ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

D2.1.1 – Reference architecture – year 2

MOS2S

Media Orchestration from Screen to Screen

DRAFT VERSION

Edited by: O.A. Niamut (TNO)

Contributions from:

- Rolf Biesbroek, Dolf Schinkel (KPN), Max Wreckers (Inmotio), Cyril Rutten (GameOn), Martin Wielaart (Amsterdam ArenA), Aschwin Brandt, Johan van der Geest, Joost Adriaanse (TNO)
- Phillippe Dobbelaere (Nokia)
- Seong Yong Lim (ETRI)

Version: v03

Date: 25-06-2018

Delivery date: 30-06-2018

Project key data

ACRONYM and full-length title

15022	MOS2S
Program Call	ITEA 3 Call 2
Full-length Title	Media Orchestration - Sensor to Screen
Roadmap Challenge	Urbanisation

Description

Novel and ubiquitous consumer-priced audiovisual sensors and data in particular, represent an important aspect of the Smart City environment, enabling a variety of applications for citizen information, participation, entertainment, experience, safety and security. Every user becomes a potential source of information, either directly or through social media buzz and its discovery. Audiovisual media provide citizens with Smart City data readily accessible to human senses. With the MOS2S project (Media Orchestration from Sensor to Screen), an international consortium of partners will develop and test audiovisual Smart City technologies and solutions in the context of citizen needs, and embed these solutions within the Smart City Playground.

Project duration & size

Size	Effort: 133.67 PY	Costs: 13.9 M€
Time frame	Start: 2016-10-1	End: 2019-09-30 (37 months)

Coordinator

Netherlands	TNO
Type	Research Institute
Contact Person	Gjalt Loots
Email Address	gjalt.loots@tno.nl

Consortium

Belgium	Nokia, iMinds, Kiswe Mobile, VRT*
Korea, Republic of	ETRI*, Mooovr
Netherlands	Amsterdam ArenA*, Bosch Security Systems B.V., Game On, Inmotio Object Tracking BV, Koninklijke KPN NV, TNO
Turkey	Bor Software inc.*, DİA Yazilim San. ve Tic. A.Ş., KoçSistem, TMOB BİLİŞİM

Table of Contents

PROJECT KEY DATA **2**

ACRONYM and full-length title *2*
Description *2*
Project duration & size *2*
Coordinator *2*
Consortium *3*

TABLE OF CONTENTS 4

LIST OF FIGURES **5**

PROJECT ACRONYMS **6**

1. INTRODUCTION 7
2. DESCRIPTION OF INDIVIDUAL ARCHITECTURES 8

2.1. Video capture platforms *10*
2.2. Video orchestration platform *10*
2.3. World Wide Stream platform (WWS) *12*
2.4. Wall of Moments platform (WoM) *13*
2.5. Kiswe Production Studio platform (KPS) *15*

3. MAPPING OF PLATFORM FUNCTIONALITY TO GENERIC BLOCKS 17

3.1. Terminology *17*
3.2. World Wide Stream platform (WWS) *22*
3.3. Wall of Moments platform *24*

4. MAPPING OF PLATFORM FUNCTIONALITY TO Y2 DEMO SCENARIOS 25
5. CONCLUDING REMARKS 27

List of Figures

Figure 1: overview of NL platforms and components. 8
Figure 2: overview of BE platforms and components. 9
Figure 3: 360/VR capture and monitoring architecture overview. 10
Figure 4: tiled streaming approach. 11
Figure 5: tiling workflow. 11
Figure 6: WWS architecture overview. 12
Figure 7: WoM architecture overview. 14
Figure 8: KPS architecture overview. 16
Figure 9: operator. 17
Figure 10: Example of a platform architecture. 19
Figure 11: generic reference architecture for video orchestration platform. 20
Figure 12: generic reference architecture for WWS platform. 22

Project acronyms

3DoF	3 degrees of freedom
BE	Belgium
CPA	Conformance Points A
CPB	Conformance Points B
DVB	Digital Video Broadcasting
IM	Instant Messaging
IT	Italy
KR	Republic of Korea
MOS2S	Media Orchestration Sensor To Screen
NL	(The) Netherlands
OB	Outside Broadcasting
PTZ	Pan-Tilt-Zoom
TR	Turkey
UHD	Ultra-High Definition
VR	Virtual Reality

1. Introduction

In Deliverable D2.1.1 the MOS2S consortium partners provide the reference architecture of the MOS2S platform, and sketch the approaches towards platform integration, related to the 2nd year use case and demonstrator. D2.1.1 is the first deliverable from WP2, in which we outline the reference architecture, starting from on the one side the available legacy platform architectures and on the other side the requirements to come to a unified architecture that at the same time allows interoperability with the legacy platforms and is a solid basis to define and implement the applications and new platform components required in WP3.

In Chapter 2, we provide a description of the individual legacy architectures, including block diagrams from D1.2.1. In Chapter 3, we then provide a mapping of these legacy architectures to generic blocks and in Chapter 4, we sketch reference architecture for platform integration between the legacy platforms, driven by the 2nd year use case.

2. Description of individual architectures

The MOS2S platform, depicted in Figure 1, enabling the Dutch 1st year use case scenario "enhancing live event experience through data-drive interactive video applications" consists of the following main parts:

- A camera capture and ingest platform for video acquisition;
- A real-time player tracking system for data acquisition and analysis;
- A video processing platform for stitching and video/data alignment;
- A video orchestration platform for interactive video streaming.

In the remainder of this document, we focus on the video orchestration platform as a candidate for integration with platforms from other partners.

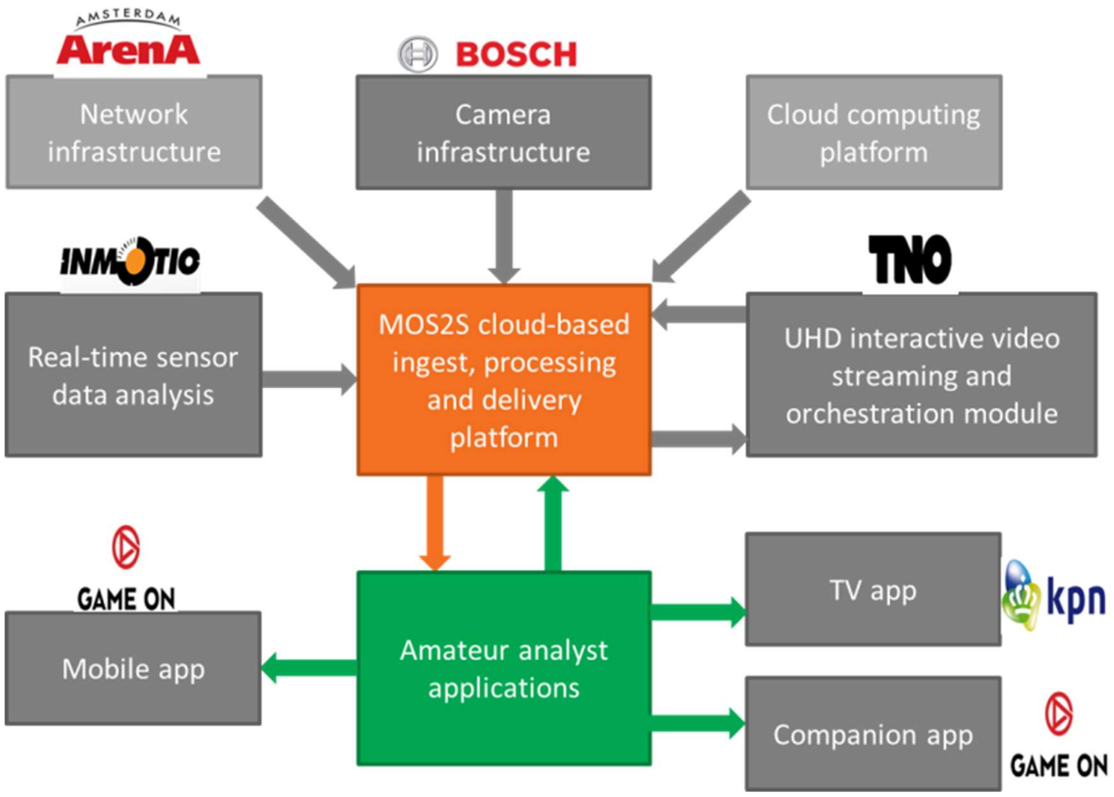

AMSTERDAM ArenA
Network infrastructure
BOSCH
Camera infrastructure
Cloud computing platform
INMOTIO
Real-time sensor data analysis
MOS2S cloud-based ingest, processing and delivery platform
TNO
UHD interactive video streaming and orchestration module
GAME ON
Mobile app
Amateur analyst applications
TV app
kpn
Companion app
GAME ON

Figure 1: overview of NL platforms and components.

