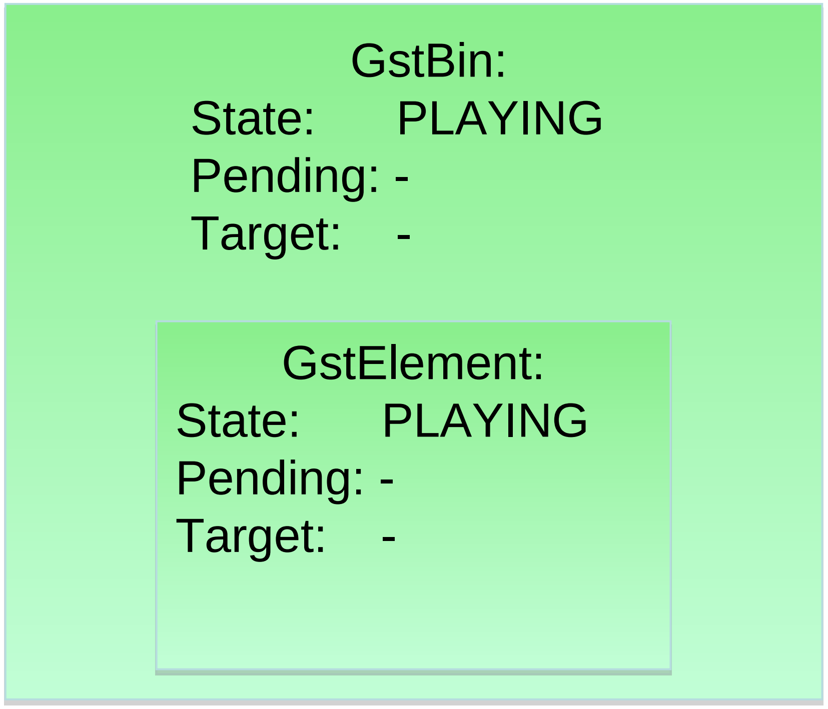



+ 300 – 60 = + 240
Src 0ms
Jitterbuffer 200ms
Decoder 20ms
Sink 20ms
Renderdelay 60ms
+ 300 – 0 = + 300
Src 0ms
Jitterbuffer 200ms
Decoder 0ms
Sink 0ms
Renderdelay 0ms
+ 300 – 20 = + 280
Src 0ms
Jitterbuffer 200ms
Decoder 30ms
Sink 0ms
Renderdelay 20ms

