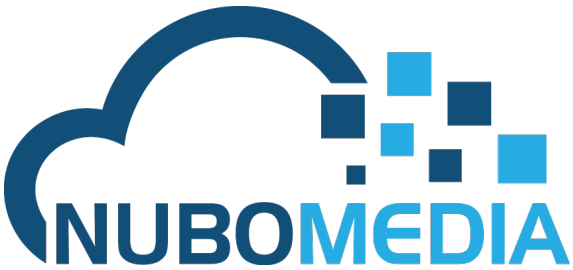
D4.3.1

Version	1.0
Author	NAEVATEC
Dissemination	PU
Date	27/01/2015
Status	FINAL



D4.3.1: Media Elements for Social and Immersive Environments v1

Project acronym:	NUBOMEDIA
Project title:	NUBOMEDIA: an elastic Platform as a Service (PaaS) cloud for interactive social multimedia
Project duration:	2014-02-01 to 2016-09-30
Project type:	STREP
Project reference:	610576
Project web page:	http://www.nubomedia.eu
Work package	WP4
WP leader	Javier López
Deliverable nature:	Prototype
Lead editor:	Javier López
Planned delivery date	01/2015
Actual delivery date	27/01/2015
Keywords	Kurento, media server, RTC, social media, group communications

The research leading to these results has been funded by the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement nº 610576

FP7 ICT-2013.1.6. Connected and Social Media

Contributors:

Javier Lopez (NAEVATEC)
Ivan Gracia (NAEVATEC)
David Fernandez (NAEVATEC)
Miguel París Díaz (URJC)

Internal Reviewer(s):

Luis Lopez (URJC)

Version History

Version	Date	Authors	Comments
0.1	03-01-2015	F. Javier Lopez	Initial Version
1.0	16-01-2015	Ivan Gracia	Generate final version for review

Table of contents

List of Figures:

Acronyms and abbreviations:

API	Application Programming Interface
AR	Augmented Reality
FOSS	Free Open Source Software
IMS	IP Multimedia Subsystem
IoT	Internet of Things
KMS	Kurento Media Server
MCU	Multipoint Control Unit
RTC	Real-Time Communications
RTP	Real-time Transport Protocol
SCTP	Stream Control Transmission Protocol
SFU	Selective Forwarding Unit
VCA	Video Content Analysis
VoD	Video on Demand
WebRTC	Web Real Time Communications

1 Executive summary

This deliverable contains a description of the social and group communication technologies that are under implementation in Nubomedia. These technologies, and particularly the conceived APIs, make possible to create RTC services where participants can interact, in real-time, following arbitrary interaction topologies and may play different roles.

In addition, this deliverable contains a brief revision of different immersive multimedia technologies and a discussion explaining why its inclusion is not appropriate at this time for Nubomedia, given its technological and exploitation strategy.

2 Introduction

2.1 State-of-the-art on social and group real-time communications

Real-Time Communications (RTC), and particularly multimedia RTC, are very often used for establishing group communication sessions. These sessions are commonly identified with different names such as conferences, video conferences, video rooms, etc. Currently, there are many different services (both paid and free) offering these type of capabilities including Google Hangouts, Skype, Facetime, WebEx, GoToMeeting, etc. In general, all of them provide a notion of "shared space" where users can find each other in real-time, communicate through different mechanism (i.e. audio, video, chat, etc.) and share documents or other information. However, from the RTC multimedia perspective, the implementation of the group communication capabilities may have one out of two flavors: MCU or SFU.