The MOS2S platform, depicted in Figure 2 enabling the Belgian 1st year use case scenario "enhancing pre/post-experience through controlled co-creation of live fan event reporting" consists of the following main parts:

- The Wall Of Moments platform as a set of software tools and services that support co-creation scenarios of editorial teams and end users;
- The World Wide Stream platform for multi-modal stream processing;

- Kiswe Production Studio platform handling and publishing incoming media streams.

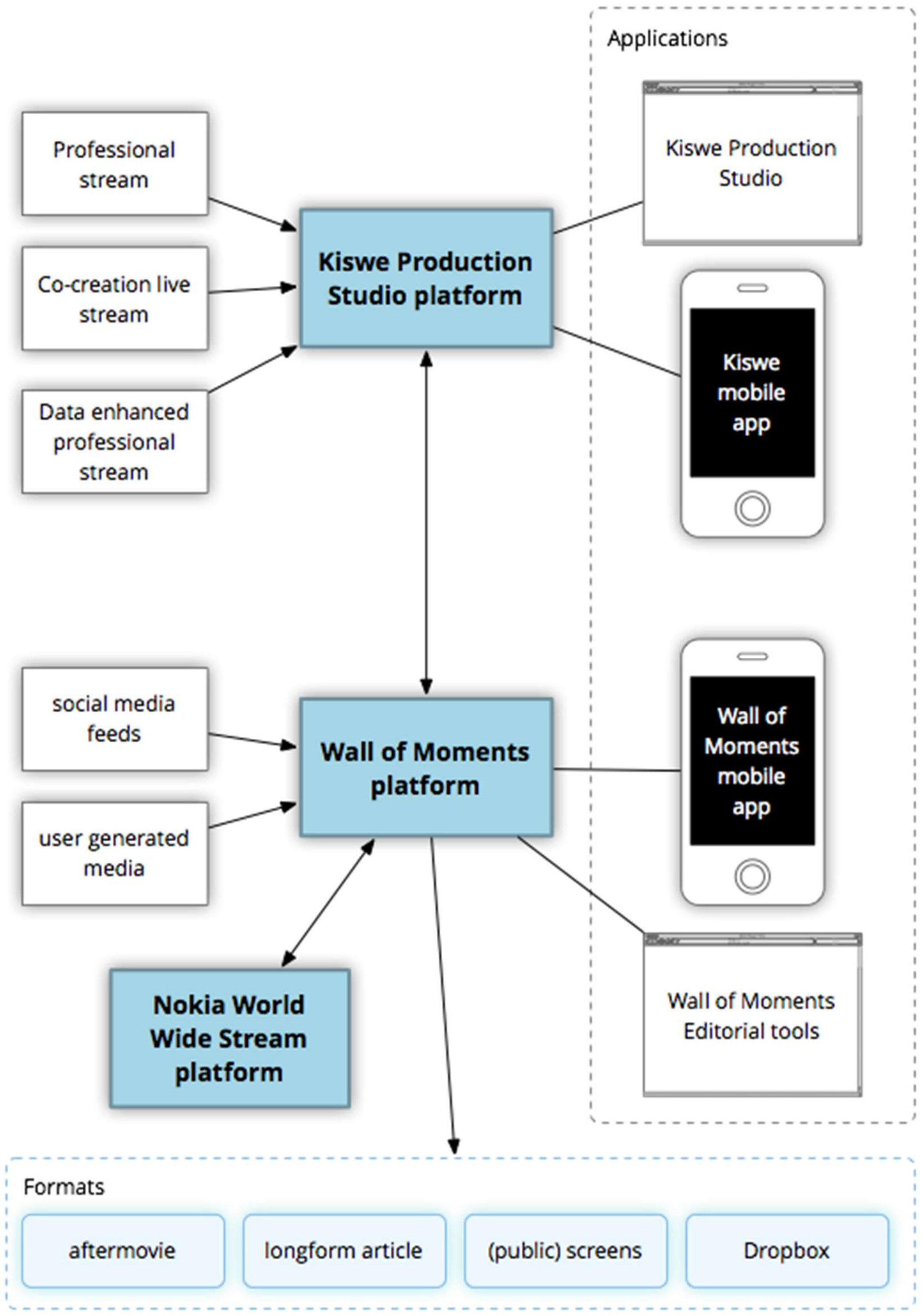


Figure 2: overview of BE platforms and components.

2.1. Video capture platforms

360/VR video capture and monitoring platform

Figure 3 shows the 360/VR video capture system and the monitoring system in this project. The capture system consists of commercial camera components with a rig system. On the other hand, the monitoring system receives the live feeds from the cameras and renders in three dimensional domain.

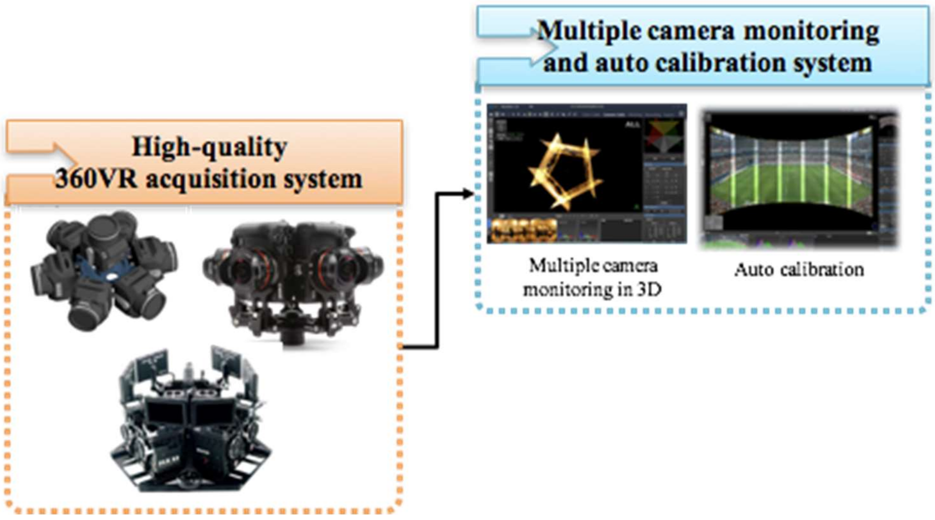
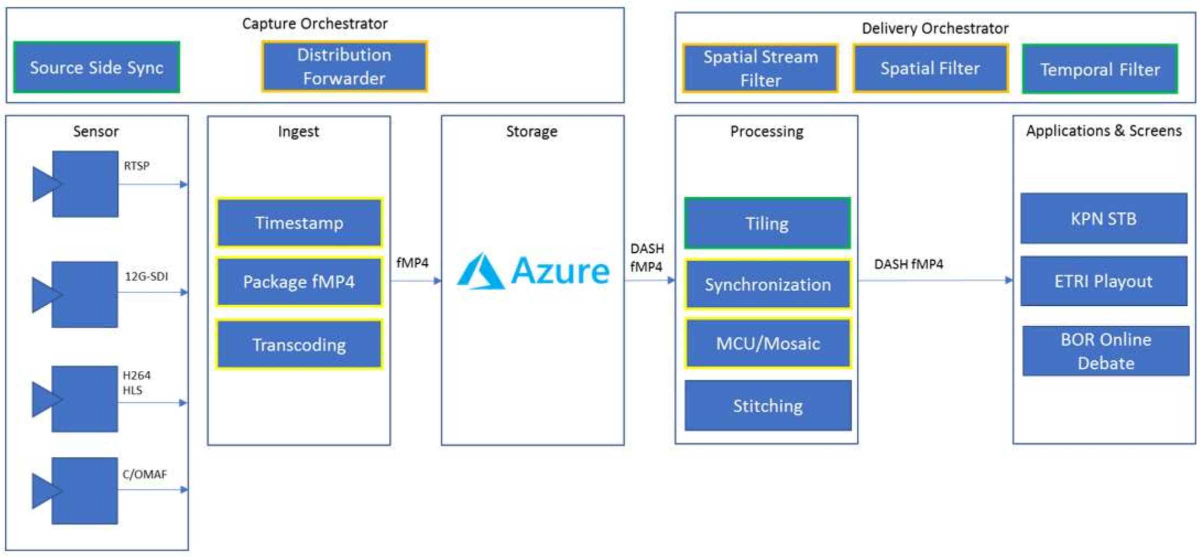

Multiple camera monitoring and auto calibration system
High-quality 360VR acquisition system
Multiple camera monitoring in 3D
Auto calibration

Figure 3: 360/VR capture and monitoring architecture overview.