Locked State

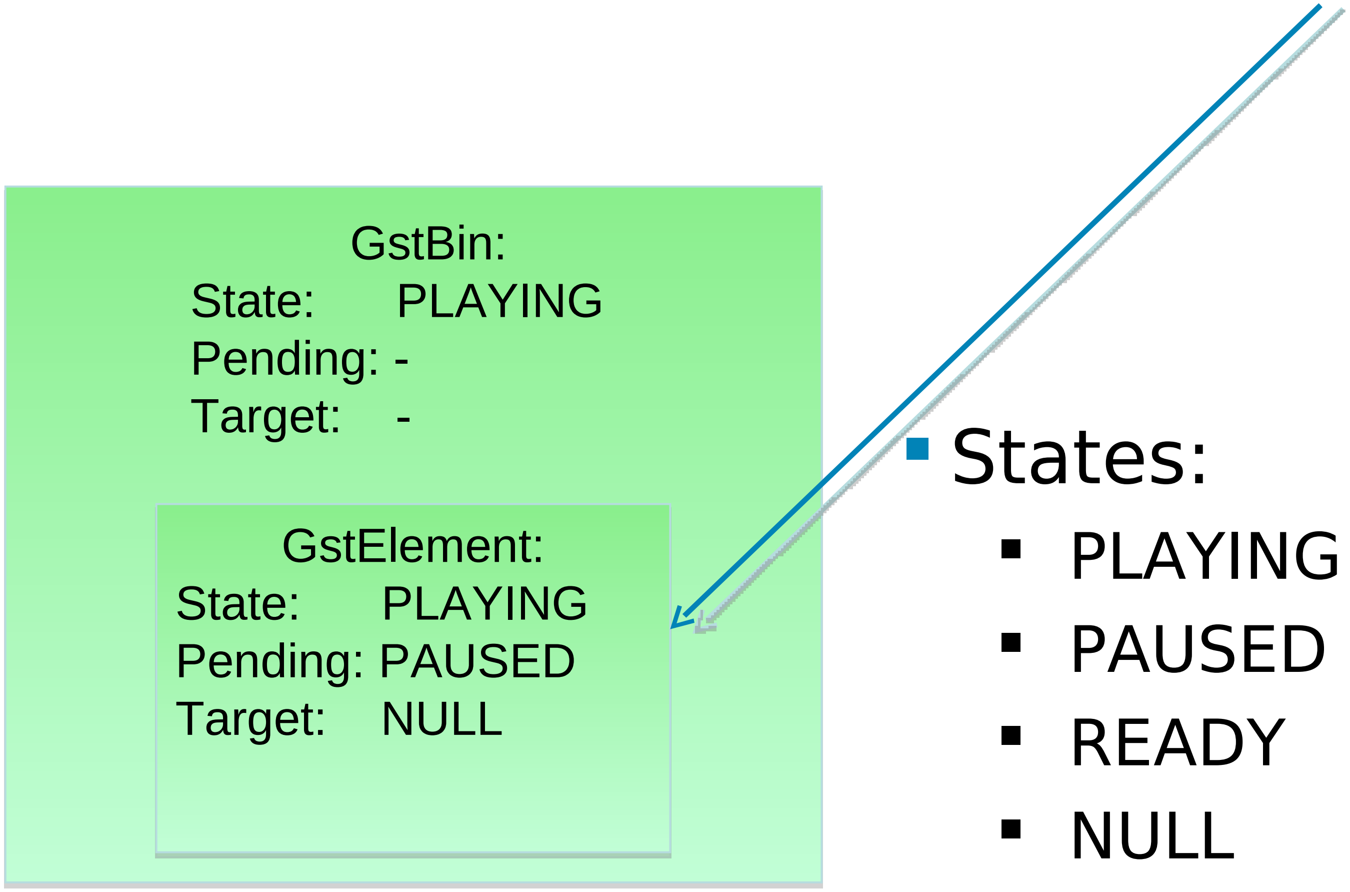
- Deciding when elements change state
- Crucial to dynamic pipelinebuilding due to a shortcoming of state-changes (is it fixed?)
- We use locked-state for all elements that are dynamically added or removed from the main pipeline.

Both elements in PLAYING



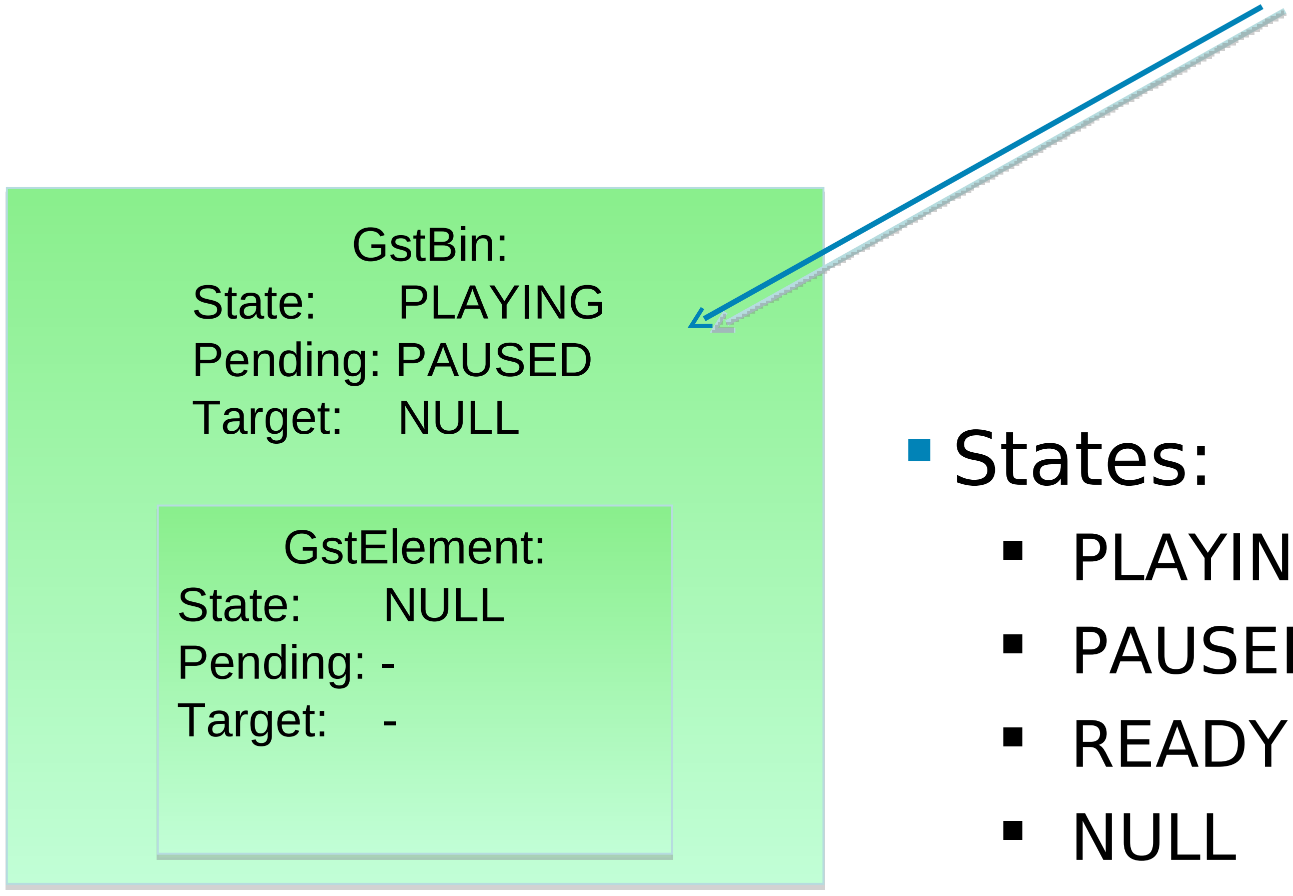
- States:
  - PLAYING
  - PAUSED
  - READY
  - NULL