Multipoint Control Unit (MCU)

MCU is a concept initially introduced by the H.323 standard as a mechanism for enabling and managing group (i.e. multipoint) communications through a communication middlebox. Although there are different types of MCUs, the most popular are the transcoding and mixing ones. In this type of MCUs, group communications are enabled by multiplexing different audiovisual streams into a single composite one, which is usually created applying the following mechanism:

- For every participant, the multiplexed audio signal is the result of linear addition of all input signals except her own.
- For every participant, the multiplexed video signal is the result of mixing in a composite matrix all the input video streams, so that all participants receive a scaled down version of the video of any other participant but placed into a grid of N rows and M columns (being NxM the total number of participants)

Mixing MCUs have the advantage of optimizing bandwidth consumption because the NxM incoming video streams are received, on each participant, as one single composite stream. On the other hand, they also have drawbacks. The first one is that the composite mixing requires a decoding of all the input streams, a mixing process (combining the different streams in raw format) and an encoding process. All this tends to increase the latency and degrades QoE of end-users. In addition, the transcoding and mixing is a very CPU consuming process, making MCUs require costly hardware when the number of participants in the session is large. To conclude, another drawback is that when the number of participants is large, the above described scheme based on mixing all the incoming streams, is not practical.

Selective Forwarding Unit (SFU)

SFU is a concept introduced by the IETF VTCORE WG. It refers to the capability of a middlebox to provide group communication by switching or routing video streams. SFUs are conceptually equivalent to switching MCUs and, for this reason, they are sometimes called "video routers". Dropping the very low level technical details, an SFU is just a routing and forwarding mechanism which makes possible to forward one incoming media stream coming from a participant to N outgoing media streams to be received by other participants. SFUs make possible to switch (i.e on/off) the video forwarding so that the receiver application is able to choose whether to receive or not a particular media stream.

SFUs have the advantage of not requiring any heavy processing given that they do not require any kind of transcoding or mixing. However, they have the disadvantage of demanding much higher bandwidth from participants (i.e. if there are N participants, any of them wishing to visualize the video of the whole group needs to receive N video streams).

Independently on whether the MCU or the SFU model is used, a revision of current state-of-the-art services, such as the one enumerated above, and of current state-of-the-art RTC APIs, such as the ones of Lynckia, Tokbox, Temasys, etc., all of them share the same philosophical principle: groups are flat. This means that the APIs behave symmetrically for all users, which makes difficult and cumbersome to create applications involving rich group topologies or different group roles.

2.2 State-of-the-art on immersive real-time communications

Current trends in immersive multimedia show lot of activity in two main areas: stereoscopic 3D video and virtual reality displays. In relation to the former, 3D video is already a reality when dealing with TV broadcasting services. There also different types of available 3D services for Video on Demand (VoD). However, these type of media capabilities are not permeated into the RTC arena and only experimental setups have been tested in lab environments. One of the reasons for this comes from the lack of efficient and cheap technologies (i.e. MVC efficient codec implementation are proprietary and have associated prohibitive costs for many RTC business application scenarios) In relation to the later, the emergence of new devices such as the Oculus Rift is opening new and interesting usage scenarios for content industries. However, again, these seem to be far beyond RTC scenarios at the time of this writing.

In particular, and in relation to WebRTC technologies, none of these immersive media capabilities are currently under consideration in the different standardization bodies and developer groups.

3 Objectives

This document has the objective of presenting the technological strategy for social and immersive multimedia capabilities in Nubomedia. In particular, in relation to social and group communications, this document presents the main ideas relative to the conception, specification and implementation of a rich toolbox of group communication capabilities. For this objective to be executed, a number of sub objectives need to be fulfilled:

- To conceive and design the most appropriate technological strategy providing a rich tool-box of group communication capabilities for Nubomedia.

- To implement such capabilities as KMS modules, so that they can be deployed as part of KMS instances.

4 Implementation strategy

4.1 Implementing social and group real-time communications

Group communications are one of the main requirements of RTC services both, among Nubomedia partners, and among other RTC users out there. For this reason, we have decided to follow a very ambitious implementation strategy for these types of features. This strategy is based in the following axes:

- We want Nubomedia to provide group communication capabilities through transcoding/mixing MCU models
- We want Nubomedia to extend state-of-the-art mixing models to provide layouts different to grids. In particular, we include in our strategy the support for alpha-blending mixing models that, combined with background extraction techniques, may provide novel and interesting models for group communications where the different participants may share a given environment (i.e. background).
- We want Nubomedia to provide group communication capabilities through forwarding/routing SFU models
- We want Nubomedia to expose all these capabilities through a consistent API suitable for the creation of rich topologies and group roles among participants.