2.2. Video orchestration platform



The video orchestration platform is responsible for pre-processing the video data prior to delivery. Its main tasks are i) storing the video in a format suitable for streaming; ii) thumbnail generation;

iii) storing the meta data associated with the video and exposing these data through a REST interface; iv) providing interfaces to the data capture platform and the video delivery platform; v) applying computer vision algorithms to the video data; vi) stitching multi-camera panoramas.

The video delivery platform concentrates on tiled streaming, a methodology for interactive streaming of ultra-high resolution content. By letting the client specify a Region of Interest (RoI) and only downloading these parts of the video the client can use the full resolution of the RoI while the rest of the pixels are not downloaded. Figure 4 shows the high-level working of the tiling process.

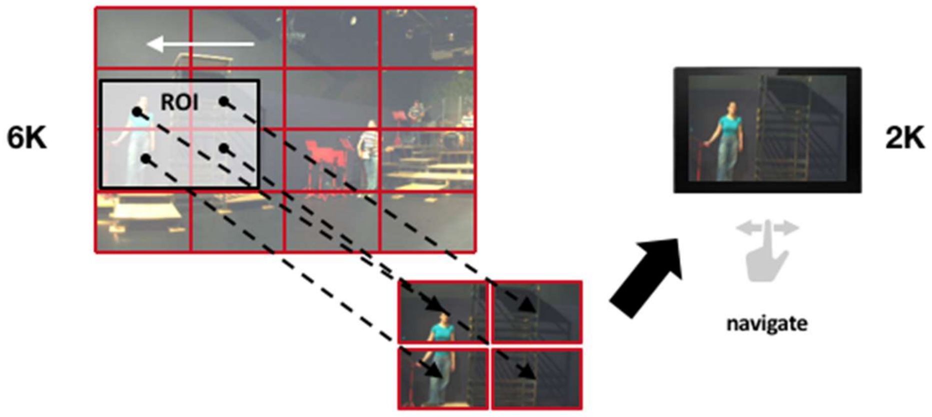


Figure 4: tiled streaming approach.

The goal of the tiled streaming within MOS2S is improving the tiled streaming backend so that it can process 4K video at the dimensions of the ArenA IP cameras and assisting in the implementation of the tiled streaming technology in the GameOn iPad application. To reach this goal the following was done on the master and the workers (see Figure 5):

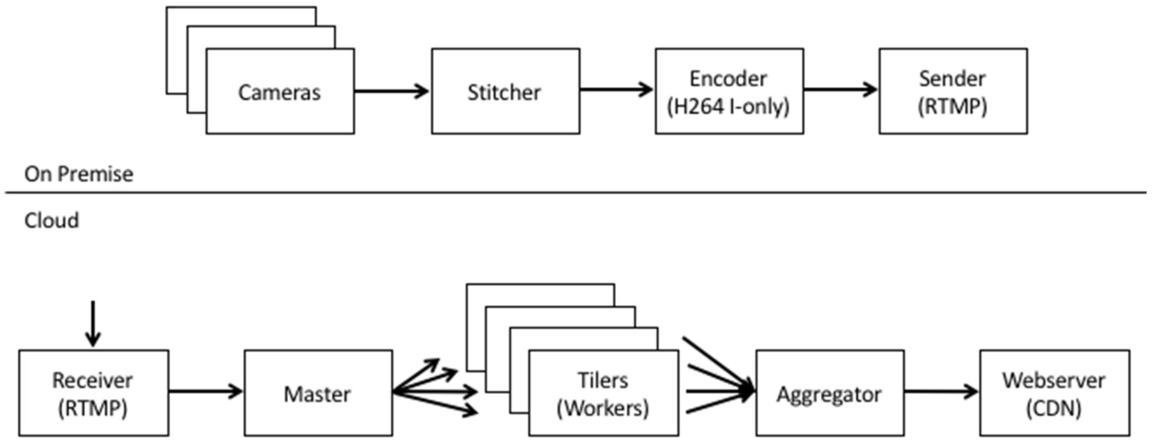


Figure 5: tiling workflow.

- Migrating the tiled-streaming backend from Amazon Webservices to Microsoft Azure with partner Innovativ.
- Sending RTP stream over UDP from the Amsterdam ArenA to the tiled streaming backend to test the live-tiling functionality with the new cameras during a soccer match.
- Developing new tiling-profile (spatial manifest file for specific use in the GameOn app).
- Improving reliability of the tiled streaming backend by performing load-tests with high number of workers and heavy workload and improving existing code-base.

2.3. World Wide Stream platform (WWS)

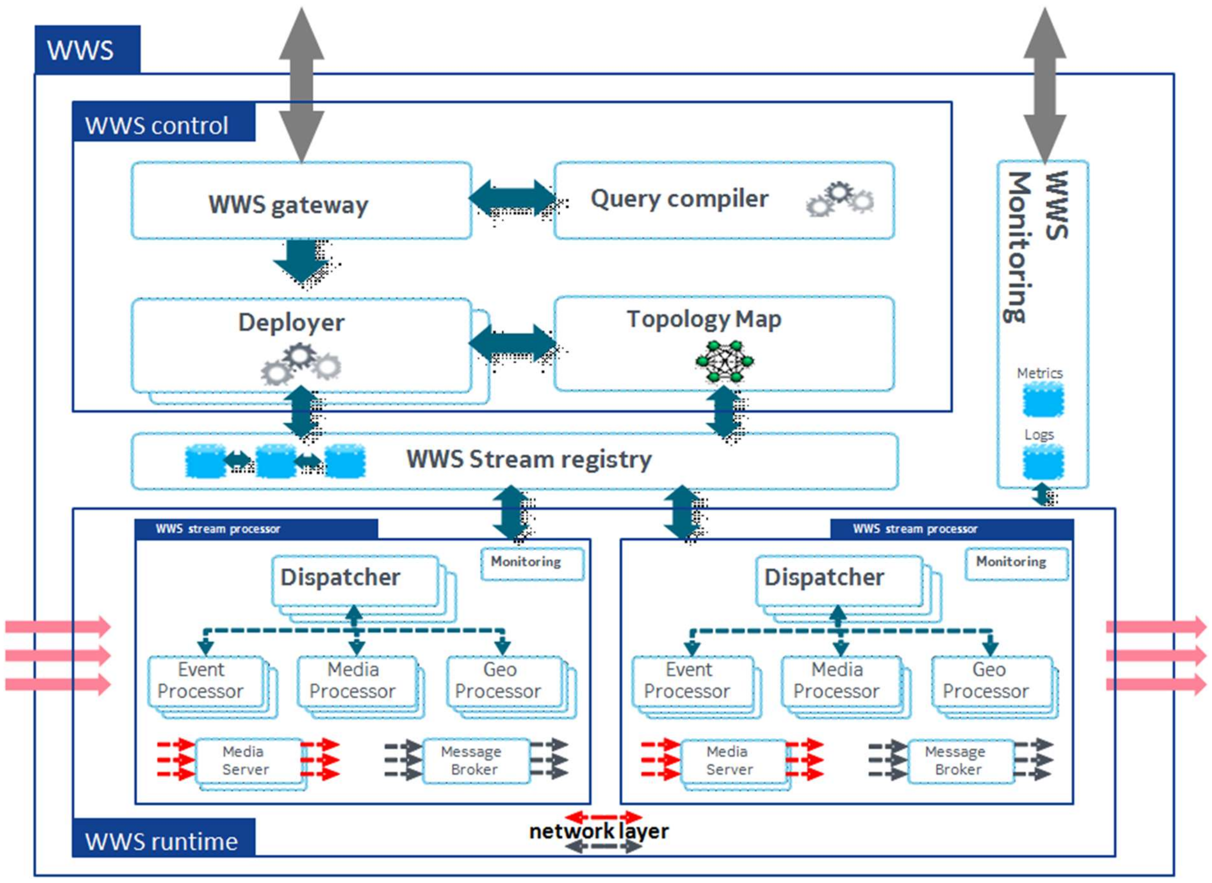


Figure 6: WWS architecture overview.