Setting the child to NULL



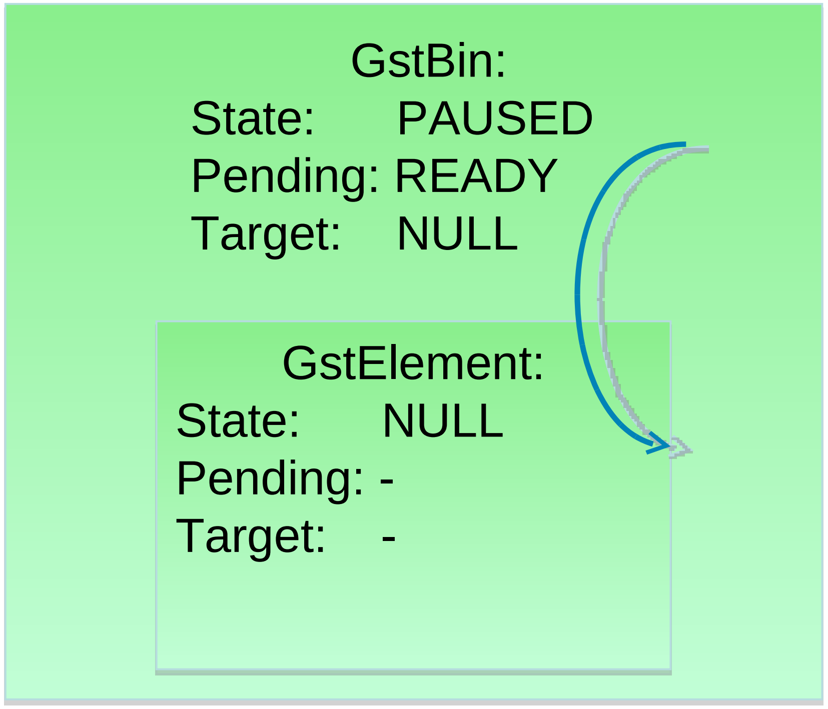
- States:
  - PLAYING
  - PAUSED
  - READY
  - NULL

