Grant Agreement No.: 731677
Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

D3.7: FLAME Technology Roadmap V2

Dirk Trossen, Sebastian Robitzsch, Kay Hänsge (InterDigital Europe)

Michael Boniface (IT Innovation)

Carlos Alberto Martin Edo (ATOS)

Aloizio Pereira da Silva (UNIVBRIS)

15 December 2018

This report is the second technology roadmap for a ground-breaking media service delivery platform being developed by the FLAME project. The report describes the software products to be delivered at infrastructure, platform and media service layers and how combinations of products are used to exploit the benefits of highly distributed software-defined infrastructures. Each product is described in terms of features, baseline implementation technologies and release schedule. At the core of the roadmap is the FLAME platform that brings together components for orchestration, Service Function Routing, Service Function endpoint management and cross-layer management and control. A systems integration and testing plan describes the DevOps environment including multi-project structure, development workflows and continuous integration processes supported by build, provisioning, configuration and automated testing tools. A software integration infrastructure is designed that replicates a part of the production infrastructures in ways that allow flexible configuration of different cross-component test scenarios. Finally, the downstream staging and production infrastructures are summarised completing the end-to-end DevOps pipeline for efficient and high-quality delivery.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Work package	WP 3
Task	Task 3.5
Due date	30/10/2018
Submission date	11/11/2019
Deliverable lead	InterDigital
Version	1.0
Authors	Dirk Trossen (InterDigital), Sebastian Robitzsch (InterDigital), Kay Haensge (InterDigital), Michael Boniface (IT Innovation), Carlos Alberto Martin Edo (Atos), Aloizio Pereira da Silva (UNIVBRIS)
Reviewers	Gino Carrozzo (NXW), Steven Poulakos (Disney)
Keywords	Media Services, Software-defined infrastructures, technical roadmap, systems integration, DevOps

Disclaimer

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731677.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to FLAME project and Commission Services	

EXECUTIVE SUMMARY

This report is deliverable D3.7 FLAME Technology Roadmap V2 of the FLAME project. The document describes the update of the roadmap for development, integration and production deployment of the ground-breaking FLAME media service delivery platform.

The roadmap aims to deliver software products deployed as operational services on real-life software-defined infrastructures for trials and experimentation. The primary purpose of the trials is to validate the FLAME offering by delivering performance and cost benefits to media service providers and enhanced quality of experience to end users.

A systems integration and testing plan describes the DevOps environment including multi-project structure, development workflows and continuous integration processes supported by build, provisioning, configuration and automated testing tools. A software integration infrastructure is designed that replicates part of the production infrastructures in ways that allow flexible configuration of different cross-component test scenarios. Finally, the downstream staging and production infrastructures are summarised completing the end-to-end DevOps pipeline for efficient and high-quality delivery. The overall roadmap is designed to ensure alignment of activities across all work packages in the project from component development, integration through to trials and experimentation.

This report updates the previously delivered report D3.5 by providing the feature plans for the individual components of the FLAME platform based on the insights of the progressing development and integration work since finalizing D3.5. Furthermore, the deliverable also provides an update to the integration environments, now covering the chain from an early 'sandbucket' environment of the sandpit used for staging and early verification of experiments down to the replication infrastructures in Bristol and Barcelona.

Co-funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

AR	Augmented Reality
AVC	Advanced Video Coding
CDN	Content Delivery Network
CI	Continuous Integration
CLMC	Cross-Layer Management and Control
CMS	Content Management System
DC	Data Centre
ETSI	European Telecommunications Standards Institute
FQDN	Fully Qualified Domain Name
FMI	Future Media Internet
HTTP	Hyper Text Transfer Protocol.
IP	Internet Protocol
KPI	Key Performance Indicator
MANO	Management and Orchestration
NFV	Network Function Virtualisation
PCE	Path Computational Element
QoS	Quality of Service
RC	Release Candidate
SDN	Software Defined Network
SF	Service Function
SFC	Service Function Chain
SFE	Service Function Endpoint
SFEMC	SF Endpoint Management and Control
SFR	Service Function Routing
SR	Service Router
TLS	Transport Layer Security

Co-funded by the Horizon 2020
Framework Programme of the European Union

TOSCA	Topology and Orchestration Specification for Cloud Applications
TRL	Technology Readiness Level
UE	User Equipment
VM	Virtual Machine
VIM	Virtual Infrastructure Manager

1 INTRODUCTION

1.1 PURPOSE

This document describes the technical roadmap for implementation, integration, testing and deployment of FLAME technologies supporting trials and experiments of media services on highly-distributed software-defined infrastructures. The goal is to provide software development teams, responsible for FLAME software products, and operations teams, responsible for service deployment, with the feature release schedules and DevOps processes that ensure timely delivery and results to an acceptable level of quality.

1.2 SCOPE

The project is structured into three iterative development phases aligned with the strategic activities of the work:

- *Jan-17 to Oct-18: Research and innovation foundations*: design, implement and deploy the Alpha release of the FLAME platform ready for trials in Bristol and Barcelona production infrastructures.

- *Nov-18 to Dec-19: Ecosystem building and disruptive experimentation:* operate trials and experiments to validate the platform, working on feature enhancements towards the Beta release.

- *Jan-20 to Jun-20: Sustainability*: transition towards exploitation and sustainability, hardening the platform for RC release and working closely with technology adoption partners

The high-level platform engineering cycle follows these project phases. The project is currently in the "ecosystem building and disruptive experimentation" phase with the current deliverable forming part of a series of public reports delivered in each phase (see Figure 1).

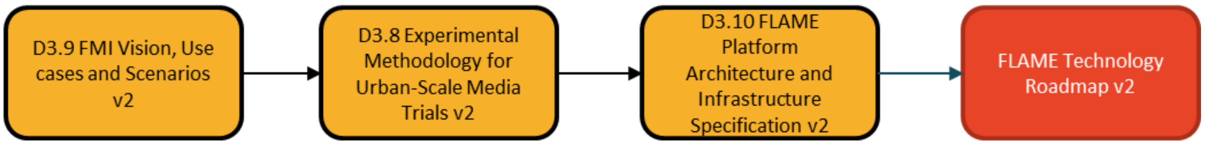

D3.9 FMI Vision, Use cases and Scenarios v2
D3.8 Experimental Methodology for Urban-Scale Media Trials v2
D3.10 FLAME Platform Architecture and Infrastructure Specification v2
FLAME Technology Roadmap v2

Figure 1: FLAME platform engineering reports

This report is the second version of the roadmap with final updates planned to be delivered at the end of the project. Related reports include:

- D3.9 [FLAME-D3.9] describes an series of updated scenarios and use cases for interactive media using the platform

- D3.8 [FLAME-D3.8] describes an updated methodology for conducting urban scale trials that explore the cross-layer and multi-stakeholder interactions within the systems-under-test.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- D3.10 [FLAME-D3.10] describes the updated architecture and infrastructure specifications for the FLAME platform, elaborating the use cases from D3.9, refining system requirements and identifying platform components and interfaces.

D3.3 is the primary reference point for the initial technical roadmap because it provides the overall structure of the platform and allows features and development tasks to be decomposed into areas of work. D3.10 will provide an update to D3.3, as shown in Figure 1, which serves as a reference point for this updated technology roadmap.

The target audience for this deliverable are developers working on FLAME software products, infrastructure owners responsible for production deployment and wider stakeholders interested in the FLAME offering, features and expected release schedules.

1.3 DELIVERY PARTNERS

The project is delivered by members of the FLAME consortium who have specific responsibilities for implementation, operations, engagement and marketing of FLAME. The partners are referred to by acronyms throughout this report as shown in Table 1.

Participant organisation name	Short Name	Country	Roadmap Leadership Roles
IT Innovation Centre	ITINNOV	UK	Platform, CLMC
Atos Spain SA	ATOS	Spain	Media Services
InterDigital Europe Ltd	IDE	UK	SFR, SFEMC
Fundacio Privada i2CAT, Internet I Innovacio Digital a Catalunya	i2CAT	Spain	Barcelona Infrastructure Operator
University of Bristol	UNIVBRIS	UK	Bristol Infrastructure Strategy & Operator
Nextworks	NXW	Italy	Validation Experiment
Martel GmbH	Martel	Switzerland	Media Services
De Vlaamse Radio En Televisieomroeporganisatie NV	VRT	Belgium	Validation Experiment
The Walt Disney Company (Switzerland) GmBH	DRZ	Switzerland	Validation Experiment
Eidgenoessische Technische Hockschule Zuerich	ETH	Switzerland	Validation Experiment
Institut Municipal d’Informàtica de Barcelona	IMI	Spain	Barcelona Infrastructure Strategy

Table 1: FLAME consortium partners

1.4 UPDATES TO PREVIOUS ROADMAP DELIVERABLE D3.5

This deliverable revises the initial roadmap provided in D3.5 with updated feature plans (Section 3 & 4), using the insights of the initial development and integration efforts. It also introduces a revised integration environment (Section 5), now providing a localized sandbucket environment as well as the sandpit environment for early functional testing of media services together with the latest updates for the available replication infrastructures in Bristol and Barcelona.

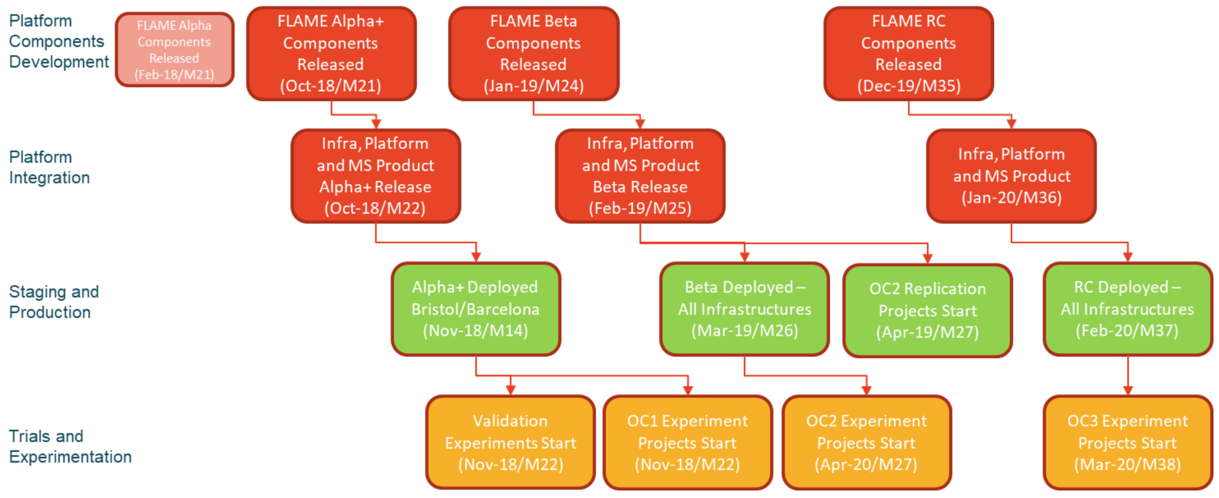
Co-funded by the Horizon 2020
Framework Programme of the European Union

2 TECHNOLOGY ROADMAP OVERVIEW

This section provides an overview of the technology roadmap and the software products provided by FLAME with updates to release dates compared to the initial roadmap provided in D3.5.

2.1 PROJECT MILESTONES

FLAME plans three major releases of the FLAME offering within the lifetime of the project. The timing of major releases is aligned with the timescales of trials. Each major release will include significant feature enhancements within the overall offering across infrastructure, platform and media services. A release at the project level indicates the launch of a "FLAME Service" for trials in contrast to the release of specific software products that the FLAME Service depends on.