WWS is a large-scale geographically distributed stream processing platform that allows multimodal stream processing across dedicated HW, edge and core clouds. Multimodality is key - WWS is optimised to do all of media-to-media, media-to-data, data-to-data transformations. Data-to-media is currently not a focus, but could be explored together with the VRT in their role of media producer. Currently not available, but certainly the focus of MOS2S is the data-toknowledge transformation, which will be an important stepping stone in the integration of multimedia and sensor streams into the redaction and production functions of MOS2S.

Any WWS application can have parts running outside WWS (e.g. a GUI running in a browser on an end-user device), and parts running inside WWS. To facilitate the connection between the external and internal parts, WWS exposes internal streams on a HTTP interface. Additionally, individual stream processors can implement exposure points where needed. To connect external streams, the platform exposes the native AMQP interface and a HTTP endpoint for datastreams and typical mediaserver interfaces for audio and video streams. To connect to specific streaming services with their own dedicated API (social media streams, …), WWS will create suitable stream sources that can be deployed as stream operators. WWS relies on external platforms with sensor gateway capability to connect individual sensors. A commercially available solution is the Nokia Impact platform. During the course of the project, we will also create interfaces to IMEC's streaming extension of the Datatank platform.

The WWS control layer only takes responsibility for the parts of the application that runs within WWS. This WWS-internal part needs to be specified as a dataflow graph with input streams, output streams and stream operators. The dataflow is specified in typescript (https://www.typescriptlang.org/). The WWS control layer will compile this dataflow into a deployable representation, decide where to put the operators based on availability of processing resources, interconnectivity of the WWS runtimes and boundary conditions such as external stream connections and potential for reuse. The WWS stream registry is not only a distributed database that knows everything about available streams in WWS, but also the distribution point of actual deployment actions through the changefeed infrastructure of it's underlying technology (https://www.rethinkdb.com/). WWS operators run inside WWS processors. WWS already has dedicated processors for specialised functions (CEP on data streams, geoindexing, multimedia processing, domain specific operators for selected vertical industries,...) and a generic nodejs processor that can support any typescript/javascript used in the dataflow specification. Processors can run on dedicated HW (e.g. when requiring NVIDIA CUDA support) or on private or public clouds (e.g. Amazon). Processor elasticity is controlled by an internal mesos (http://mesos.apache.org/) / marathon scheduler infrastructure. During the course of MOS2S, individual operator placement (the role of the dispatcher) will also be brought under control of an innovative mesos scheduler that will be designed from scratch to cater for specific WWS requirements.

2.4. Wall of Moments platform (WoM)

The Wall of Moments platform is a set of automated software tools that function between the mobile app, the tool for the editorial team and the external services.

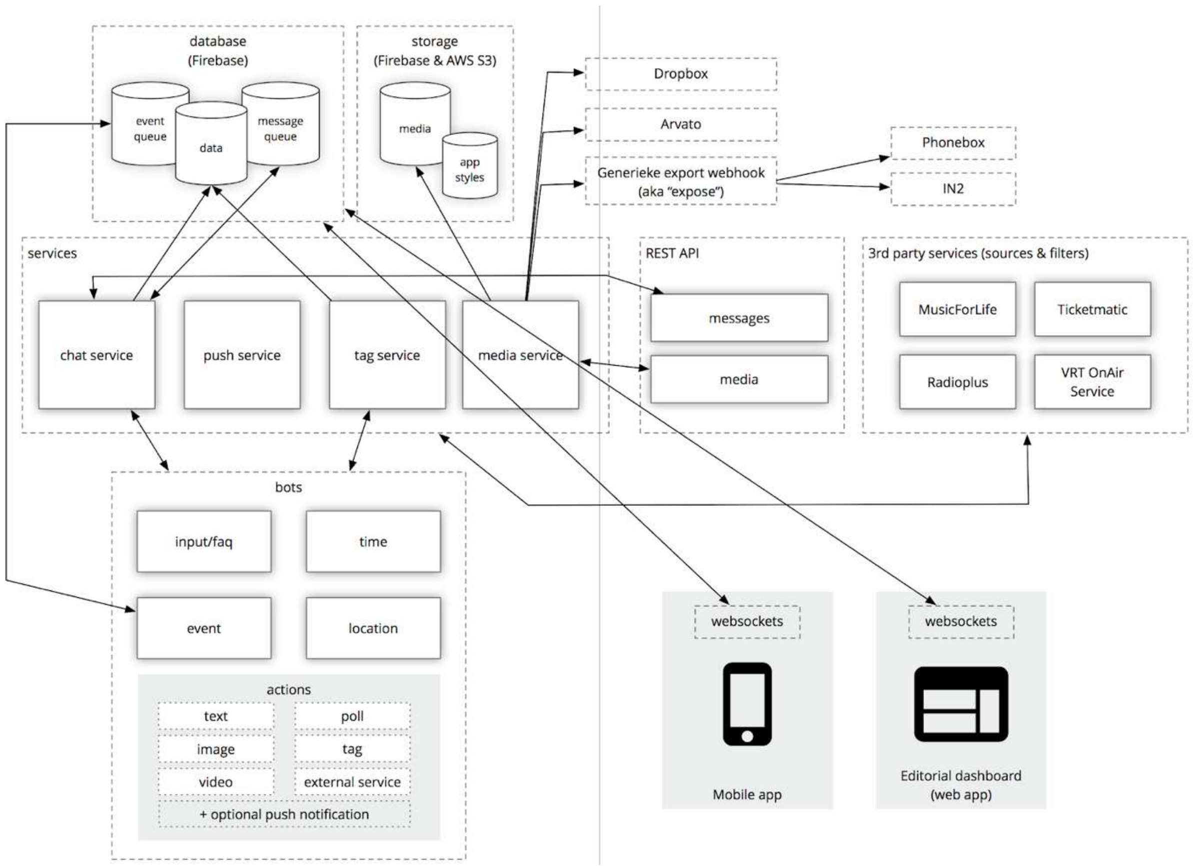


Figure 7: WoM architecture overview.

Data [(1)] is contained in a Firebase data storage bucket. There are several data processing queues to optimize the data workflow. Because of optimization reasons, social media data could be stored on a different platform. File storage [(2)] is based on Google Cloud services.

We have built - and are building - server based applications, written in Node.js [(3)]. They are the main services of the platform and are the connectors between the data and file stores and the bots [(10)] or external services.

The chat service [(4)] processes all messages between the end users and the editorial team or the bots. When the editorial team sends a message to a selection of users, the message will be duplicated for each user independently. Therefore in the future we could add some personalisation to these messages coming from the editorial grouped messaging.