As a result, our implementation strategy requires implementing all major models available in current state-of-the-art plus some extensions associated to novel mixing schemes.

Carrying out this strategy requires the following:

- To create the appropriate transcoding and mixing capabilities required by the MCU models. This can be achieved by adapting the appropriate GStreamer features, and integrating them as modules in KMS.
- To design the appropriate media routing capabilities and implement them into KMS as one or several modules.

4.2 Implementing immersive real-time communications

At the time of this writing, the consortium does not plan to integrate any kind of immersive capability into Nubomedia. This is due to the following reasons:

- From the agile project management methodology no requirements including these types of features have emerged. Hence, the project needs to pivot to prioritize other features related to WebRTC capabilities that have more priority.
- Open source technologies related to immersive environments are almost inexistent and, the few initiatives that exist are immature or have been abandoned.
- Current Nubomedia technological and transfer strategy is concentrated on providing a full and mature WebRTC stack. This is incompatible with the incorporation of 3D features given their lack of support on WebRTC standards and software stacks.

5 High-level architecture

5.1 Social and group real-time communications

For implementing Nubomedia group communications, we have selected to use a number of abstractions capable of providing the different MCU and SFU described above. The following subsections are devoted to introducing them:

5.1.1 Dispatcher model

The Dispatcher group communications capability implements a media router following the SFU without requiring any kind of transcoding or mixing. Its architecture is depicted in Figure 1

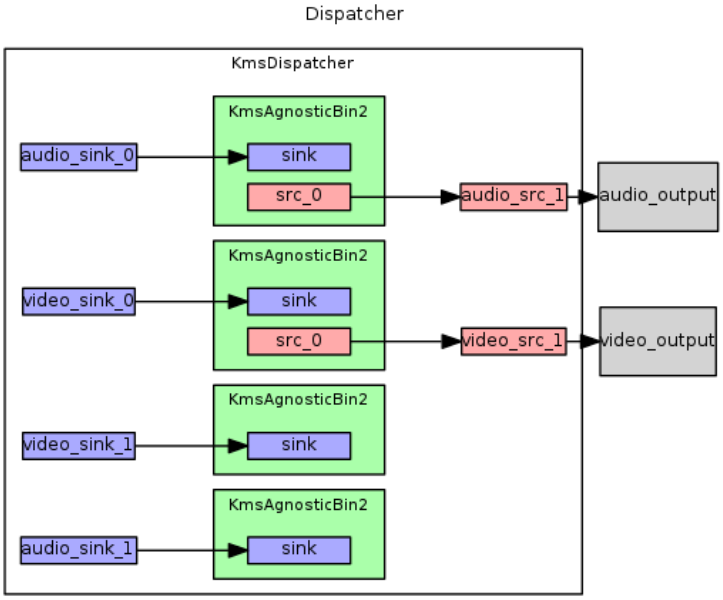


Figure 1: Dispatcher internal structure

In a Dispatcher with N media inputs, the API will made available the following forwarding mechanisms:

- Any give user can chose to receive any of the available N media input streams.
- Video and audio, when available, are routed together, so that receiving a media streams involves receiving two flows: one for the audio and one for the video.

5.1.2 Composite model

The Composite group communication capability implements a media mixing following the MCU model, which requires one decoding for every input stream, the mixing of all input streams, and one encoding for the resulting output stream. Its architecture is depicted in Figure 2