Parent is set to NULL



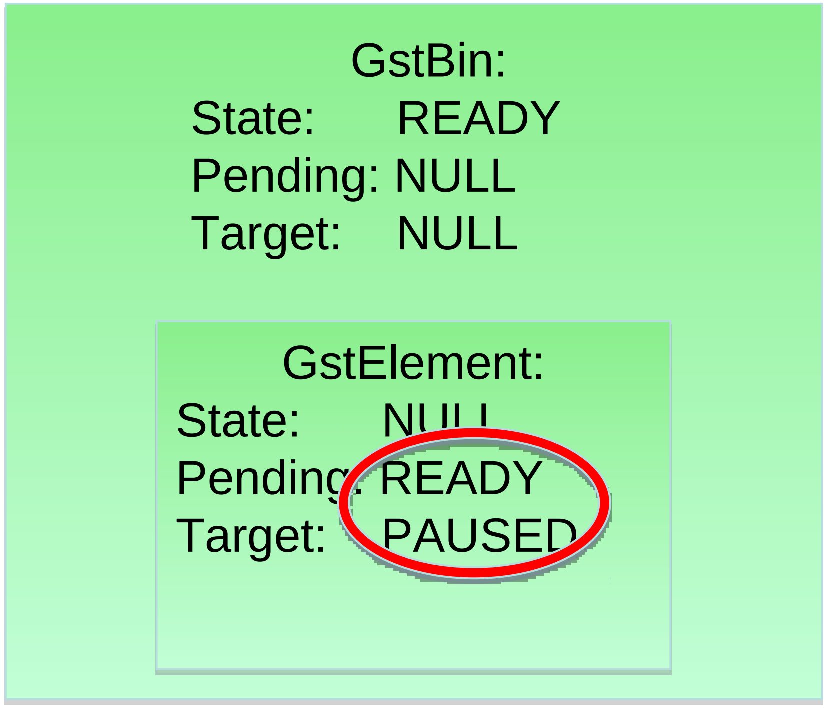
- States:
  - PLAYING
  - PAUSED
  - READY
  - NULL