The media service [(5)] processes all incoming photos and videos. It analyses the media assets and adds keywords to it. These keywords are derived by using an external service. We are doing some tests with the Clarify.io service. The media service also optimizes the pixel size of the images (resizing and cropping) for easier handling in the mobile app and the editorial dashboard. If a media asset is a video, the service will optimize the video to the appropriate video codec and screen size. The media service sends the media assets to a dedicated Dropbox account [(16)], so that the assets could be used in a different workflow (ex. for publishing on television). Media assets can also be sent to or received from the KISWE Production Studio platform [6]. The functionalities of the KISWE Production Studio platform are described in section 2.7 of this document.

The tag service [7] listens on requests to add tags to or remove tags from media assets, users and streams. Tags are the basis for filtering content and users.

The social media service [8] captures social media posts from Instagram and Twitter. It interacts with the Nokia Research World Wide Stream platform [9] which will analyse the social media data. The requirements for the WWS platform are described in section 2.5 of this document. The details of the integration is described in section 4.6 of this document.

Bots [10] are a special type of automated services that optimize the interactions with the end users. We have defined 4 types of bots: input based, time based, event based and location based bots.

An input bot (or FAQ-bot) [11] listens on the questions that the end user sends through the mobile app. The editorial team needs to create “automated messages” through the editorial tool. These are based on collections of keywords. The bot itself analyses the question users send on existing keywords. If a match has been found, the bot activates a serie of actions (*which will be explained further in this section*).

A time bot [12] executes a set of actions at a specific time and for a specific selection of users. The user selection will be based on tags where users are associated with. An event bot [13] listens on the event queue. This queue will be activated when the user performs a specific action in the mobile app. As an example a series of actions could be run when the users has logged in for the very first time. A location bot [14] executes actions when a user will be located near a defined geographic location, based on latitude/longitude coordinates. It could also be triggered when a user exits a defined geographic location. A bot can execute one or more actions [15], defined in a specific execution order. Actions that we think are necessary for the test cases we want to do are: sending text, an image, a video, a poll. But also adding or removing tags to the user or media asset that has been sent through the bot.

A connection with external services should also be possible. Examples include executing a specific action on the Kiswe Production Studio platform or sending a request to the World Wide Stream platform to further analyse the data that has been picked up by the bot.

2.5. Kiswe Production Studio platform (KPS)

The current platform consist of 3 layers: the kiswe cloud, the interface and their applications.

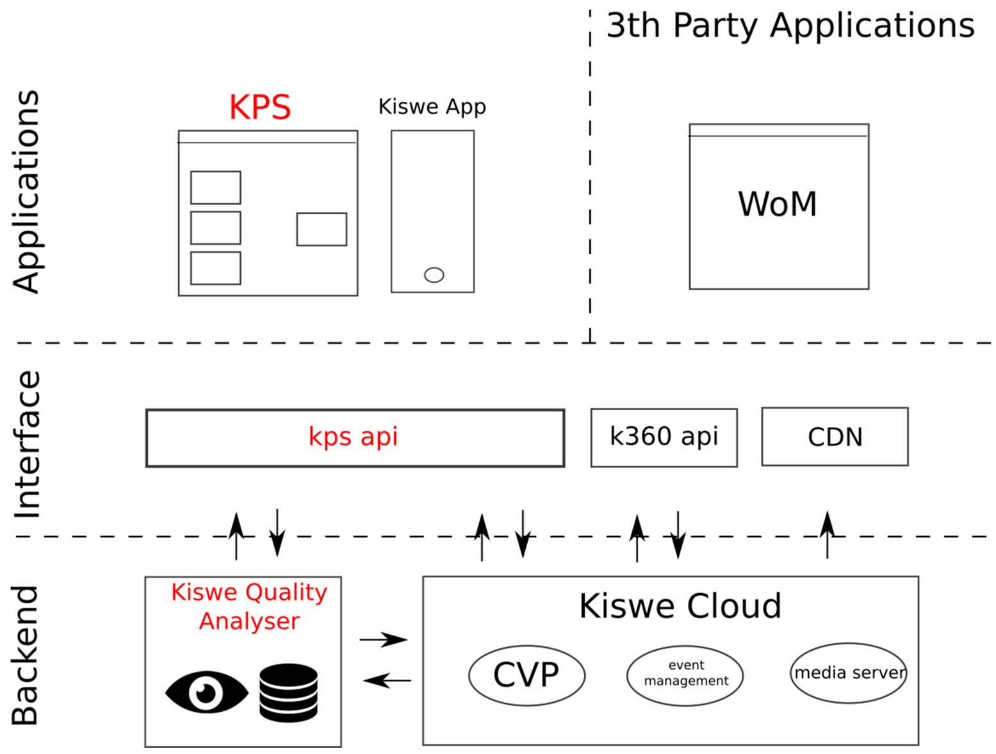


Figure 8: KPS architecture overview.

In the cloud most of the computation is performed. Incoming streams are gathered by the media servers and picked up by the CVP (Cloud Video Processing). This module can combine multiple streams into a single video and convert the outputs to HLS (HTTP Live Streaming protocol). This module has the option to manipulate incoming streams like scaling, repositioning and applying overlays. The generated HLS are stored for each event and pushed through a CDN for (global) distribution to large amount of clients. The event manager manages a database of events and handles the transitions between states such as start, archiving and delete. There is currently 1 public API available as an interface to communicate with the Kiswe cloud, which we call K360 api. This REST API manages events, clipping and chats. Another, private API is present to allow connecting directing tools and applications. This KPS API (Kiswe Production Studio API) allows for replacing the content of each output stream, or to inject overlays on top of a stream. The current Kiswe app is built on top of the K360 API, whereas the KPS tool is mostly using the private KPS API. Any (new) integration with third party applications can be done against each of these 2 APIs, depending on the purpose.

3. Mapping of platform functionality to generic blocks

3.1. Terminology

- Sources: data and video capture systems
- Sinks: apps and screens
- Streams: data and video flows within the platforms
- Connectors: common stream formats between 2 platforms
- Operators: platform processing functionality
- Brokers: stream publish/subscribe functionality
- Registries: databases for streams and operators

"stream" = continuous flow of data or multimedia, "continuous" does not imply that new data occurs at regular intervals, and the limit case of a single data item on a stream can be a useful abstraction to fit on-shot events in the streaming model.

"source" = data or multimedia stream flowing into the platform; can originate from e.g. a sensor, capture device or another platform.

"sink" = data or multimedia stream flowing out of the platform; can terminate at e.g. a screen, rendering device or another platform.

"connector" = common stream format and protocol between two platform instantiations, i.e. a connection between two platforms where one platform acts as a sink, and a second platform as a source.

"operator" = platform functionality that acts upon one or more input and output streams; can be stateless or stateful.

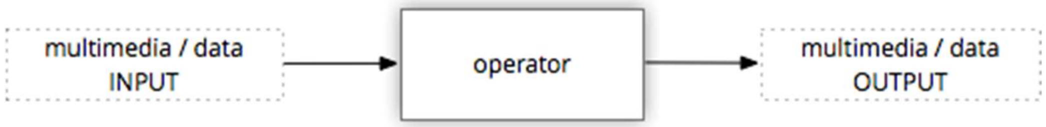


Figure 9: operator.

In MOS2S, operators can be classified in 4 major types. More complex operators can always be decomposed in a combination of operators of these major types (with some typical examples):

- Data to Multimedia (data2MM)
  - meta-data (time-code) insertion
  - VR operators
  - closed captioning operators that embed captions as images
- Multimedia to Multimedia (MM2MM)
  - transcoders
  - resizers

    - denoisers
    - gstreamer pipelines with multimedia output
- Multimedia to Data (MM2data)
    - scene analysis
    - image quality
    - georeferencing
    - object recognition and tracking
    - gstreamer pipelines with non-multimedia output
- Data to Data (data2data)
    - typical CEP operators
    - NLP operators
    - any user defined function

“data flow” = the directed, possibly cyclic, graph consisting of operators as vertices and streams as edges;

‘broker’ = provides stream publish/subscribe functionality. The broker serves as a rendez-vous point between producers and consumers of data or multimedia items, organised in a stream. It decouples the producer from the consumer in a number of ways: protocols used by producer and consumer can differ, push/pull behaviour does not need to match, …

Examples of data broker implementations in MOS2S are RabbitMQ (supporting protocols such as AMQP, MQTT and STOMP)

TODO NOKIA: include STOMP publish and subscribe example code

An HTTPD server with actions associated to incoming HTTP requests can also considered to be an implementation of this broker model: the publisher typically uses HTTP post or put to transfer data into the broker, the consumer is hardwired in the coded actions. This makes the brokers the natural external interfaces for the MOS2S data plane (defined below), both when connecting partner platforms internally and when connecting the MOS2S platform to the external world.