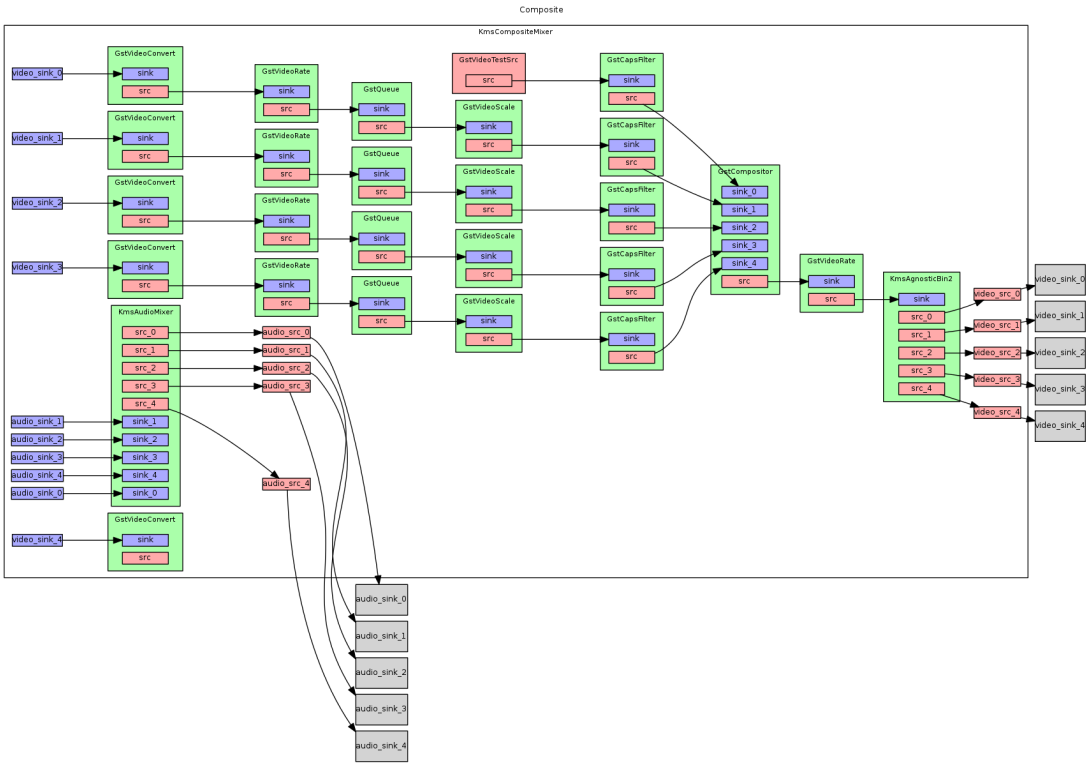


Figure 2: Composite internal structure

In a Composite with N media inputs, the API will allow the following mechanisms:

- All users will receive as video output stream a composite grid (with two columns) mixing all the available input streams with the appropriate scaling.
- All users will receive as audio output stream the addition of all audio input streams, except her own.

5.1.3 AlphaBlending model

The AlphaBlending group communication capability implements a media mixing following the MCU model, which requires one decoding for every input stream, the mixing of all streams and one encoding for the resulting output stream. As a difference with the composite, in this case the mixing is based on alpha-blending, which means that all streams are overlaid and their transparency (i.e. alpha) channels are applied. This mixing makes only sense when all streams except one make transparent relevant parts of the viewport (usually the background). Its architecture is depicted in Figure 3.