Platform Components Development
FLAME Alpha Components Released (Feb-18/M21)
FLAME Alpha+ Components Released (Oct-18/M21)
FLAME Beta Components Released (Jan-19/M24)
FLAME RC Components Released (Dec-19/M35)
Platform Integration
Infra, Platform and MS Product Alpha+ Release (Oct-18/M22)
Infra, Platform and MS Product Beta Release (Feb-19/M25)
Infra, Platform and MS Product (Jan-20/M36)
Staging and Production
Alpha+ Deployed Bristol/Barcelona (Nov-18/M14)
Beta Deployed – All Infrastructures (Mar-19/M26)
OC2 Replication Projects Start (Apr-19/M27)
RC Deployed – All Infrastructures (Feb-20/M37)
Trials and Experimentation
Validation Experiments Start (Nov-18/M22)
OC1 Experiment Projects Start (Nov-18/M22)
OC2 Experiment Projects Start (Apr-20/M27)
OC3 Experiment Projects Start (Mar-20/M38)

Figure 2: Platform releases in relation to project milestones

After the initial releases in Feb-18 (for the initially planned alpha release for internal testing), updated releases are planned for Oct-18, Jan-19 and Dec-19 with the working names of Alpha, Alpha+, Beta and Release Candidate (see Figure 2), respectively. The major releases correspond to milestones for FLAME feature implementation. The project implements DevOps processes that offer greater agility in the implementation of release of features. As such Minor releases will be delivered in between major milestones to incorporate new features when they are available and hot bug fixes when they are critical to service operations.

2.2 OVERVIEW OF SOFTWARE PRODUCTS

FLAME delivers three types of software products that reflect the layering in the architecture, as shown in Figure 3 and described in Table 2.

Software Product Type	**Description**
Platform Product	A product offering flexible management and delivery of media services deployed on replication infrastructures. Those infrastructure are provided by infrastructure providers as a so-called Infrastructure Product. One platform product is expected to be delivered. This product will be able to be configured for different replication infrastructure. The platform product is the primary outcome of the FLAME project.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Software Product Type	Description
Media Service Product	A product offering content production, management and/or distribution features that directly benefit from the features of the Platform Product. Many Media Service Products are expected to be offered. The selection is based on the Media Service products that benefit most from Platform Product features and are in demand for delivery of new forms of user experience and social interaction.
Integration Environment	An environment offering access to and management of infrastructure resources based on specific hardware configurations. The infrastructure abstraction offered must be common across all Infrastructure Products although it is expected there will be some variation in function and performance for different Infrastructure Products. Multiple integration environments are expected covering different integration, staging and production environments. It is expected that those environments will build on and adapt widely used open source available solutions, e.g. OpenStack[1], OpenDaylight[2] and Floodlight[3] and where necessary contributions will be made to open source extensions of existing infrastructure products

Table 2: FLAME software products

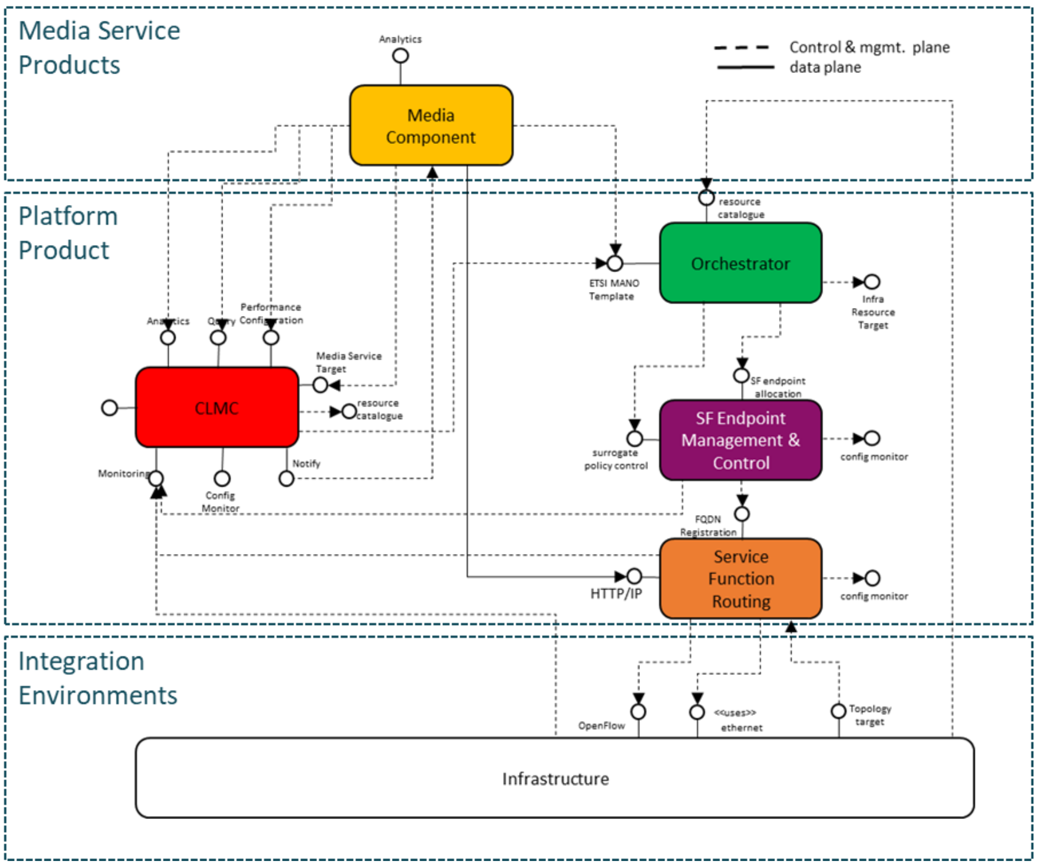


Figure 3: FLAME software products in relation to architecture

Figure 4 and Figure 5 provide an overall summary of the approach. Software products are developed within a dedicated continuous integration pipeline provided by the FLAME project. The integration

[1] https://www.openstack.org/

[2] https://www.opendaylight.org/

[3] http://www.projectfloodlight.org/floodlight/

Co-funded by the Horizon 2020 Framework Programme of the European Union

environment of a **sandbucket** provides a virtual machine based environment that can be used by media service providers to test key aspects of the orchestration and packaging of their media service. This integration environment is also used for platform integration and testing before moving to the **sandpit** environment. The sandpit is realized in dedicated server hardware, allowing for functional system-level testing of both platform and media service products. For performance-related integration and testing, the **staging integration** environment is being used, emulating the actual production environment as closely as possible and putting emphasis on the performance aspects. Finally, the testing and integration moves to the **production** (or experimentation) **infrastructure** with the previous phases aiming to minimize said testing in order to maximize the utilization of the production infrastructure for the actual service delivery.

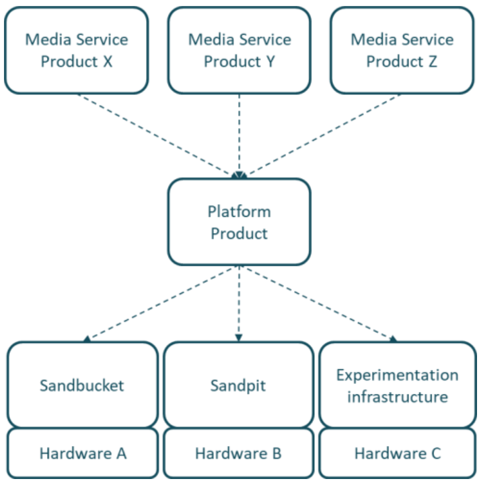


Figure 4: High level product dependencies

Any changes to software products in the pipeline may trigger continuous integration tests for products downstream in the pipeline that depend on the product. The level of automation in the continuous integration process across products and the scheduling of integration tests at different phases in the pipelines depends on the level of human control desired and the cost of integration testing. The final stage of deployment on the production infrastructure can also form part of the continuous deployment processes, however, this depends on the policy of infrastructure operators.

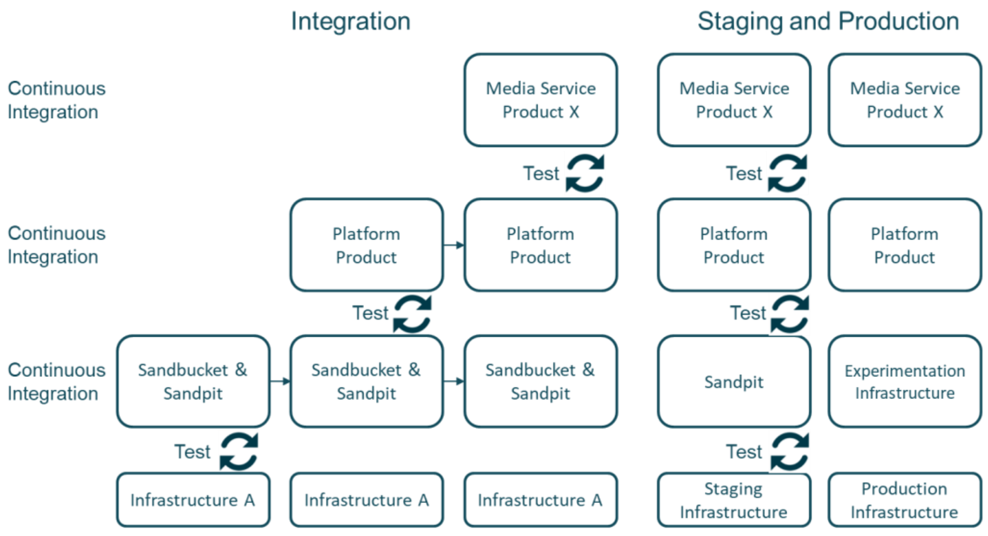


Figure 5: Overview of software product integration and release

Co-funded by the Horizon 2020 Framework Programme of the European Union

FLAME products are implemented, integrated and tested through contributions from multiple organisations. Table 3 shows the distribution of responsibilities for technical partners contributing to the implementation of the Platform product and Media Service products. Each component has an owner responsible for delivery of the components to integration based on contributions from other organisations.

Software Product	**Project Owner**	**Responsibility**
Integration Environments	University of Bristol	• Integration Infrastructure (ITINNOV) • Bristol infrastructure (UOB) • Barcelona infrastructure (i2CAT)
Platform	IDE	• Orchestration (IDE) ○ FLAME enhanced TOSCA specification language (IDE) ○ FLAME orchestrator (IDE) ○ Platform orchestrator (IDE) ○ Media service orchestrator (IDE) • SF Endpoint Management and Control (IDE) • SF Routing (IDE) • Cross Layer Management and Control (ITINNOV)
Media Service	Atos	• Media service selection, adaptation and packaging (Atos) • Media service packaging (Martel) • Media service monitoring (Atos)

Table 3: Partner responsibilities across product implementation, integration and deployment activities

Co-funded by the Horizon 2020
Framework Programme of the European Union

3 PLATFORM PRODUCT ROADMAP

A Platform Product offers flexible management and delivery of media services deployed on Infrastructure Products described in Section 5. The Platform Product is the major software outcome of FLAME providing advanced service management through Orchestration, Service Function Endpoint Management and Control, Service Function Routing and Cross Layer Management and Control. The overall benefits of the Platform are delivered through an aggregation of component features.

The following section provides an overview of the features as being implemented already in the Alpha release and planned for the upcoming releases Alpha+ and beyond. New requirements have not emerged during the initial implementation and integration work, confirming those presented initially in the FLAME architecture specification D3.4. However, feature release dates have been adjusted in some components, largely motivated by earlier realization in the initial Alpha release or by moving them to later releases due to a recognized lower importance in initial validation and open call experiments.