Examples of a multimedia broker are Janus (supporting webRTC), Wowza (supporting RTMP) and a HTTPD server implementing DASH mp4 playout.

TODO NOKIA: include RTMP/RTSP publish and subscribe example code

‘stream registry’ = the database that has an overview of all the data and multimedia streams present in the MOS2S system. Allows some searching functions for subscribers to find back relevant streams. Could be “federated”, in the sense that partner platforms have different implementations that are just made accessible through a wrapper providing a unified API.

‘operator registry’ = the database that has an overview of all the operators working on data or multimedia streams present in the MOS2S system. Can be considered as a representation of the dataflows present in the MOS2S platform. Could be “federated”, in the sense that partner platforms have different implementations that are just made accessible through a wrapper providing a unified API.

‘data plane’ = the ensemble of dataflows present in MOS2S, with a focus on the transport of data and multimedia messages flowing through operators and inside streams, and ignoring the control logic required to make this happen. Integrating different MOS2S platforms SHALL imply we integrate the data planes of the respective platforms.

“control plane” = the ensemble of control logic that is required to instantiate, remove, configure and manage data plane functionality for MOS2S. Integrating different MOS2S platforms MAY imply we integrate control planes of the respective platforms.

“application code” = any code that is not MOS2S platform data plane (protocols, operators) or control plane code that is required to instantiate a particular application.
An example is the VRT chat app from the Wall of Moments platform, that implements very chat-app-specific trigger actions on reception of well-known data coming out of analysis pipelines that have been implemented in MOS2S dataflows. These trigger actions typically include the publishing of messages on MOS2S output streams.

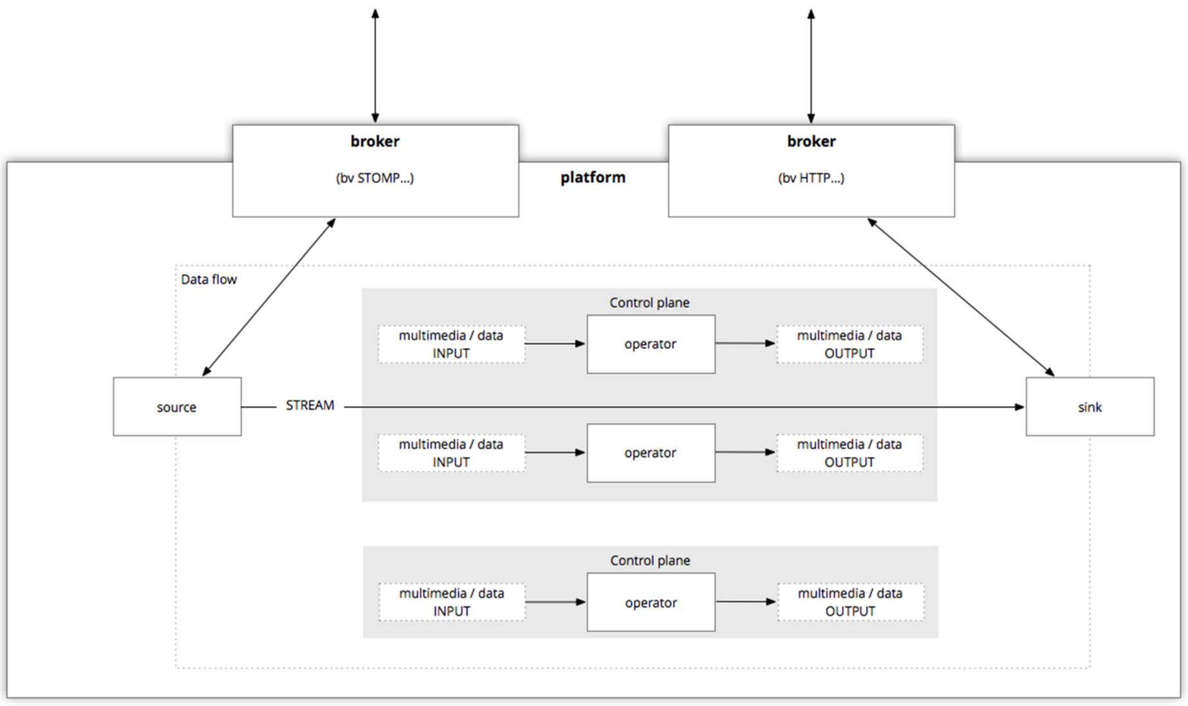


Figure 10: Example of a platform architecture.

2.2. Video orchestration platform

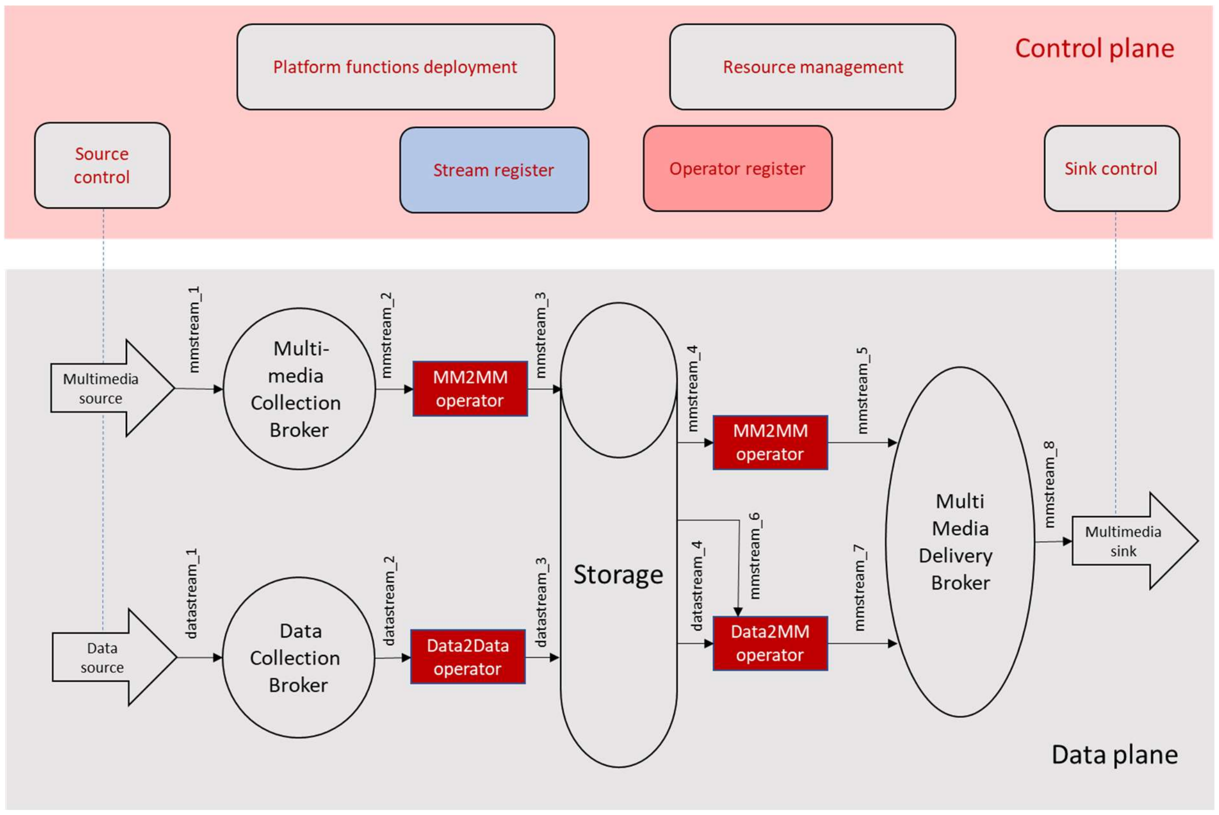


Figure 11: generic reference architecture for video orchestration platform.