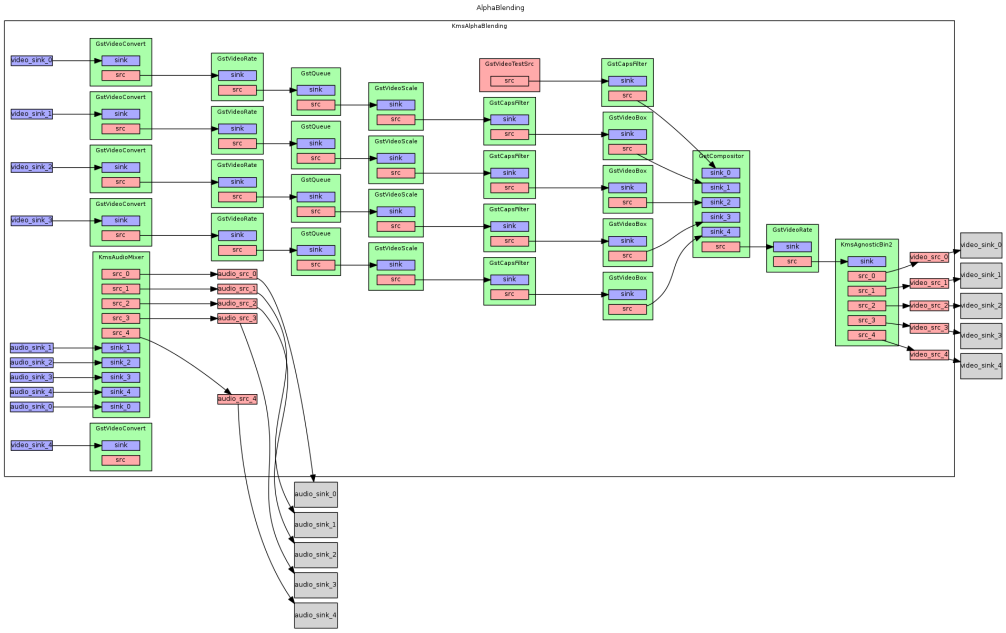


Figure 3: Alpha Blending internal structure

6 Software architecture

Hubs are the family of elements enabling group communications. Each MediaElement that connects to a hub will do so through a special MediaElement called HubPort, which can be connected and disconnected from Hubs to inject or receive media.

The base class for all hubs is the abstract class BaseHub, and provides extending classes with the ability to connect and disconnect HubPorts for media transmission. The following diagram shows the inheritance model, and the different types of hubs that exist. It is clear from, that hubs are another type of MediaElement, that are not KmsElements.

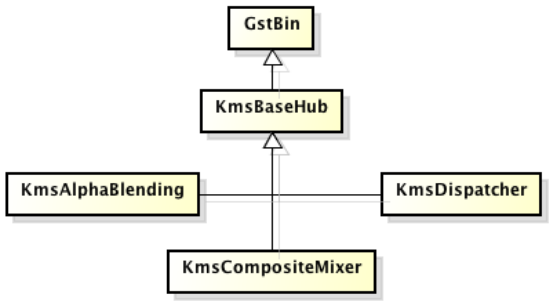


Figure 4: KmsBaseHub UML diagram

As stated before, Hubs send and receive media through HubPorts, a special type of KmsElement. This implies that, in order to send/receive media from/to a hub element, it will have to first be connected to a HubPort, and the port in turn will have to be connected to the hub itself.

6.1.1.1 Composite

This element allows combining several input streams in one output stream, divided in two columns, and as many rows as needed depending on the incoming streams. Thus, if there are eight input streams, the output stream will have two columns and four rows.

All videos will have the same size in the grid, even if their size is different. The streams will appear in the grid from left to right, top to bottom in order of connection.

One of the most appealing uses of this hub, is to save bandwidth in a multiconference scenario. Without it, if 10 users are to hold a meeting, each user would be emitting it's own media, and receiving the media from the other 9 participants. Having a total of 10 connections per user, this amounts to 100 connections in the media server. Apart from the amount of bandwidth consumed in the server, these numbers are more important in the world of mobile communications, where portable devices connected through 3G/4G networks, depend heavily on the quality of their link, and most of the times have monthly quotas for internet traffic. Thus, combining all streams into one single feed, each user can have only one incoming media feed, having a total of only 2 connections per user. This does not only save bandwidth, but in some cases also computing power, since some transports encrypt media (WebRTC, for instance), which is a power-hungry operation. So by reducing the amount of connections, we can also reduce the computational power required in the client, thus allowing less powerful devices to participate in such multiconference applications.

The following figure represents the internal structure of a Composite element, with it's internal GStreamer elements shown in the KmsCompositeElement box. The grey boxes outside of this box, represent the audio and video sinks of 5 different HubPorts, and appear in the schema in order to conceptually show how the video goes from the source pads in the KmsCompositeMixer, to the sink pads in the HubPort.