3.1.1 Platform Components and Features

This section describes the feature roadmap, service function chain, implementation technologies, ownership of components that will form part of the Platform Product. Each service function chain is analysed to determine the background technologies and the expected enhancements and adaptations needed to deliver the features. The Technology Readiness Level[4] is provided to give an indication of the level of work that needs to be completed to ensure the component is ready for integration into the Platform Product.

The ownership and licensing situation for components is identified including 3rd party licenses to identify restrictions on access to the Platform Product. The Platform Product will be distributed as software for deployment on production infrastructures by infrastructure providers initially (e.g. Smart Internet Lab - University of Bristol and i2CAT) and then 3rd parties. The Platform Product will also be made available for evaluation by 3rd parties for evaluation and trials. If restrictions are identified then design and implementation decisions will be needed to isolate such components or seek alternative implementations that are consistent with the usage objectives.

3.1.1.1 Orchestrator

The Orchestrator component supports the interaction with Cross-Layer Management and Control (CLMC) and media components, leading to an orchestration of compute, storage, and communication resources, including the suitable configuration for SF endpoint control policies.

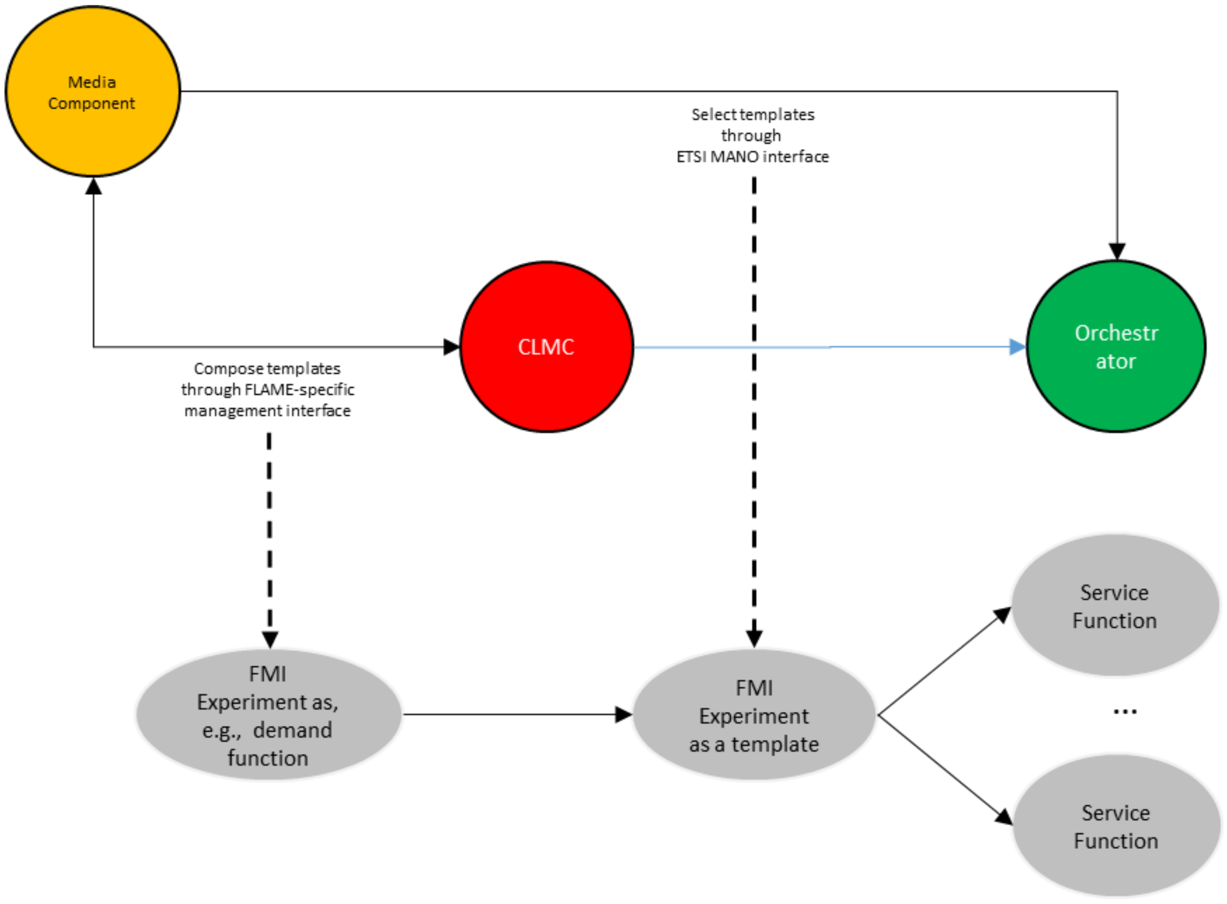
The features of the orchestrator are defined in Table 4. These features are organised in accordance with the interfaces towards other system components including:

- ETSI NFV MANO APIs, used to receive and parse a suitable TOSCA template that outlines the required resources to be orchestrated
- Resource APIs: Used to receive a suitable TOSCA-based infrastructure resource catalogue that can be used to match against orchestration requests

[4] https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Orchestration APIs: Used to support the various orchestration frameworks and platforms being utilised for FLAME, specifically those at the infrastructure, platform and media services level. Figure 6 shows these levels of orchestration being realised through this feature.



Media Component
Select templates through ETSI MANO interface
CLMC
Orchestr ator
Compose templates through FLAME-specific management interface
Service Function
...
Service Function
FMI Experiment as, e.g., demand function
FMI Experiment as a template

Figure 6: Orchestration Workflow through Media Services or CLMC

Feature ID	Req	Feature Description	Component Interface	Release
ETSI MANO				
ORCH-1	Req-O1	Provide TOSCA template to FLAME platform orchestrator	ETSI MANO	Alpha
ORCH-2	Req-O1 Req-O2 Req-O3 Req-I1	Parse FLAME-TOSCA, as defined in T4.1, template and check for consistency	ETSI MANO	Alpha+
Resource				
~~ORCH-3~~	Req-I1	Receive TOSCA template as infrastructure catalogue information	Resource	dropped since media resources are now deployed in clusters
~~ORCH-4~~	Req-O2 Req-O3	Provide topology information towards SF routing component	Resource	now directly queried by SFR component
Orchestration				
ORCH-6	Req-O1 Req-O2	Support container (lxc) based media service orchestration	ETSI MANO	Alpha+

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
	Req-O3			
ORCH-7	Req-O1 Req-O2 Req-O3	Full consistency checks of FLAME-TOSCA template, including consolidating deployment state with orchestration request	ETSI MANO	Alpha+
ORCH-8	Req-O2 Req-O3 Req-SEM1 Req-SEM3	Provide SF endpoint control policies in TOSCA template extensions towards SFEMC component	ETSI MANO	Alpha
ORCH-9	Req-O2 Req-O3 Req-SEM1 Req-SEM3	Provide SF endpoint state information in TOSCA template extensions towards SFEMC component	ETSI MANO	Alpha+

Table 4: Orchestrator features

The delivery is expected to be organised around key interface features. The SF Endpoint Management and Control component is responsible for changing resource configurations in response to demands expressed in the orchestration process by TOSCA [ETSINFV] templates being provided to the orchestration component. Said TOSCA templates, which will be based on existing specifications for the alpha release while envisioned to be extended for FLAME-specific requirements (e.g., to support geo-location constraints) in the alpha+ and beta, are referred to as FLAME-TOSCA in our feature table. The orchestration feature will initially merely separate the management from the control parts in the extended TOSCA template. For this, the template is provided to the orchestrator component for the initial placement as well as the SFEMC component for the control of the SFE. In turn, the necessary information is provided to the SF Endpoint Management and Control component for the initialisation of the SF endpoint state. The orchestration feature provides the suitable control policies to the SF Endpoint Management and Control component, while the resource feature provides the suitable topology information to the SF Routing component.

The relevant SFC for the alpha+ release is shown in Figure 7. The main interactions of the orchestration component are illustrated there, i.e., first, the distribution of information derived from the received TOSCA template to sub-components of the SF Endpoint management and control, second, as the SF Routing components, specifically for the SF endpoint control policies, and finally the topology information, obtained through the infrastructure provided resource information.

The Orchestrator component has been implemented using existing TOSCA parser and validation software, adapted to the workflow of the orchestration and all features in Table 4 will be delivered as part of the alpha+ release. The key for the alpha+ phase is to interface with the orchestration platforms developed at the infrastructure level in Bristol and Barcelona, which is based on OpenStack[5]. We will align the technology platform used in Bristol for the platform orchestrator, while initially using the same platform for media service orchestration. Furthermore, we offer also (linux-)container-based platforms for media services.

[5] http://www.openstack.org

Co-funded by the Horizon 2020
Framework Programme of the European Union

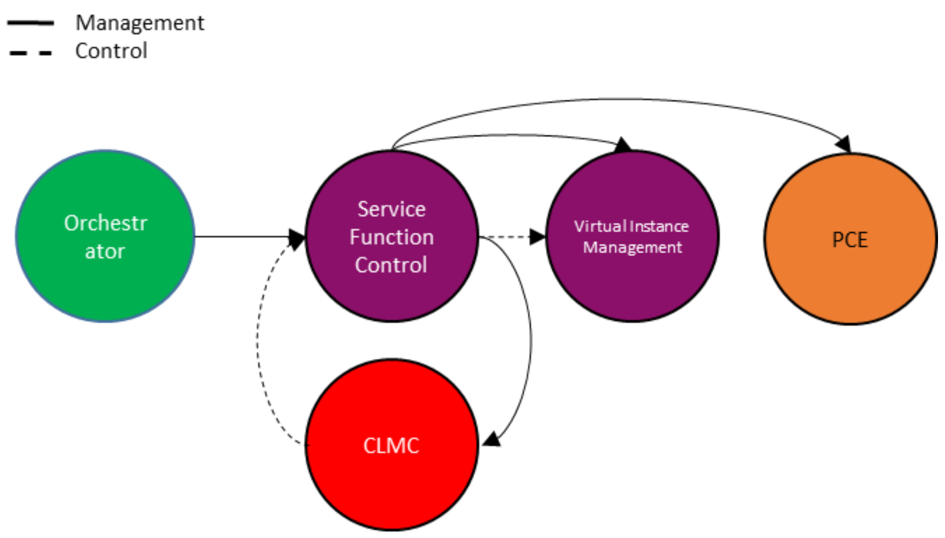


Figure 7: Relevant Orchestration Service Function Chain for Alpha+ Release

Table 5 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
Orchestration	Open Source MANO	ASLv2	Parsing of TOSCA extensions to include control policies	IDE	6

Table 5: Orchestrator implementation technology summary

3.1.1.2 SF Endpoint Management and Control

The SF Endpoint Management and Control component supports the orchestration process by adding the flexible control capabilities outlined initially in D3.3 "FLAME Platform Architecture and Infrastructure Specification V1" and updated in D3.10 by maintaining SF Endpoint instance state in collaboration with the SF Routing component.

The features of the SFEMC are defined in Table 6. These features are organised in accordance with the interfaces towards other system components including:

- Surrogate Policy Control: Used to receive and parse a suitable control policy from the orchestration component, querying required monitoring data pertaining to such control policy and realising a decision logic that matches the monitored data against the policy provided. This feature mainly resides at the Service Function Control component.