VO data plane

In the data plane of the VO platform multimedia data streams are processed with MM2MM, Data2Data and Data2MM operators. On the collection side many different sensors (in most cases cameras) are the sources which provide multimedia streams to the Multimedia Collection Broker (implemented by the Source Side Synchronization function and the Distribution Forwarder). A data source can be any system that provides relevant data related to one or more multimedia streams. The Data Collection Broker will collect data and related timing info required for synchronization of stream, which will be performed by the operators behind the broker. All relevant streams are forwarded by the brokers towards pre-processing functions (acting as the operator on the collection side). The operators will perform synchronization, timestamping, fragmented MP4 packaging and transcoding. The results are stored in the storage function (for which Microsoft Azure, Kubernetes and Minio are used).

On the outgoing or delivery side the stored multimedia streams are processed by another MM2MM operator which provides functions like tiling, synchronization, Mosaic processing, stitching and different types of filtering. This operator will provide its output to the Multimedia Delivery Broker. This broker will provide multimedia stream to all the connected applications and/or CDNs. For fusion of data streams into multimedia content, the Data2MM operator will provide synchronized and integrated multimedia stream to the Multimedia Delivery Broker. The Multimedia Delivery broker also provides orchestration functions which will use the processed multimedia streams from the operators and drive specific filtering functions (spatial stream filter, spatial filter and temporal filter) within the operator. The operators and brokers are implemented using Kubernetes and Apache OpenWhisk, which handle the orchestration of the infrastructure and is capable of automatically scale operators for stream processing. The operators can have an arbitrary number of input and output streams and can share their state information using registries (for which Minio is used). The operators can internally provide parallelism to support scaling of functions. This is for example done for tiling processes, which can be partly parallelized in a cloud to provide sufficient capacity of tiling functionality (see Masters, Tilers (Workers) and Aggregators in VO description).

The VO platform also supports a large storage environment (which is currently migrated from Amazon Webservices to Microsoft Azure). The MM2MM and Data2Data operators can use this storage to store pre-processed (e.g. transcoded) multimedia streams or retrieve available pre-processed multimedia stream for further processing (like tiling, synchronization, MOSAIC processing, stitching and filtering) so streams can be delivered to the sinks.

VO control plane

Source and Sink control

In the VO platform the multimedia sources (the sensors) can communicate with the platform to discover for example where streams van be delivered, in which bitrate should be used and when the stream should start. Functions like the ‘distribution forwarder’ module and the ‘spatial stream filter’ will manage the sources to ensure that the correct input is provided in order to generate the requested output. In the sink control of the Vo platform the receiving devices can provide preferences for video quality. The delivery orchestration functions within the operator will than ensure that the preferences are met.

Orchestration management (platform functions deployment)

An important function of the orchestration management is to orchestrate the data and multimedia capturing processes (using the Data and Multimedia Capturing Brokers) and to orchestrate the distribution of the multimedia content (using the Multimedia Delivery Broker).
The VO platform provides extensive deployment functionality to scale the operator and broker functions locally and in the cloud. The scaling will be based on the registering of streams on the platform.

Resource management

In the resource management component within the control plane consist of sync and filtering functions and a specific hardware management function (which is partly based on OpenWhisk).

All available data and multimedia streams are registered in the stream registery. Information about the collection and delivery operators are located in the Operator register.

3.2. World Wide Stream platform (WWS)

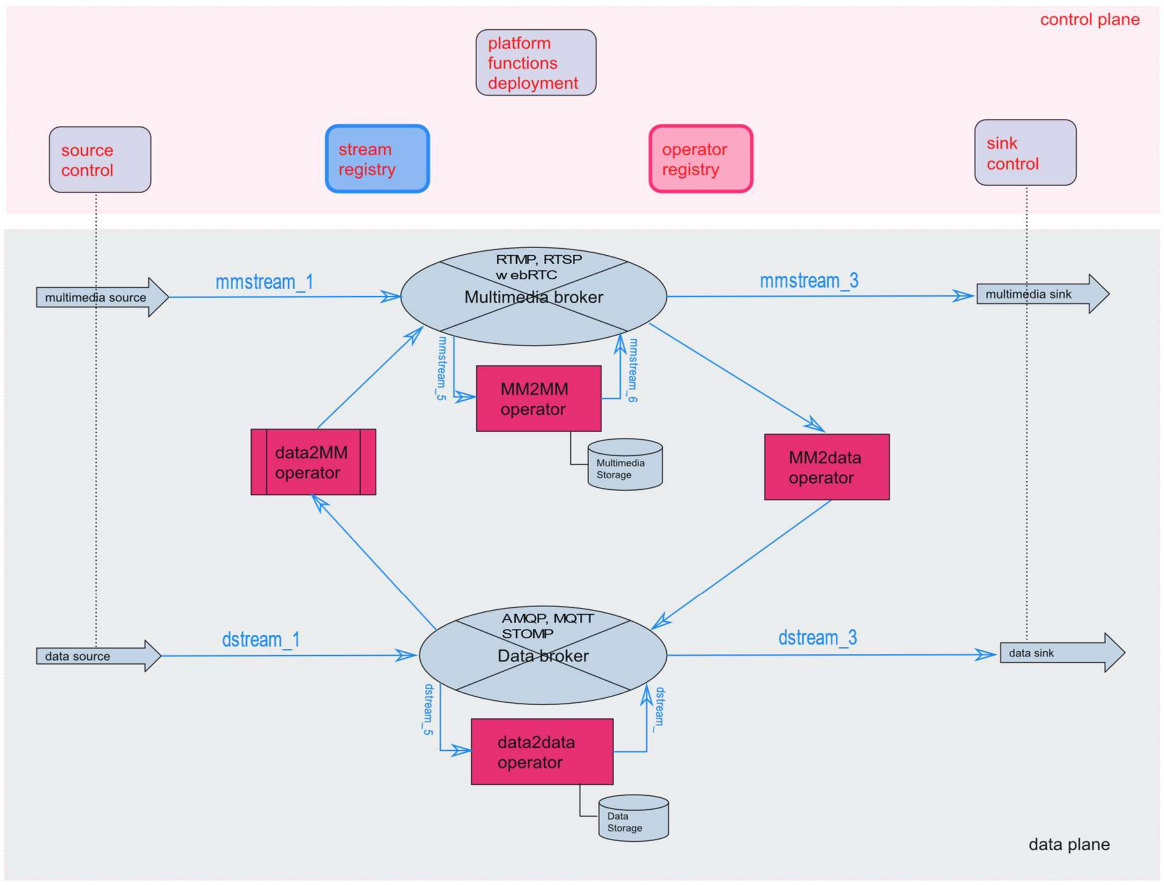


Figure 12: generic reference architecture for WWS platform.

WWS data plane

WWS makes the distinction between multimedia (audio, video) streams and datastreams. WWS as a platform typically does NOT store streams to disk. To the external world, the WWS dataplane is terminated by our stream bridge component (see in section **Error! Reference source not found.**).

Datastreams are handled by the data broker. Currently WWS uses RabbitMQ.

Multimedia streams are handled by the multimedia broker. WWS uses nginx-rtmp for RTMP, JANUS for webRTC.

Streams are processed inside operators, which are deployed on WWS processors. Operators can have an arbitrary number of input and output streams, and can optionally have state. If operators are stateful, this state is localized in the operator. Any shared state between operators would require the use of a shared database infrastructure (WWS supports redis, amongst others).

Although WWS has no platform support for multimedia or data storage, individual operators can use attached storage (database, filesystem) available to the containing processors.

WWS supports all 4 operator classes defined by MOS2S: data2data, data2MM, MM2data and MM2MM. Obviously, from these simple operator classes it is possible to create more complex operators (by subgraph composition) that have data and MM both at input and output ports.
The brokers in WWS allow re-use of datastreams: if a graph is deployed that shares inputs and possibly derived streams (due to an identical subgraph connected to these inputs), the broker will make the single derived result available to the new stream.
WWS is a distributed system: the operators can be deployed on a range of available processors that can go all the way from sensor gateways over edge compute to core compute facilities. To support this, the brokers are also deployed in a distributed way and automatically interconnected by the platform itself.