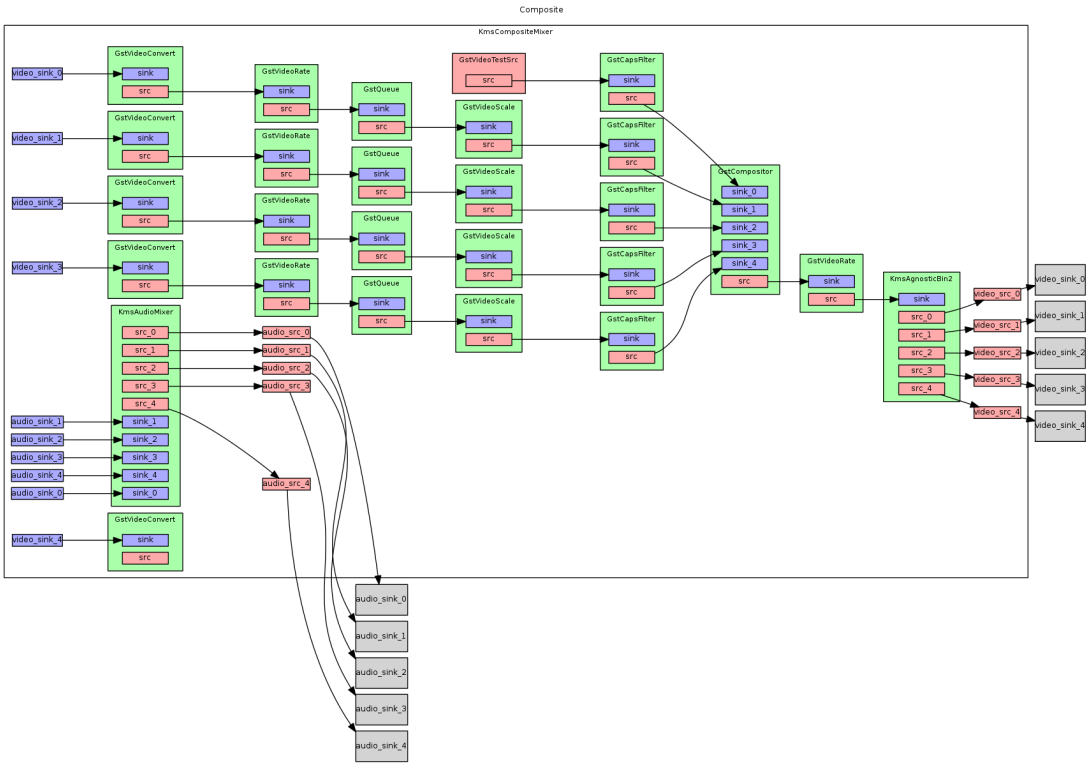


Figure 5: Composite internal structure

6.1.1.2 Dispatcher

The dispatcher is a hub that allows selecting which media stream will receive each of the clients connected to it.

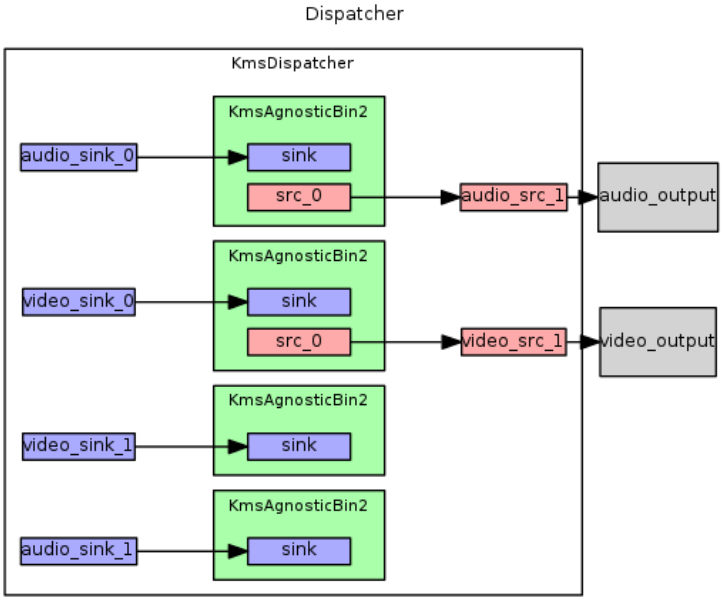


Figure 6: Dispatcher internal structure

6.1.1.3 Alpha blending

This type of hub allows showing several input streams over a master stream. This master stream defines the output size, and the other streams should adapt, according to their configuration, to the master. The internal structure is represented in the following picture. As in the Composite, the grey boxes outside of the KmsAlphaBlending box represent different HubPorts connected to it.