- SF Endpoint Allocation: Used to initialise and maintain an SF Endpoint specific state as well as the compute/storage images that define the SF Endpoint functionality, while also realising delegated name authorisation for the SF Endpoint. This feature mainly resides at the Virtual Instance Manager component.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
Surrogate Policy Control				
SFEMC-1	Req-SEM1	Parse surrogate policy based on TOSCA template extension	Surrogate Policy Control	Alpha
SFEMC-2	Req-SEM1	Parse surrogate policy based on FLAME-TOSCA template extension	Surrogate Policy Control	Alpha+
SFEMC-3	Req-SEM1 Req-SEM4	Query monitoring data	Surrogate Policy Control	Alpha
SFEMC-4	Req-SEM1 Req-SEM4	Offering an event interface to match policy triggers and perform defined actions	Surrogate Policy Control	Alpha+
SE Endpoint Allocation				
SFEMC-5	Req-SEM4	Initialise and maintain SF endpoint state	SF Endpoint Allocation	Alpha
SFEMC-6	Req-SEM4	Maintain SF endpoint compute/storage images	SF Endpoint Allocation	Alpha
SFEMC-7	Req-SEM2	Allow for delegated name registration for SF endpoint images	SF Endpoint Allocation	RC

Table 6: SF Endpoint Management & Control Features

The SFEMC component delivers features to the SF Routing component as part of the overall orchestration process in general and the control process in particular. The delivery is expected to be organised around key interface features. The SF Endpoint Management and Control component is responsible for changing resourcing configurations in response to demands expressed in the orchestration process by having received the suitable control policies from the Orchestrator component. The surrogate policy control feature will establish suitable monitoring capabilities aligned with the surrogate policy constraints defined. It will also realise the decision logic to match the monitored data against said policy constraints. The SF endpoint allocation feature maintains the SF endpoint state according to the control policy provided while utilizing the SF Routing component for service routing related state changes of the SF endpoint.

The relevant SFC for the alpha release is shown in Figure 7 with the surrogate manager SF representing the surrogate policy control and SF Endpoint Allocation features of the SFEMC in Table 6. The VIM SF represents the functionality being used by available virtual instance platforms, such as KVM or LXC, in our realisation. As can be seen, we foresee the interaction between Orchestrator component and SFEMC to be realised between our extended functionality (i.e., the orchestration and the SFEMC features of Table 6), while realising the initialisation and maintenance of the SF endpoint state through suitably interfacing with existing VIM solutions, particularly KVM and LXC on Linux. While an ultimate deployment of the SFEMC would foresee a single VM for this purpose, it is likely to utilise several VMs for the FLAME-specific extensions and the re-used VIM parts.

As indicated in Table 7, all features of the SFEMC component will be available for the alpha+ release, against which the roadmap in this deliverable is written against, apart from the delegated name registration, which will be integrated in the RC release.

Table 7 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
Surrogate Manager	FLIPS	Access via Consortium Agreement	Realization of features according to feature table	IDE	6
VIM	OpenStack	ASLv2	Integration	IDE	6

Table 7: SFEMC implementation technology summary

3.1.1.3 Service Function Routing (SFR)

The SF routing component realises the service request routing at the data plane between media components, including all operational and management features for supporting route changes, registration of SF endpoints, etc.

The features of the SFR are defined in Table 8. These features are organised in accordance with the interfaces towards other system components including:

- *Protocol mapping*: Used to map IP-based protocols onto Layer2 only transactions and restoring the IP-level interactions at the egress of the FLAME network.
- *Routing*: Used to support various constraint-based routing decisions as well as manage the topology and forwarding information used for the data plane, including the assignment of IP addresses towards media components in the FLAME platform.
- *Registration*: Used to support the registration of service function endpoints (SFEs).
- *Resource management*: Use to support link failover and QoS through traffic classes
- *Diversity support*: Used to support multi-source retrieval, net-level indirection as well as in-session switching for HTTP
- *Mobility*: Used to support direct path mobility of users as well as UE mobility use cases
- *Security*: Used to support data plane encryption and resilience for centralised components.

Feature ID	Req	Feature Description	Component Interface	Release
Protocol Mapping				
SFR-1	Req-SR1/2/3	Implement HTTP level protocol mappings according IDE specifications for HTTP-over-ICN	HTTP/IP	Alpha
SFR-2	Req-SR1	Implement IP level protocol mappings according to IDE specifications for IP-over-ICN	HTTP/IP	Alpha
SFR-3	Req-SR1	Implement IP multicast protocol mappings according to IDE specifications for IP-over-ICN	HTTP/IP	RC
Routing				
SFR-4	Req-SR7	Support for shortest path routing	HTTP/IP	Alpha
SFR-5	Req-SR8	Support for geo constrained routing	HTTP/IP	RC
SFR-6	Req-SR9	Support for policy routing	HTTP/IP	RC
SFR-7	Req-SR7/8/9	Parse topology information model	Topology target	Alpha
SFR-8	Req.SR1/2	Support for topologies larger than 256 links	Topology target	RC
SFR-9	Req.SR1	Managed DHCP-based IP address assignment	HTTP/IP	Alpha+

Co-funded by the Horizon 2020 Framework Programme of the European Union

Registration				
SFR-10	Req.SR11	FQDN registration based on configuration	FQDN Registration	Alpha
SFR-11	Req.SR11	FQDN registration based on distribution protocol	FQDN Registration	Alpha+
Resource Management				
SFR-12	Req.SR10	Support for traffic classes based on protocol classes or FQDN	HTTP/IP	RC
SFR-13	Req.SR12	Support for link failure through path updates	HTTP/IP	Alpha
Diversity support				
SFR-14	Req.SR13	Support HTTP in-session switching	HTTP/IP	Alpha
SFR-15	Req.SR5	Support HTTP multi-source retrieval	HTTP/IP	RC
SFR-16	Req.SR4	Support HTTP net-level indirection	HTTP/IP	RC
Mobility				
SFR-17	Req.SR6	Support UE-level inter-SR mobility	HTTP/IP	Alpha
SFR-18	Req.SR6	Support UE mobility in 5GLAN type environment, i.e., SR integrated with UE	HTTP/IP	RC
Security				
SFR-19	Req-SR3 Req-S1	Support for HTTPS & TLS	HTTP/IP	Alpha
SFR-20	Req.SR1	Support against PCE failure	HTTP/IP	RC
SFR-21	Req.SR1 Req.SEM2	Support for FQDN authority delegation	HTTP/IP	RC
SFR-22	Req.SR1 Req.SEM2 Req.SR3 Req.S1	Support for manual content certificate distribution	HTTP/IP	Alpha
SFR-23	Req.SR1 Req.SEM2 Req.SR3 Req.S1	Support for automatic content certificate distribution	HTTP/IP	Alpha+

Table 8: SF Routing features

The SFR component delivers features to the media component in the form of data plane connectivity at the level of HTTP/IP based protocols, as outlined in the relevant service function chain in Figure 8. The delivery is expected to be organised around key interface features. The SF routing component is responsible for realising such data plane connectivity based on the availability of SF endpoints in the FLAME network and the current conditions of the transport network, e.g., in the form of available links being available. For this, the routing feature parses the topology information model provided by the orchestration component to suitably configure the infrastructure component, while initially providing shortest-path routing functionality to the protocol mapping feature. The latter realises the media component facing IP protocol termination and mapping onto Layer 2 protocol exchanges. It utilises the registration information realised by the registration feature, while providing the basis for in-session switching for HTTP, realised by the diversity support feature, and for encryption support, provided by the security feature.

Co-funded by the Horizon 2020
Framework Programme of the European Union

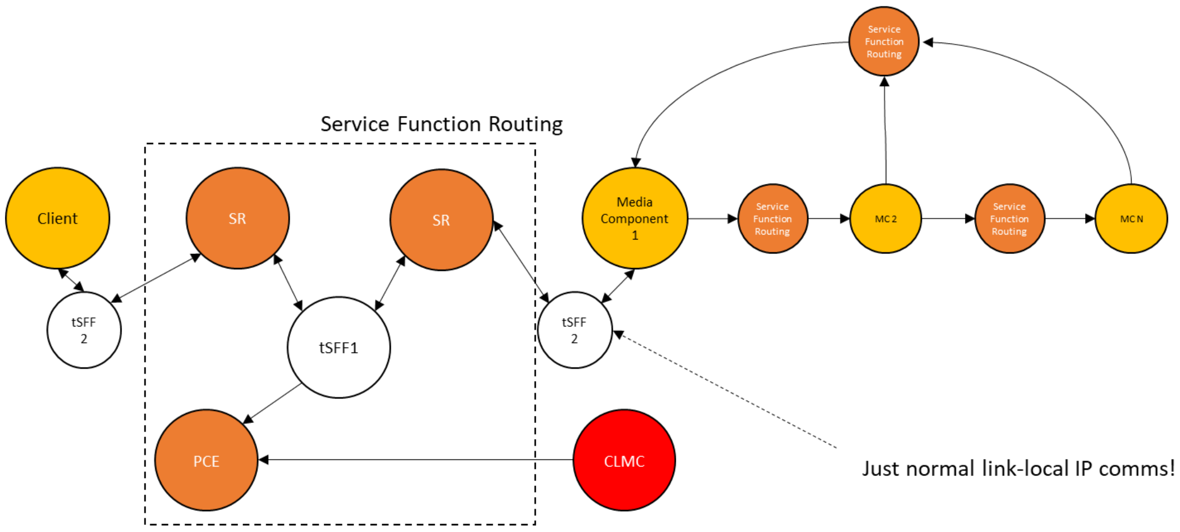


Figure 8: Relevant SFR Service Function Chain for Alpha+ Release

For the alpha+ release accompanying this deliverable and the soon-to-come beta release, we expect all major features of the SFR to be realized while a number of features will remain for the release candidate at the end of the project. Most relevant from a media service perspective is the realization of the multi-source retrieval (using network coding) as well as the network-level indirection. From an operational viewpoint, features on PCE resilience and support for larger topologies are being deferred to the release candidate while we foresee also new features such as 5GLAN-compliant terminal mobility to be realized in the release candidate.

Table 9 provides a summary of the orchestration implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the orchestration are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	Technology Starting Point	License	Expected enhancements	Expected foreground ownership	TRL
SR & PCE	FLIPS	Access via separate license agreement	Realization of features according to alpha feature table	IDE	6
SDN Controller	FloodLight	ASLv2	Integration	IDE	6

Table 9: SFR implementation technology summary

3.1.1.4 Cross Layer Management and Control

The CLMC component supports the monitoring, measurement and assessment of media service and platform performance, in addition to configuration of processes supporting the integration and organisation of data for analytics.

The features of the CLMC are defined in Table 10. These features are organised in accordance with the interfaces towards other system components including:

Co-funded by the Horizon 2020 Framework Programme of the European Union

- *Media Service Configuration*: Used to monitor changes in media service configuration including the lifecycle of a media service instance, media components and service functions
- *Monitoring*: Used to monitor the state and performance of media services including all configured items (media service instance, media components and service functions) and the performance of underlying infrastructure resources allocated to them
- *Analytics*: Used to integrate and aggregate higher-level facts about media service KPIs and dimensions
- *Query*: Used to query, filter and visualise media service and platform information
- *Performance Configuration and Notification*: Used to specify and monitor specific KPI performance metrics of interest
- *Security*: Used to control access to information to those that are authorised to do so.