WWS control plane

Source and Sink control

As explained in **Error! Reference source not found.**, WWS has the notion of a controller stream that can be used to manage (for now: start and stop) attached sources and sinks.

Dataflow management (platform functions deployment)

As specified in **Error! Reference source not found.**, dataflows can be deployed and removed through command line tools or a REST API. Section **Error! Reference source not found.**, explains xstream, our typescript DSL used to define dataflows.
The stream registry stores all information (including metadata and the broker endpoints) regarding the streams deployed in WWS.
The operator registry stores all information regarding the operators deployed in WWS (which is essentially a database representation of the dataflow graphs).
WWS has deployer (geographically large scale) and dispatcher (within datacenter) functionality that helps to decide on which processor operators get deployed.

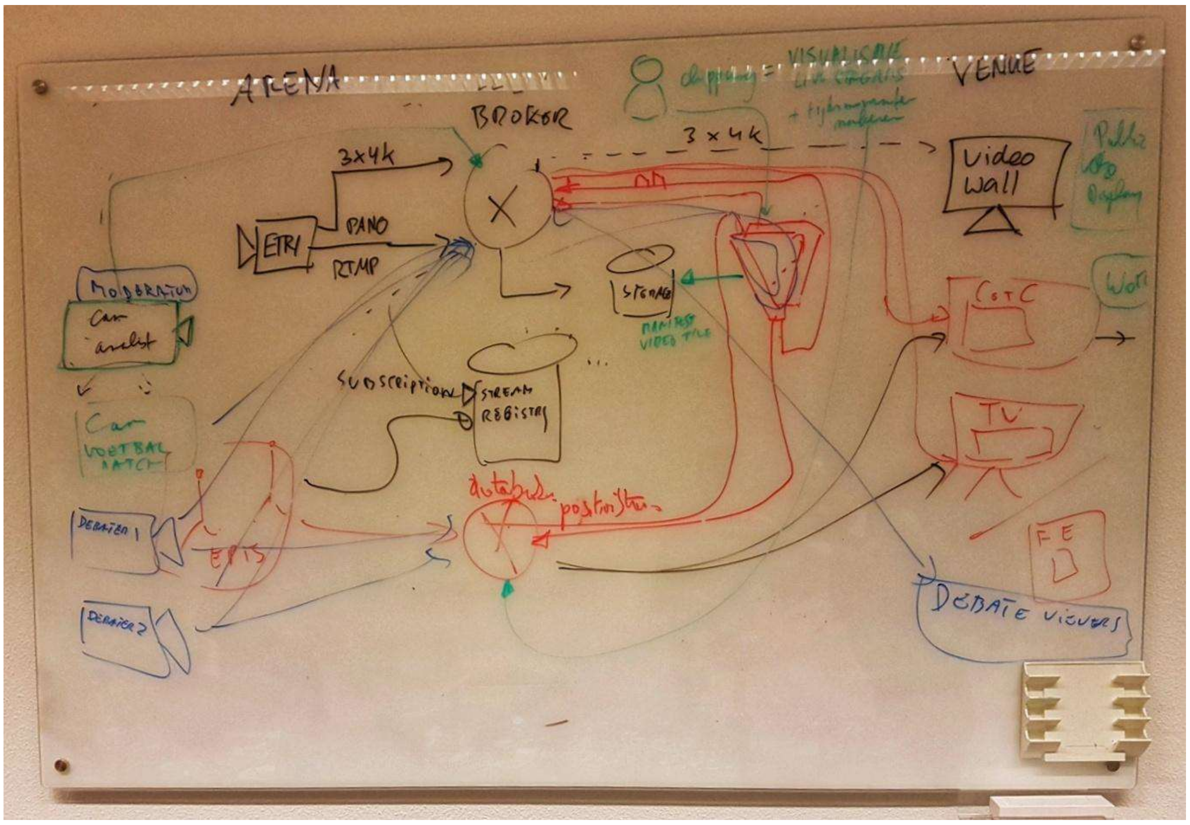
Resource management

WWS has elasticity support to manage the amount of running processors and monitors load and latency on processors and brokers.

3.3. Wall of Moments platform

TODO

4. Mapping of platform functionality to Y2 demo scenarios



Scenario 1: (zwart & rood) - Ultra Wide View

- Source: ETRI UWV capture platform (transport protocol: RTMP)
  - 3x 4K streams
  - stitched panorama, not georeferenced
- Source: Player positioning data (Inmotio) [TODO Inmotio platform interface]
- optional Source: GameOn capturing platform streams
  - georeferenced (TODO check) panorama [TODO Gameon output interface]
- Multimedia broker
- Data broker
- Sinks:
  - Rendering engine voor ETRI video wall setup
  - Cotc app
  - TV/mobile device playout of platform generated content
  - Turkey debate app
  - Storage (enabling replay)
- Data flows:
  - Downscaling of 4K streams for tablet app
  - georeferencing of ETRI UWV panorama or 4K streams
    - possibly based on GameOn algorithm [TODO Gameon ?input?/output interface]
  - Cropping of specific region based on player position data + derived stream generation for TV and Venue app *(TODO WHAT venue app?)*

- Innovation:
  - new stream generation based on available data+MM sources
  - showcase of data+video based on panoramic video
- Sources:
  - (MM) ETRI UWV capture platform, UWV content (transport protocol: RTMP
    - 3x 4K streams
    - single stitched panorama, not georeferenced
  - (MM) GameOn server
    - 4x Bosch 4k camera's (twee achter de goal, twee op de middellijn)
    - 2x FullHd Bosch PTZ camera's
    - 1x 24MP Hikvision camera, bestaande uit 3 streams van elk 8 MP
    - live KPN TV stream (deze halen we via UDP multicast binnen)
  - (Data) Inmotio player positioning data [TODO Inmotio data interface]
- Sinks:
  - (Public screen) ETRI - stitching and rendering engine for video wall setup
  - (2nd screen) GameOn - coach-on-the-coach tablet app
  - (TV) STB playout of live TV and platform generated content
- Storage (enabling replay)
  Processing:
  Downscaling of 4K streams for tablet app,
  georeferencing of UWV panorama or 4K streams, based on GameOn algorithm? [TODO Gameon ?input?/output interface]
  Cropping of specific region based on player position data + derived stream generation for TV and Venue app (TODO WHAT venue app?)
  Rendering of augmented views, i.e. graphical overlays on top of UWV-based video

Scenario 2: (groen) - Clipping van realtime streams
- Sources via Multimedia broker
  - Camera op de analist (eventueel Erik Meijer [voorstel van Cyril]) + moderator
  - Coach on the Couch (het analyse-verhaal)
  - User generated content
- Sources via Data broker:
  - Tijdsmomenten op de live videostreams (= analist + coach on the couch output) markeren via een human operator
- Applicatie/dataflow: clipping interface
  - Ingevoerde tijdsmomenten (gestuurd via de databroker) worden gebruikt om automatische clips te genereren. Deze clips worden als afzonderlijke video assets op de file storage bewaard
- Sink via multimedia broker
  - Publishen van de offline video assets (on demand)

Scenario 3: (blauw) - Output kanalen worden sources voor andere applicaties
- Sources via multimedia broker
  - Camera op de analist + moderator
  - Camera's van de debaters
  - Optional other sources:
    - Coach on the Couch (het analyse-verhaal)
    - Professional broadcast video stream
    - Video's from people's mobile phones at the stadium
- Dataflow:
  - Spatiële / temporele video stream (*tbd*)

5. Concluding Remarks

This Deliverable D2.1.1 provides the reference architecture platform, component and interface requirements for the 2nd year milestone. The document outlines the various platforms that the MOS2S partners bring to the project, the component functionality that is being developed and the interfaces that are required for interoperability.
In the period towards the demonstrator (Q3 2018), the MOS2S consortium focus on platform integration to enable a first integrated showcase.