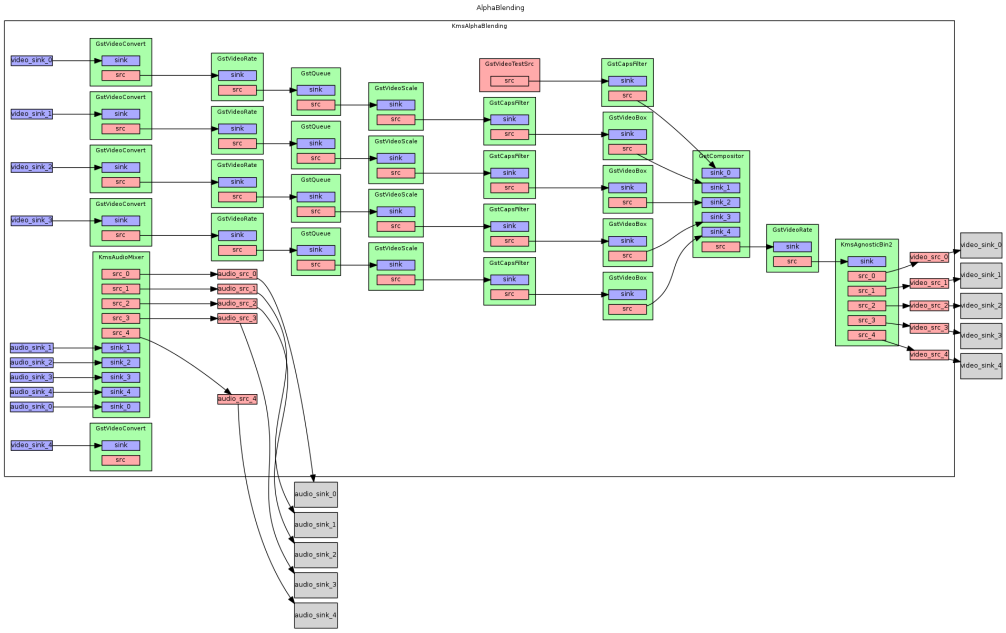


Figure 7: Alpha Blending internal structure

All streams not acting as master can be configured in their size and position (relative to the master), defining where the stream should be shown. The Z coordinate can also be modified, in order to offer several presentation layers.

7 Implementation status (Nubomedia Release 3)

7.1 Licensing models and restrictions

All software artifacts belonging to Kurento Media Server described in this document have been released and made public through the GNU Lesser General Public License (LGPL) v2.1, with the exception of the Alpha-blending capability, which has not been released and is still maintained ad proprietary software © Naevatec.

Kurento Media Server extends and uses many parts of the GStreamer software stack. All these extensions and modifications have been shared with the GStreamer community in compliance with the GStreamer open source software license: LGPL v2.1.

The distribution of Kurento Media Server does not involve the distribution of GStreamer nor any of its plugins. In particular, Kurento Media Server software bundles don't integrate any kind of codec implementation. The installation of Kurento Media Server may require users to obtain and install, at their own risk, different codecs which may be protected by royalties and patents.

7.2 Obtaining the source code

All source code belonging to the Kurento Media Server described in this document can be obtained in the Kurento Github repository accessible at the following URL:

- https://github.com/kurento

In particular, all FOSS group communication capabilities source code is part of the kms-elements project, which can be found at the following URL:

- https://github.com/Kurento/kms-elements

7.3 Obtaining the documentation

Kurento Media Server documentation has been created in Sphinx, a powerful documentation facility which transforms text documents written in the reStructuredText format into hyper-linked documentation in several formats, which include HTML, PDF, epub and LaTeX.

Kurento Media Server documentation has been made public in the following project located inside the Kurento Github repository.

- doc-kurento
  - Description: Public Kurento.org documentation
  - Source: https://github.com/Kurento/doc-kurento
  - URL: Documentation is public in http://www.kurento.org/docs/current

8 References

[1] https://tools.ietf.org/html/draft-ietf-avtcore-rtp-topologies-update-05