Feature ID	Req	Feature Description	Component Interface	Release
Media Service Configuration				
CLMC-1	Req-C1	Define media service information model	Config	Alpha
CLMC-2	Req-C1	Define configuration information model including failure taxonomy	Config	Alpha
CLMC-3	Req-C1	Store configuration data	Config	Alpha
CLMC-4	Req-C1	Monitor media service lifecycle config events	Config	Alpha
CLMC-5	Req.C6	Monitor SF lifecycle config events (including geolocation) for SRs, hosts and service function instances	Config	Alpha
CLMC-6	Req.C2	Flexible configuration of dimensional data abstractions	Config	RC
Monitoring				
CLMC-7	Req-C1	Define monitoring information model	Monitoring	Alpha
CLMC-8	Req-C1	Monitoring data acquisition for media component, service function endpoint and service function routing	Monitoring	Alpha
CLMC-9	Req-C1	Store monitoring data	Monitoring	Alpha
CLMC-10	Req-C1	Delete monitoring data	Monitoring	Alpha
Analytics				
CLMC-11	Req.C2	Basic monitoring data aggregation functions	Analytics	Alpha
CLMC-12	Req.C2	Dimensional data abstraction across (time, space, content representation, content navigation, resource configuration, etc.).	Analytics	Beta
CLMC-13	Req-C2	Define data quality model for accuracy, completeness, timeliness and consistency	Analytics	Beta
CLMC-14	Req.C2	Generation of new media service templates with human in the loop	Analytics	Beta
CLMC-15	Req.C2	Generation of new media service templates through machine learning	Analytics	RC
Query				
CLMC-16	Req.C2	Query monitoring data	Query	Alpha

Co-funded by the Horizon 2020
Framework Programme of the European Union

Feature ID	Req	Feature Description	Component Interface	Release
CLMC-17	Req.C2	Visualise monitoring data	Query	Alpha
CLMC-18	Req.C2	Query by KPIs and dimensions	Query	Beta
Performance Configuration and Notify				
CLMC-19	Req.C2	Specification of KPIs for measured facts	KPI	Alpha
CLMC-20	Req.C2	Monitor KPI events based on measured facts	KPI	Alpha
CLMC-21	Req.C2	Monitor KPI events based on aggregated facts	KPI	Alpha
CLMC-22	Req.C7	Publish and subscribe to KPI events	KPI	Alpha
Security				
CLMC-29	Req.C7	Restrict access to stakeholder viewpoints on monitoring data	Security	Beta

Table 10: CLMC features

The CLMC implementation depends on the specification for a media service. According to the D3.3 architecture:

> *"[Media Service] Specification of the descriptors required for the definition, deployment and management of Media Services, including dynamic behaviours that can be explored within experimentation, testing and operations. Specification-Language-compliant Templates will be available for the Media Service Providers to make the definition of Media Service easier. The Specification Language will take into account current orchestration specs for cloud environments, such as TOSCA."*

The media specification provides the logical configuration structure for a media service. This structure defines context for monitoring information acquired when the media service is operated. The structure offers key relationships between information whilst the logical naming of service functions will allow for monitoring data to be integrated through the use of correct references. The overall naming scheme for items within the media service specification is a critical input for different aspects of the CLMC information model.

The CLMC delivers features to all other Platform components. The delivery is expected to be organised around key interface features. The orchestrator and SF Endpoint Management and Control components are responsible for changing resourcing configurations in response to media service provisioning events and media service demand. These events need to be captured by the CLMC to track the changes in service configuration over time. For the alpha release the event logs will be limited to key media service and SF lifecycle events. A protocol for reporting configuration events including the message format with pub/sub service implementation is needed. With the configuration in place, the CLMC has the context for monitoring information produced by different system components. The infrastructure, platform and media services are Monitoring Producers that depend on the availability of a pub/sub monitoring pipeline that offers a protocol and a messaging format for monitoring data. Although these two feature streams are related they can be implemented in parallel if the information model is agreed and information exchange is achieved through pub/sub protocols.

The SFC for the CLMC is shown in Figure 9. The SFC is designed to allow SFs instances to be distributed in the same way as we expect for media services. During the development it is possible to deploy SFCs 3-8 within a single VM. However, for production, we'd expect these to be distributed across different VMs depending on scale and performance needs.

Co-funded by the Horizon 2020 Framework Programme of the European Union

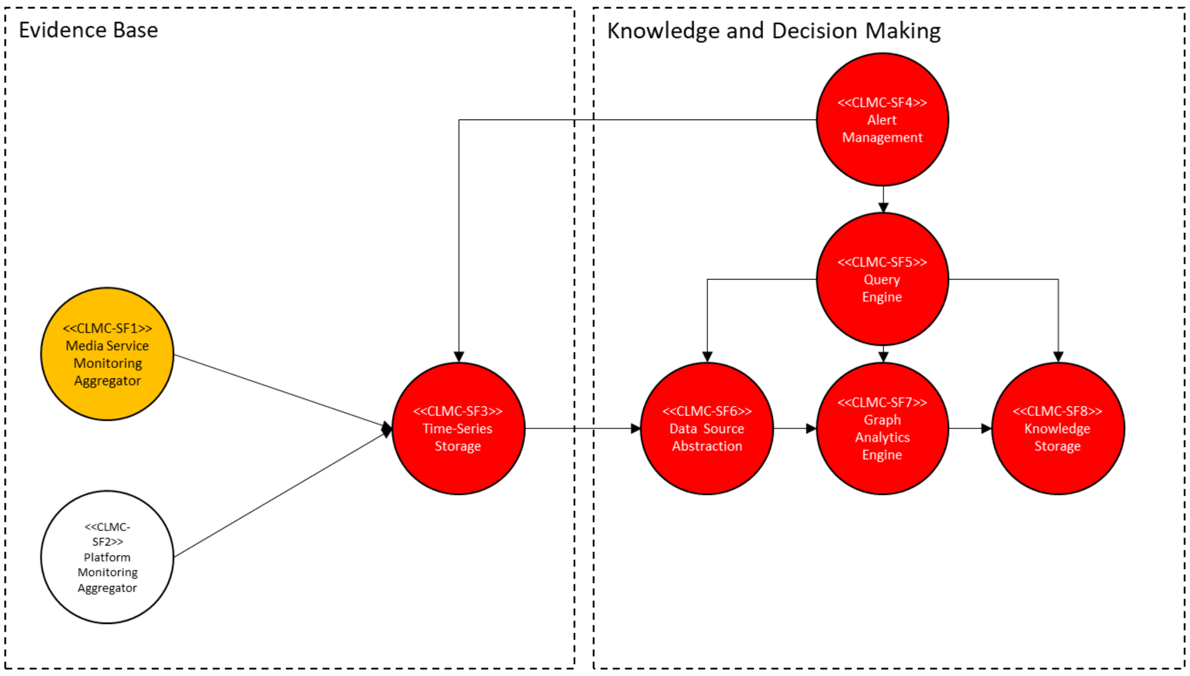

Evidence Base
Knowledge and Decision Making
<<CLMC-SF1>> Media Service Monitoring Aggregator
<<CLMC-SF2>> Platform Monitoring Aggregator
<<CLMC-SF3>> Time-Series Storage
<<CLMC-SF4>> Alert Management
<<CLMC-SF5>> Query Engine
<<CLMC-SF6>> Data Source Abstraction
<<CLMC-SF7>> Graph Analytics Engine
<<CLMC-SF8>> Knowledge Storage

Figure 9: CLMC Service Function Chain

The CLMC service function chain will be implemented through adaptation and enhancement of an existing open source software that has been developed for the purpose of service monitoring. The key for the alpha phase is to put in place the technologies supporting the acquisition of the data. Higher level analytics can then be implemented in later releases to help improve the way media services are managed. Table 11 provides a summary of the CLMC implementation technologies including the licenses, expected enhancements, foreground and TRL starting point. All background technologies of the CLMC are offered on permissive software licenses that allows aggregation and distribution of foreground in accordance with the Platform Product distribution within the project and beyond to 3rd parties wanting to evaluate the software.

Service Function ID	**Technology Starting Point**	**License**	**Expected enhancements for CLMC Service**	**Expected foreground ownership**	**TRL**
CLMC-SF1	Telegraf	MIT	Integration of measurement points from specific media services	Atos	6
CLMC-SF2	FLIPS or Telegraf	Commercial and MIT	Integration of measurement points from platform services	IDE and ITINNOV	6
CLMC-SF3	Influx DB	MIT	Configuration of transactional time series data model; Configuration of time series data aggregation functions.	ITINNOV	6
CLMC-SF4	Kapacitor	MIT	TOSCA Alert Specification integrated to stream-processing engine	ITINNOV	6
CLMC-SF5	InfluxDB	MIT	Information model for platform supporting context for measurement data	ITINNOV	6
CLMC-SF6	InfluxDB	MIT	Configuration of data source abstraction functions using temporal analysis of data points	ITINNOV	6
CLMC-SF7	Neo4J	GPL v3	Dynamic graph building derived from system topologies, and graph	ITINNOV	6

Co-funded by the Horizon 2020 Framework Programme of the European Union

Service Function ID	Technology Starting Point	License	Expected enhancements for CLMC Service	Expected foreground ownership	TRL
			analytics supporting understanding of system level behaviours		
CLMC-SF8	Either SQL DB or Graph DB	TBD	A repository for storage of configuration templates and system response datasets that can be browsed and queried	ITINNOV	6

Table 11: CLMC implementation technology summary

3.1.2 Summary of Platform Releases

Table 12 provides a summary of component features across each platform release. As we can see from this overview, although positioned as two intermediary releases, most features are provided for the alpha+ release in preparation for the open call and validation trials. This release is then tested in deployment infrastructures and released with the same feature set albeit hardened through those tests as the beta release in January 2019. It will therefore be effectively the release used by experimenters. Any new features are pushed to the release candidate and will benefit the second wave of experimenters in FLAME.

Component	Alpha+ Features (Oct-18)	Beta Features (Jan-19)	RC Features (Dec-19)
Orchestration	• Parse FLAME-TOSCA, as defined in T4.1, template and check for consistency • Support container based media service orchestration • Provide TOSCA template to FLAME platform orchestrator • Full consistency check of FLAME-TOSCA template, including consolidating deployment state with orchestration request • Provide SF endpoint control policies in TOSCA template extensions towards SFEMC component • Provide SF endpoint state information in TOSCA template extensions towards SFEMC component		
Service Endpoint Management and Control	• Parse surrogate policy based on FLAME-TOSCA template extension • Parse surrogate policy based on TOSCA template extension		• Allow for delegated name registration for SF endpoint images

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	Alpha+ Features (Oct-18)	Beta Features (Jan-19)	RC Features (Dec-19)
	• Query monitoring data • Decision logic matching monitoring data against policy constraints • Initialise and maintain SF endpoint state • Maintain SF endpoint compute/storage images		
Service Routing	• Monitor service function routing • Implement HTTP level protocol mappings according to IDE specifications for HTTP-over-ICN • Implement IP level protocol mappings according to IDE specifications for IP-over-ICN • Support for shortest path routing • Parse topology information model • FQDN registration based on configuration • FQDN registration based on registration distribution protocol • Support for link failure through path updates • Support HTTP in-session switching • Support UE-level inter-SR mobility • Support for HTTPS & TLS • Support for manual content certificate distribution • Managed DHCP-based IP address assignment • Support for automatic content certificate distribution		• Implement IP multicast protocol mappings according to IDE specifications for IP-over-ICN • Support for geo constrained routing • Support for policy routing • Support for topologies larger than 256 links • Support for traffic classes based on protocol classes or FQDN • Support HTTP multi-source retrieval • Support HTTP net-level indirection • Support UE mobility in 5GLAN • Support against PCE failure • Support for FQDN authority delegation

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	Alpha+ Features (Oct-18)	Beta Features (Jan-19)	RC Features (Dec-19)
CLMC	• Define media service information model • Define configuration information model • Store configuration data • Monitor media service lifecycle config events • Monitor SF lifecycle config events (including geolocation) for SRs, hosts and service function instances • Define monitoring information model • Monitoring data acquisition for media component, service function endpoint and service function routing • Store monitoring data • Delete monitoring data • Basic monitoring data aggregation functions • Query monitoring data • Visualise monitoring data • Specification of KPIs for measured facts • Monitor KPI events based on measured facts • Monitor KPI events based on aggregated facts • Publish and subscribe to KPI events • Define data subject information model • Define information security model • Query for data related to a data subject • Delete data related to a data subject	• Dimensional data abstraction across (time, space, content representation, content navigation, resource configuration, etc.) • Define data quality model for accuracy, completeness, timeliness and consistency • Generation of new media service templates with human in the loop • Query by KPIs and dimensions • Secure communication of personal data • Restricted access to personal data • Restrict access to stakeholder viewpoints on monitoring data	• Flexible configuration of dimensional data abstractions • Generation of new media service templates through machine learning

Table 12: Platform feature roadmap

Co-funded by the Horizon 2020
Framework Programme of the European Union

4 MEDIA SERVICE PRODUCT ROADMAP

Similar to the platform product roadmap in Section 3, this section provides an overview of the roadmap for the foundational media services provided by the FLAME project. The updates compared to D3.5 include information for the packaging for those services and an update of the functionalities provided by those services and their availability in the future Alpha+ and Beta releases.