Going through the states, next is PAUSED, which is set on the children

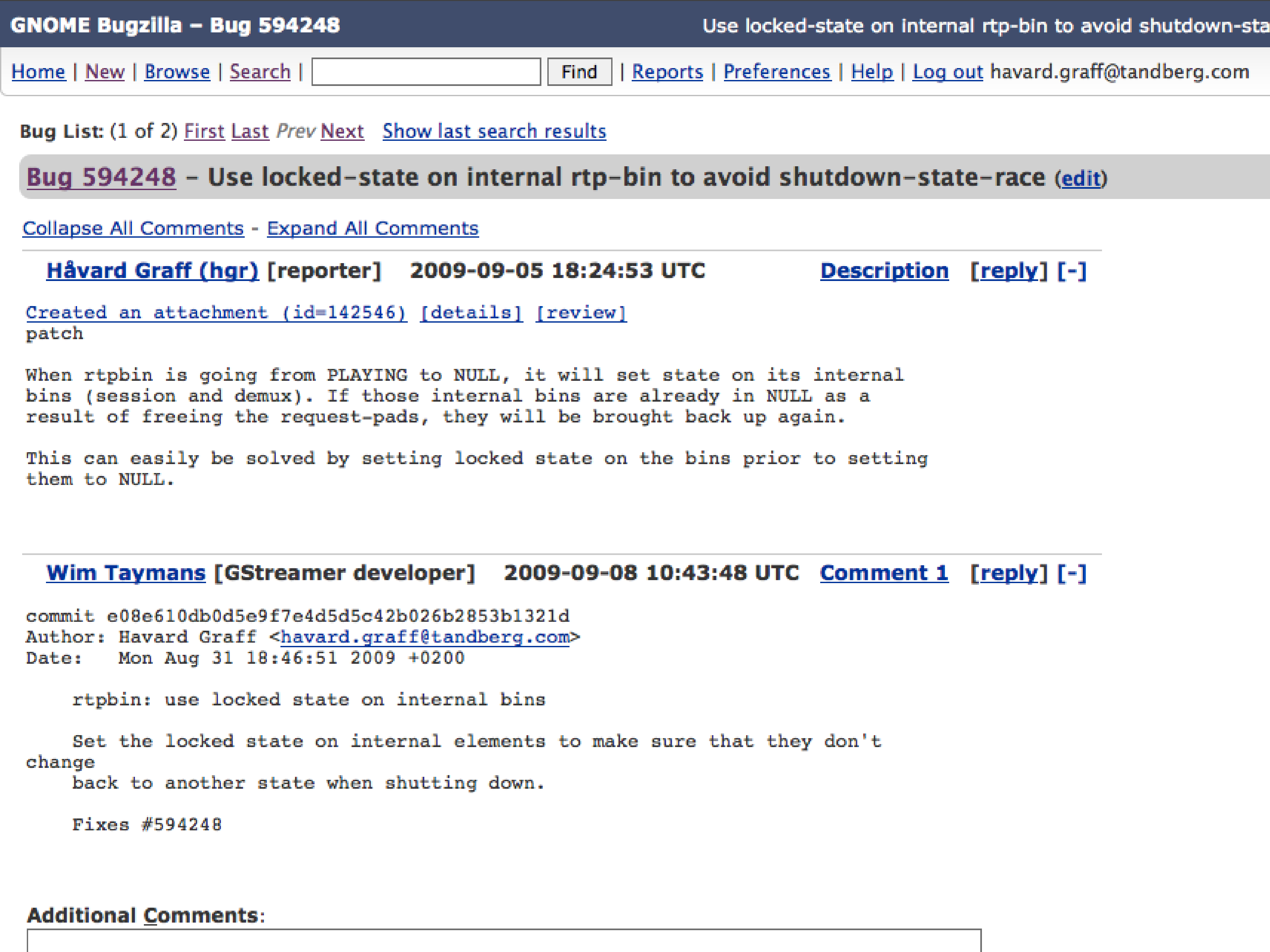


- States:
  - PLAYING
  - PAUSED
  - READY
  - NULL

Effectively bringing the element back up from NULL



- States:
  - PLAYING
  - PAUSED
  - READY
  - NULL



GNOME Bugzilla – Bug 594248 Use locked-state on internal rtp-bin to avoid shutdown-sta

Home | New | Browse | Search | Find | Reports | Preferences | Help | Log out havard.graff@tandberg.com

Bug List: (1 of 2) First Last Prev Next Show last search results

Bug 594248 – Use locked-state on internal rtp-bin to avoid shutdown-state-race (edit)

Collapse All Comments - Expand All Comments

Håvard Graff (hgr) [reporter] 2009-09-05 18:24:53 UTC Description [reply] [-]

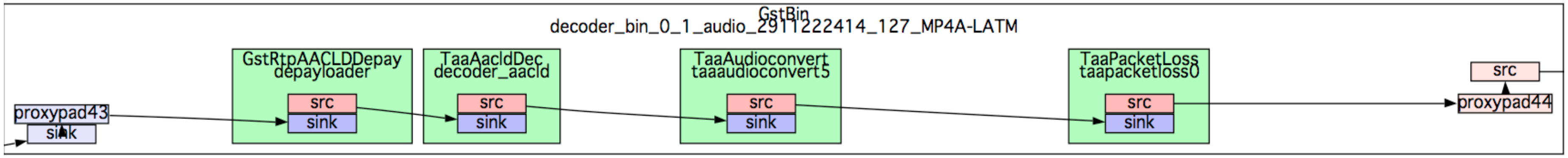
```
Created an attachment (id=142546) [details] [review]
patch

When rtpbin is going from PLAYING to NULL, it will set state on its internal
bins (session and demux). If those internal bins are already in NULL as a
result of freeing the request-pads, they will be brought back up again.

This can easily be solved by setting locked state on the bins prior to setting
them to NULL.
```

Wim Taymans [GStreamer developer] 2009-09-08 10:43:48 UTC Comment 1 [reply] [-]

```
commit e08e610db0d5e9f7e4d5d5c42b026b2853b1321d
Author: Havard Graff <havard.graff@tandberg.com>
Date:   Mon Aug 31 18:46:51 2009 +0200

    rtpbin: use locked state on internal bins

    Set the locked state on internal elements to make sure that they don't
change
    back to another state when shutting down.

    Fixes #594248
```

Additional Comments:

The lost-event

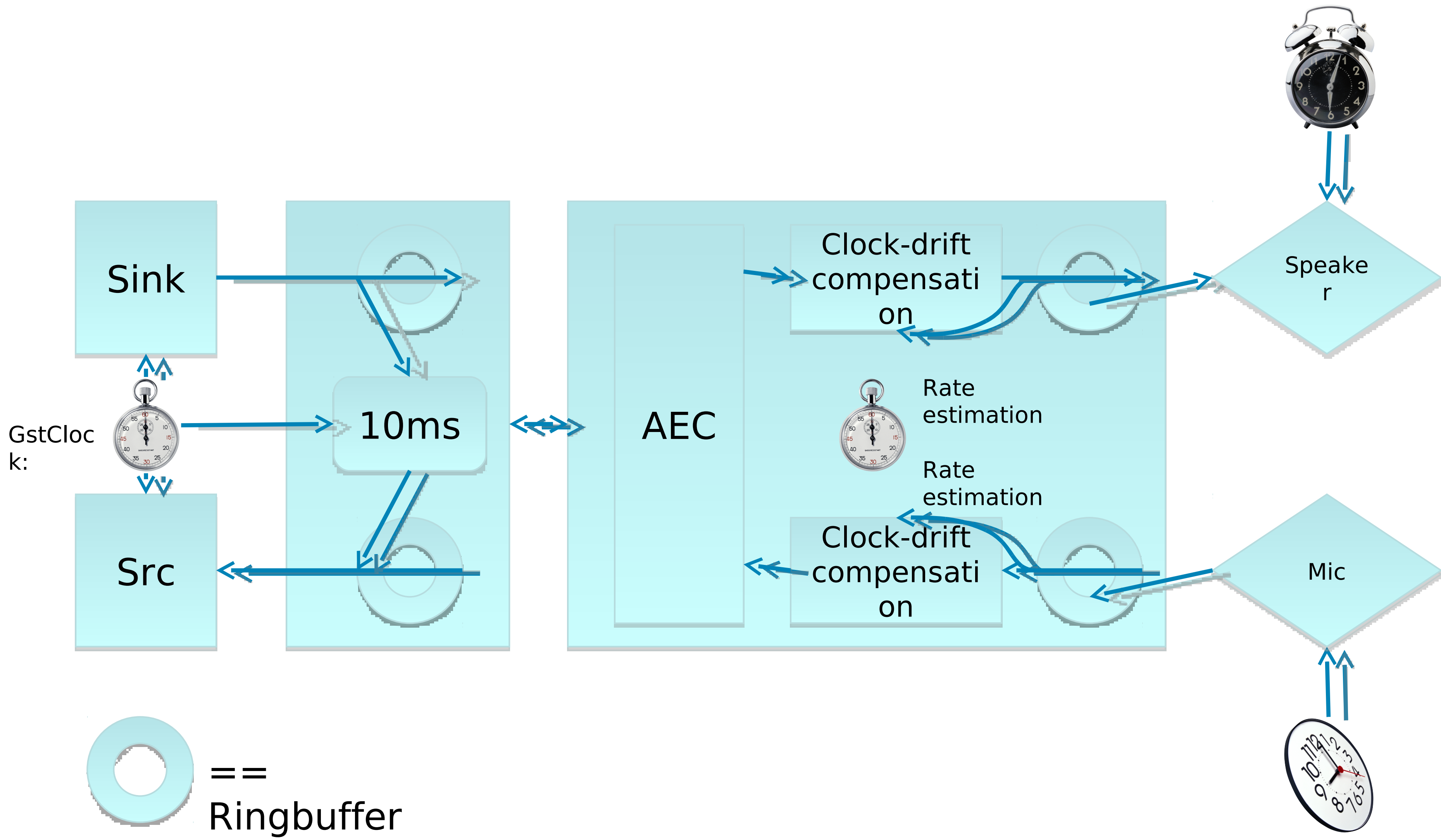


- Emitted from the GstRtpJitterBuffer
- Will wait for a packet until its last possible time to be played out expires
- We use the lost-event for:
  - Audio packetloss-concealment
  - Video decoder error-reporting
    - PLI / FUR
- FEC sits upstream from the jitterbuffer, reconstructing some lost packets

GstTestClock

- You have to manually drive the clock, as well as releasing the waits
- Make deterministic synchronization tests
- What really goes on inside a live pipeline?
- Done some work here, a lot more to come
  - Lipsync, latency etc…
- Not yet upstream, coming soon!

The Audio System

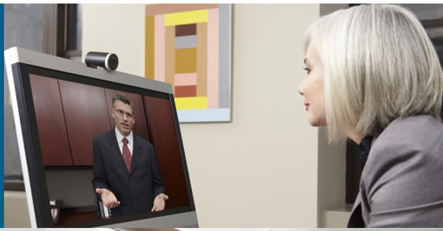


Acoustic Echo Canceller

- Gives -35dB
- Works with all known sound devices, on Linux, OSX and Windows
- Double-talk! (not suppression)
- Several TANDBERG patents involved
- Same code that runs on the endpoints

One more thing…

THANK YOU!
(questions?)



See you on #gstreamer ☺

TANDBERG is now part of Cisco.