4.1.1 Media Services Overview

A media service product is a software product offering content production, management and/or distribution capabilities. Media service products are integrated, tested and packaged including a default template specification for deployment on a FLAME platform to create media services. A media service product is dependent on one or more media component products implementing underlying service functions within the overall media service function chain.

A media service product is no more than a set of media components described in terms of topology, performance and resourcing using templates. Media services themselves are not part of the FLAME platform but are deployed and managed by it. The FLAME platform orchestrates the deployment of media components as well as internal service functions. Throughout the project the goal is to build an initial set of foundation media services and then extend the available media services through developments conducted by 3rd parties.

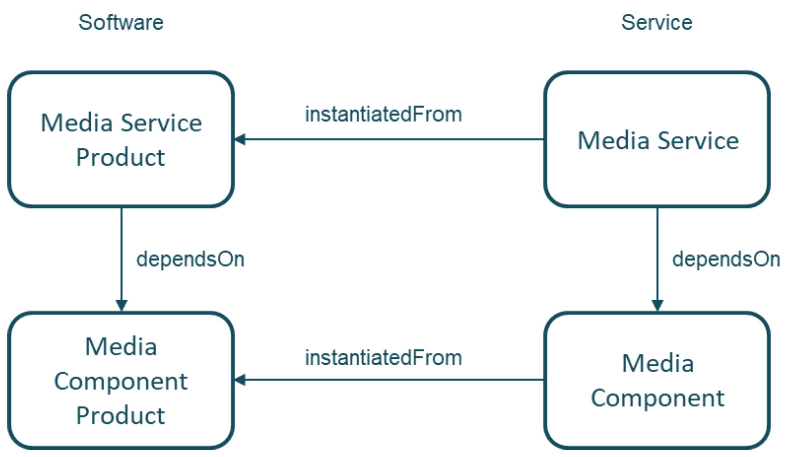


Figure 10: Media services and media components

4.1.2 Media Service Packaging and Provisioning

The packaging of service functions is fully decoupled from the provision. As illustrated in the figure below, the packaging of a service function is conducted outside of the FLAME platform using a bash-based toolchain provided by the FLAME consortium. The provisioning of service functions then is achieved through the FLAME Orchestrator which requires a TOSCA-compliant descriptor communicating the resource configurations for each service function to the platform's SFEMC and the lifecycle policy which should be invoked upon receiving a specific trigger from CLMC.

Co-funded by the Horizon 2020
Framework Programme of the European Union

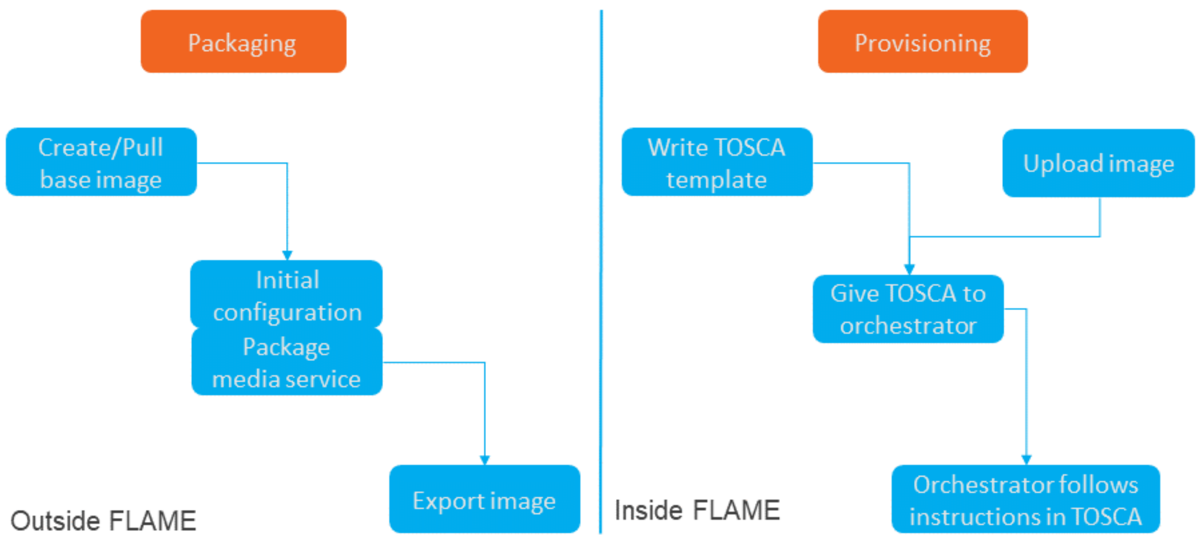


Figure 11: Service function packaging and provisioning workflows

All FLAME clusters have the KVM and LXD hypervisor installed which allows the media service provider to choose from the two at packaging time. The provided packaging toolchain hides the complexity of installing and configuring the hypervisors and only demands an Ubuntu 16.04.5 64bit machine set up outside the FLAME platform where the media service provider runs the toolchain which takes care of creating the desired base image, configures it to FLAME's needs and exports it as a compressed TAR ball after the media service provider has copied their software into the virtual instance and configured it according to their needs. Note, the supported base images for KVM and LXD are Debian derivates and CentOS at the moment with plans to extend the range of supported Linux flavours based on the experimenters' feedback.

4.1.3 Media Component Products

FLAME has defined a list of media component products that offer common capabilities necessary to construct media service products. For example, a content conditioning process will require transcoding and trans-rating media components. The initial media components products are used to create FLAME's "Foundation Media Services" (FMS) providing examples of capabilities that benefit from the FLAME platform. The foundation media components and services form part of the FLAME offering.

Due to the number and variety of the foundation media components criterion have been established to select components to be implemented. Firstly, the prioritisation process has carefully analysed the validation scenarios proposed in FLAME. D3.1 – FMI Vision, Use Cases and Scenarios describes the first version of the mentioned validation scenarios [FLAME-D3.1]. The validation scenarios considered in FLAME are:

- Participatory media for interactive radio communities (City Fame)
- Personalised media mobility in urban environments (Follow Me)
- Collaborative interactive transmedia narratives (Interactive Storytelling)
- Augmented reality location-based gaming (Gnome Trader).

Some of the foundation media services have been specifically defined to cover modules of these validation scenarios, as described in D3.3 FLAME Platform Architecture and Infrastructure Specification

Co-funded by the Horizon 2020 Framework Programme of the European Union

V1 [FLAME-D3.3]. Secondly, the prioritisation of the media services has considered the suitability of the media services to show the advantages of the FLAME platform benefits and technical innovations as described in Section 2.5 of D3.3. Finally, the prioritisation process has considered the terms for the different FLAME releases and the development requirements for each service. Additionally, a certain foundation media service can include different implementations along the project (e.g., involving different existing software initiatives) with the respective temporal terms.

Table 13 identifies the media components that fulfil the requirements of modules identified in the FLAME validation scenarios. These foundation media components can be clustered according to their functionality and the step they cover in the production and distribution chain, as detailed later. During the component implementation activities, some media components planned in D3.5 have been combined. Table 13 also clarifies the previous IDs and names of the respective functionalities, as stated in the previous version of the deliverable.

FMS	**ID and Name in D3.5**	**Description**
Metadata database	MC-1 Media content database MC-12 Metadata Transmission and Management	This service consists in a generic database, which is a required module in most of media services. Thus, the four FLAME validation scenarios include a database, as shown in FLAME deliverable D3.3. Metadata consists of data that describe the media assets. This database keeps two different kinds of information: 1) objectives and quality parameters of the media assets stored in the media service and 2) a description (name, synopsis, duration) of the assets. For the second kind of information, FLAME offers a schema by default. However, it also admits a schema supplied by the media service provider since media producers and distributors are used to consider their own content categories and description formats.
Media quality analysis	MC-2 Media Quality Analysis	The objective of this service is the evaluation of the media characteristics of a certain media asset. These characteristics include data such as the resolution, the video and audio codecs and also an automatic estimation of the quality. This last functionality may be required to determine the suitability of contents provided by prosumers.
Content ingest and storage	MC-3 Content Ingest MC-4 Content Storage	This media component enables the insertion and hosting of media assets to make them available in a media service. Concerning content ingest, this component will satisfy two different functionalities. On one hand, it will enable the provision of contents to deploy an experiment. In this case, the service is used before the experiment deployment. For instance, a media service provider may want to test a Video-On-Demand (VoD) service using the FLAME platform. This service would allow the provider to “upload” the assets. On the other hand, this service will enable the ingest of content as a part of an experiment, as in the City Fame scenario. In this case, the service is used during the experiment itself. Besides the ingest, this media component is in charge of storing the media assets for the provision of the services. This kind of functionality is widely required by media services. This is for example the case of a video on demand service. This component satisfies the requirements of the content provisioning module in the Follow me scenario, among others. This FMS also works as a retrieval component, able to deliver assets.
Virtual CDN	MC-10 Virtual CDN MC-5 Content Caching Management	This service consists in the creation of a CDN using virtual nodes to optimise the advantages of this kind of networks, such as bitrate, low latency, load balance and scalability. CDNs perform caching of data to enable faster access by the end users. Moreover, CDNs approach content to end users with high availability and high performance. Video distribution networks are

Co-funded by the Horizon 2020 Framework Programme of the European Union

		typically CDNs. FLAME benefits, via Platform products, enable new ways of implementing CDNs. This component satisfies two different functionalities: the distribution of content replicas across the infrastructure and the caching management. Whereas the first functionality requires a previous design of the content delivery, the second one is reactive and enabled when the required piece of content is not stored in the respective node. The mechanism specified in this service is intended to work jointly with the metadata database and the content ingest and storage This component satisfies the functionalities of the Caching manager module in the Interactive Storytelling scenario.
CMS	MC-6 Content Management System	The content management system or CMS is a widely used media component that supports the creation and modification of digital content. These systems usually offer a web user interface to control the existence and availability of media assets.
Transcoding, transrating and content conditioning	MC-8 Transcoding and Transrating MC-7 Content Conditioning	Transcoding consists in the change of the video or audio specification to represent the content of an asset. This kind of encoding is named source encoding. For example, this component is in charge of converting an AVC/H.264 video clip into a HEVC/H.265 one. The use of a more recent specification (and this is the case of HEVC with regard to AVC) enables a reduction of the bitrate required to represent the information while it preserves the subjective quality. Transrating is a similar process but in this case the encoding specification does not change. It typically consists in an additional source encoding to reduce the bitrate (this processing will cause a reduction of the quality, too). Conditioning is the processing of the media assets to make them available. For example, assets must be split in chunks and encoded at different bitrates to offer a video-on-demand adaptive streaming service.
Adaptive streaming	MC-9 Adaptive Streaming	Adaptation is the process that allows a player to consider the network (and the receiver) capabilities to automatically and instantaneously adapt the transmitted bitrate (and the quality) in a streaming service. In this way, adaptive streaming optimises the instantaneous quality along the asset duration.
Adaptive data transmission	MC-11 Adaptive Data Transmission	This component extends the mentioned adaptive video concept to other kinds of data transmission. For example, a certain media service could require the transmission of 3D models to be rendered in the user equipment or in AR applications. This component optimises the bitrate (and quality) of the transmission of this additional data. The Gnome trader scenario requires this kind of functionality.

Table 13: Media component products

As stated in the description of these media services and components, several of them are related. Two main clusters can be distinguished:

- Services and components related to content management and processing (e.g., content ingest and storage, content management system and transcoding, transrating and content conditioning).
- Services and components related to information transmission and distribution (e.g., adaptive streaming, virtual CDN, adaptive data transmission).

4.1.4 Releases and Media Services Product Implementation Roadmap

Table 14 summarises the roadmap for the implementation of media service products in FLAME according to the prioritisation criteria explained in the previous subsection.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Component	**Alpha+**	**Beta**	**Comments**
Metadata database	Yes	Yes	All the validation scenarios contain a database. It can show FLAME benefits. This implementation in FLAME will be based on mongoDB.
Media quality analysis	No	Yes	This service is planned for the beta release.
Content ingest and storage	Yes	Yes	A first version of this component is included in the alpha release. It will be improved for the beta release.
Virtual CDN	No	Yes	Virtual CDN. FLAME proposes an innovative implementation based on FLIPS, one of the new technologies involved in FLAME. The design of this implementation is planned for the medium term. For this reason, this component will be ready for the beta release.
CMS	No	Yes	This service is planned for the beta release.
Transcoding,transrating and content conditioning	Yes	Yes	Different versions of this component are included and planned for both alpha and beta releases, depending on 1) software to perform the encoding and 2) solution conceived for live content or video on demand. We propose two different software initiatives: Wowza and FFMPEG. Wowza is a commercial and consolidated product for the deployment of adaptive streaming services, including transcoding and transrating whereas FFMPEG is an open source initiative that provides a variety of encoding tools. FLAME will offer the encoding formats covered by these external tools. Particularly, for the beta release, FLAME will include the new and efficient HEVC encoding format.
Adaptive streaming	Yes	Yes	A first version of the adaptive streaming service is available in the alpha release. The service will be refined for the beta release. The alpha release is based on Wowza. The beta release will include nginx and nginx-ts-module to enable de streaming engine. Additionally, different tools may be integrated in the beta release, according to the evolution of available streaming initiatives. This service will support different adaptive streaming technologies and particularly MPEG-DASH and HLS.
Adaptive data transmission	No	Yes	This service is planned for the beta release.

Table 14: Media services release plan

Co-funded by the Horizon 2020
Framework Programme of the European Union

5 INTEGRATION PRODUCTS

As outlined in Section 2.2, FLAME provides several integration products, which we will detail in the following sections. Specifically, those products cover, as updates to D3.5, the range from a localized sandbucket environment over a staging sandpit environment to the replication infrastructures in Bristol and Barcelona.

5.1 SANDBUCKET ENVIRONMENT

FLAME-in-a-Box is a VirtualBox-base mini-FLAME platform which allows for testing

1. SFC orchestration templates
2. SF provisioning
3. Basic communication tests of deployed SFEs

All instances that come as a single OVA and can run on a normal laptop 4 cores and 8GB of RAM.

FLAME-in-a-Box is composed of the following nodes:

- **ue**: A client (service initializer) for all deployed SFEs and the platform
- **vbox-cluster**: The cluster into which SFEs are getting deployed
- **vbox-sr-ue**: The service router for the ue and vbox-cluster
- **vbox-pce-sfemc**: The path computation and service function management and control instance
- **floodlight**: The SDN controller for FLAME-in-a-Box
- **vbox-sr-ps**: The service router acting as the GW for all other IP endpoints in the platform
- **vbox-ps**: The platform service instance which hosts the DHCP server, DNS. IP GW and the SF repository

The resulting logical topology is illustrated in the figure below, which shows the UE and the vbox-cluster nodes connected to the vbox-sr-ue and the vbox-ps and vbox-pce-sfemc to vbox-sr-ps. The underlying SDN-enabled switching fabric with the innovative SFR is illustrated with blue lines.

Co-funded by the Horizon 2020
Framework Programme of the European Union

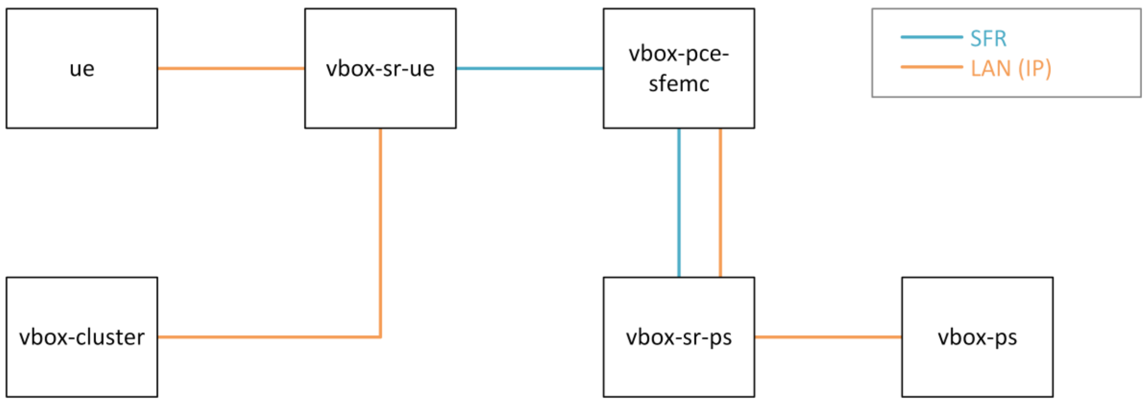


Figure 12: Logical topology of the FLAME-in-the-Box environment

The vbox-cluster node is configured to 4 cores (allowed CPU utilisation of 50%), 4096MB RAM and 15GB disk. As SFEMC, reserves 1 core, 1GB RAM and 1.5 GB of free disk for the cluster system only the remaining cloud resources are available to any deployed SFE.

5.2 SANDPIT ENVIRONMENT

The purpose of the Sandpit Environment is to support integration testing, functional testing and limited load testing of the Platform Product and Media Service Products. The infrastructure is designed to test product features within an infrastructure that replicates key aspects of the FLAME replication environments in Bristol and Barcelona. The infrastructure only clones part of the replication infrastructure due to cost constraints. However, by using a software-based infrastructure it can be flexibly configured to support different test cases representative of those expected in real-life trials.

Figure 13 shows the target logical topology of the Sandpit Environment data plane. The design is based on supporting a hierarchical topology of edge and metro data centres with different capacity constraints. Each SR represents a connection to a cluster and user equipment via an access network. Service Routers offer connectivity to media services deployed in clusters and allow User Equipment representing one or more end user devices to access the network and place load on the system.

This configuration offers a practical baseline for testing scenarios. The use of four switches allows different SF routes to be explored including cases of routing loops. The heterogeneity in DC and Edge resources allows SF endpoint management policies to be explored under different resourcing constraints. The distribution of User Equipment allows for demand to be generated from different parts of the network.

Co-funded by the Horizon 2020 Framework Programme of the European Union

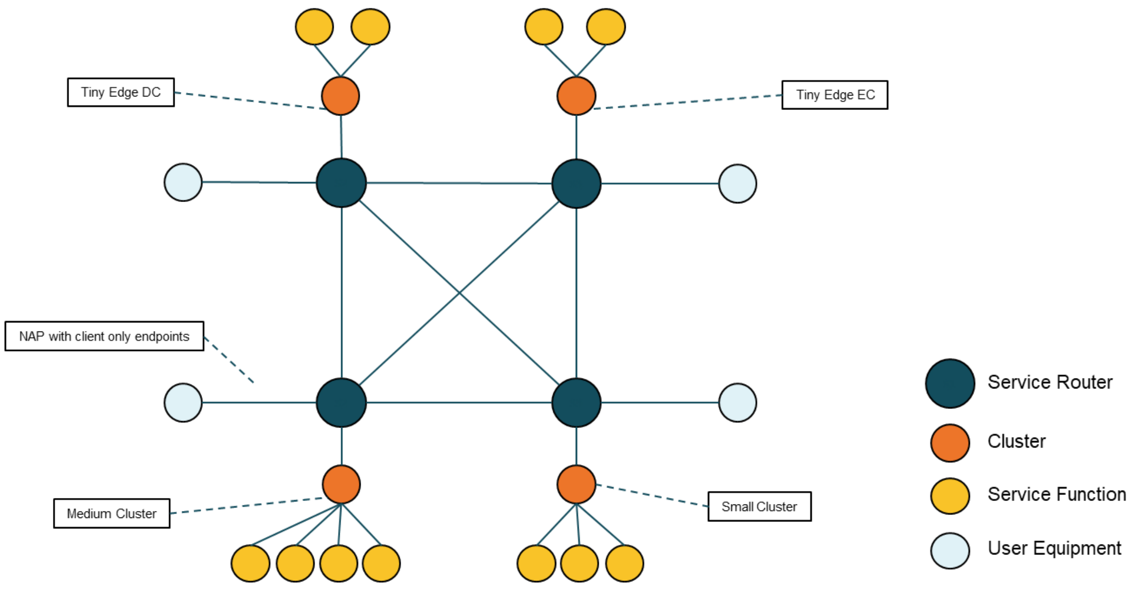


Figure 13: A logical topology of the Sandpit Environment data plane

The configuration is not fixed and different setups may be established for specific test cases or when resources need to be shared. For example, the media service resources may be aggregated to create a larger data centre or it in some situations may be more optimal to allocate resources to specific integration tests rather than offer the entire set up for each test.

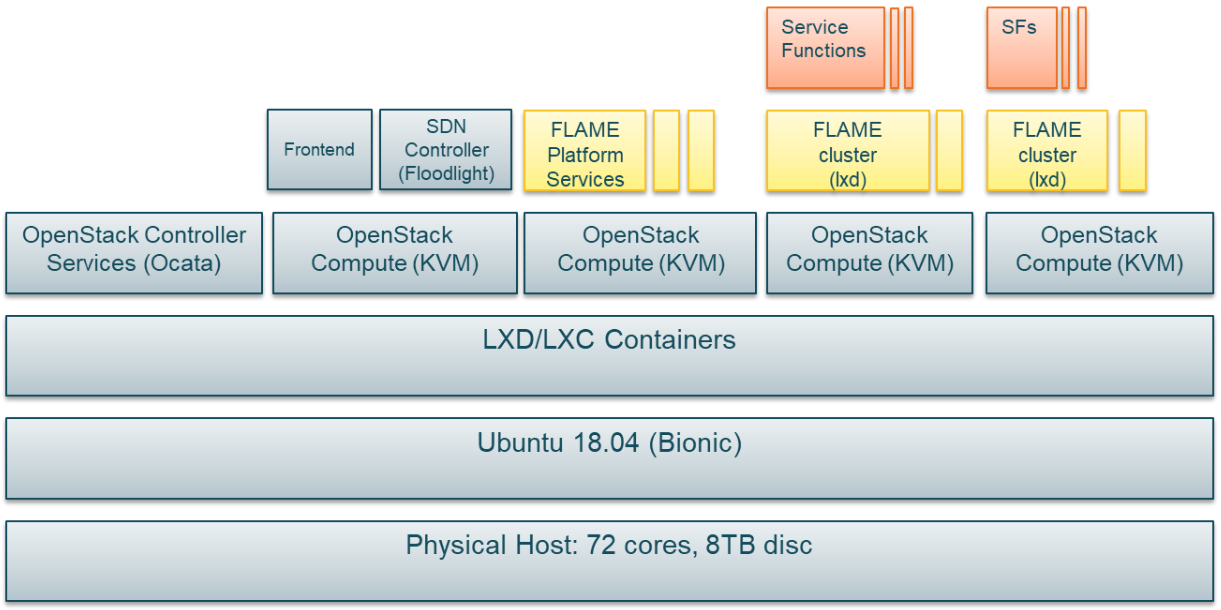


Figure 14: Sandpit stack

The Sandpit stack is shown in Figure 14. The physical infrastructure is a single machine with 72 cores and 8TB of disk. At the lowest level OpenStack and Floodlight are deployed to provide management of virtual compute and the SDN fabric. OpenStack controller services and compute nodes are deployed within LXD/LXC containers. This allows the topology of the compute infrastructure and the capacity constraints of each compute to be flexibly configure. There is no physical SDN fabric beyond the switches deployed as part of the FLAME platform itself. As such the SDN Controller is deployed as an OpenStack VM and made available exclusively to the FLAME platform tenant for registration of switches that are part of each FLAME service router. The Platform services are then deployed as a stack

Co-funded by the Horizon 2020
Framework Programme of the European Union

of VMs within an OpenStack tenancy. This includes all of the SFs required for FLIPS, Orchestrator and CLMC components as defined in 2.2 Even though the integration infrastructure is small scale and will not be implementing load balancing and clustering it is important to provide a reasonable mirroring of separation between components. To achieve this the infrastructure controllers and platform controllers are separated whilst the CLMC is also isolated considering the requirements for low contention on storage I/O. The FLAME platform is deployed using Heat orchestration according to a Sandpit infrastructure slice specification. The deployment uses ARDENT providing a consistent approach to deployment as used in all replication sites. A frontend node is deployed as an OpenStack VM to provide an http gateway to all platform services.

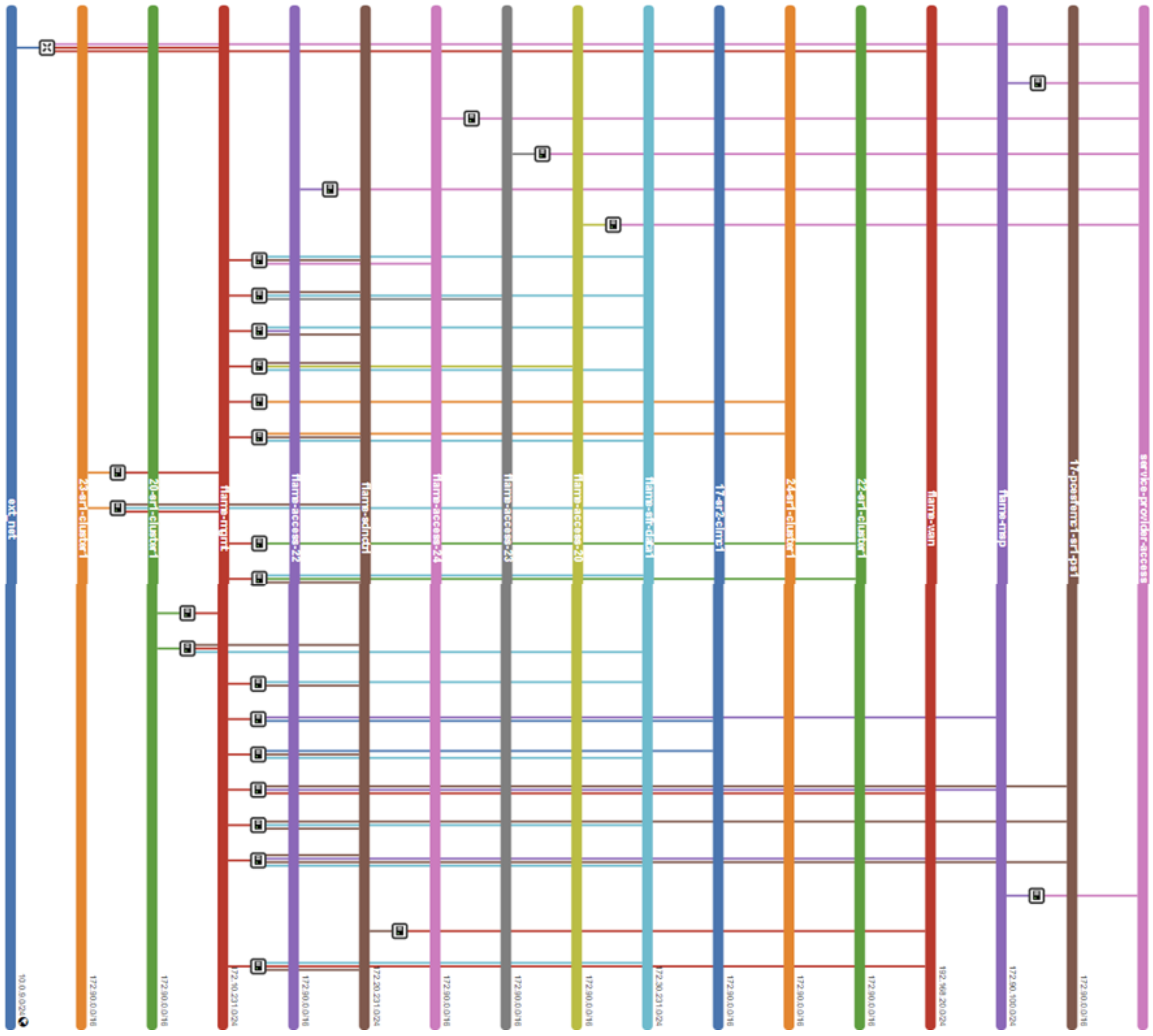


Figure 15: Sandpit deployment

The sandpit deployment for the scenario described in Figure 12 is shown in Figure 15. The figure shows a series of networks for control and data planes required by the FLAME platform. Access to the sand put is via “ext_net” using SSH. By using an SSH tunnels, clients such as web browsers can connected to the platform services, to emulated user equipment and to service function endpoints themselves.

Table 15 shows a capacity plan for the sandpit considering the logical topology (Figure 13), the services required for the management and control plane, and continuous integration services including load test drivers. Overall the total capacity required for the sandpit is:

Co-funded by the Horizon 2020
Framework Programme of the European Union

- number of VMs/Containers = 65
- number of CPUs = 68
- RAM = 148 GBytes
- storage = 5 TBytes

These values consider a standard media service size that can be used as the basis for functional testing of the Platform. The capacity plan does not consider how to support the requirements for all media services as these requirements have a high degree of variation and often require significant capacity beyond what can be provided in integration. Media services with service function chains requiring significant resources can only be tested at replication sites where such capacity exists.

VMTypes	**#elements (VMs)**	**#CPUs per VM**	**Total CPUs**	**RAM**	**Storage**	**Storage Type**	**Server**	**Workload assumptions**
Integration Services								
Continuous Integration	1	2	2	2	1000			Intermittent load based on build frequency, storage of build artefacts and test data
Infrastructure Product								
OpenStack Services	24	0.5	12	24	1000			Baseline estimate
Floodlight	1	1	1	8	100			Baseline estimate
Platform Product								
FLIPS PCE	1	1	1	1	20			Baseline estimate
FLIPS SR	4	1	4	4	80			Baseline estimate
FLIPS MOOSE	1	1	1	1	100			Baseline estimate
Orchestrator OSM	1	8	8	16	100	Disk		
CLMC Graph Database	1	4	4	32	500	Disk		Buffer Mem = write_throughput * buffer_seconds Storage = write_throughput * log_retention_hours Separate drive to avoid disk IO contention with other services
CLMC Time Series DB	1	4	4	24	1000	SSD, IOPS 500		Assuming a single node Low = 5K writes a second, 5 queries a sec, 100K unique series Med = 250K, 25K, 1M Low Compute CPU: 2-4 cores RAM: 2-4 GB IOPS: 500 Storage Size Non-string values require approximately three bytes. String values require variable

Co-funded by the Horizon 2020 Framework Programme of the European Union

VMTypes	#elements (VMs)	#CPUs per VM	Total CPUs	RAM	Storage	Storage Type	Server	Workload assumptions
								space as determined by string compression.
CLMC Service and Dashboard	1	1	1	2	1	Disk		
Media Service Products								
Media Service Functions	24	1	24	24	600	Disk		Depends on scenarios
User Equipment	4	1	4	8	200	Disk		Depends on the test scenarios, could be co-located with the CI server will most likely be idle when the tests are running unless we run parallel build and integration tests
Totals	**65**	**0**	**68**	**148**	**4801**			

Table 15: Sandpit infrastructure capacity planning

The final allocation of VMs to servers is a trade-off between performance, isolation and cost. Consolidating VMs on fewer servers will reduce the cost of the integration infrastructure. However, this will result in poorer performance and more contention between the components during tests increase difficulty and time to resolve defects on test failure.

5.3 INFRASTRUCTURE ENVIRONMENT

The following two sub-section will provide insights into the staging and production environments we have established in the two replications located in Bristol and Barcelona. Specifically for the Bristol environment, the update compared to D3.5 includes the move from the Bristol-is-Open environment to the University of Bristol environment at Millennium Square, while the Barcelona infrastructure is updated to capture the insights from the first replication phase.

5.3.1 Bristol Infrastructure

5.3.1.1 Bristol Staging Infrastructure Specification

The purpose of the infrastructures is to support the acceptance testing of Platform Product and Media Service Products on the UoB 5GUK Test Network Infrastructure Product.

Co-funded by the Horizon 2020 Framework Programme of the European Union

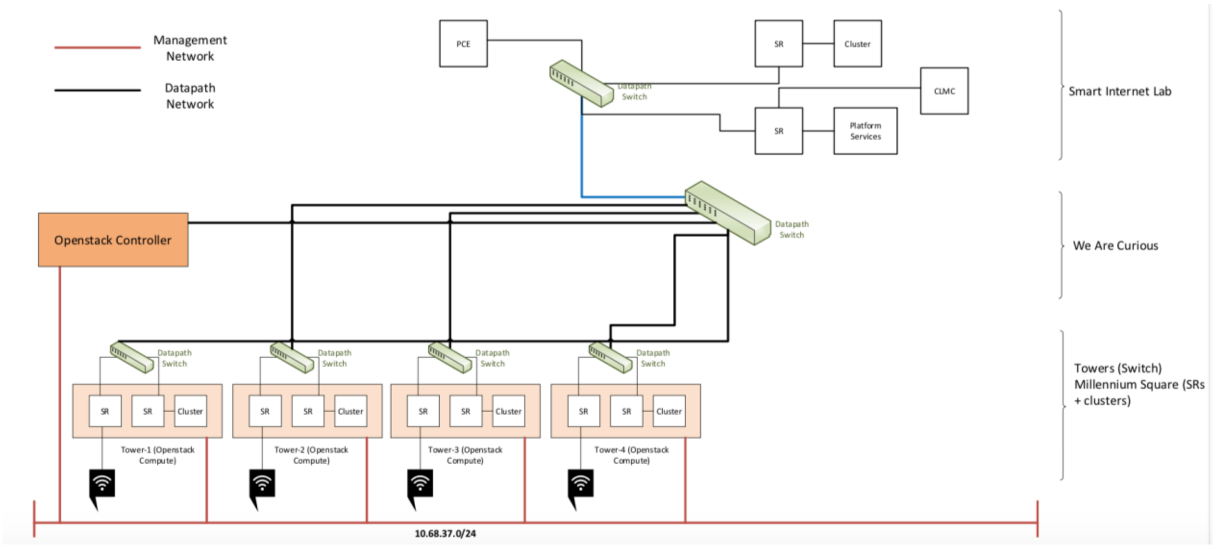


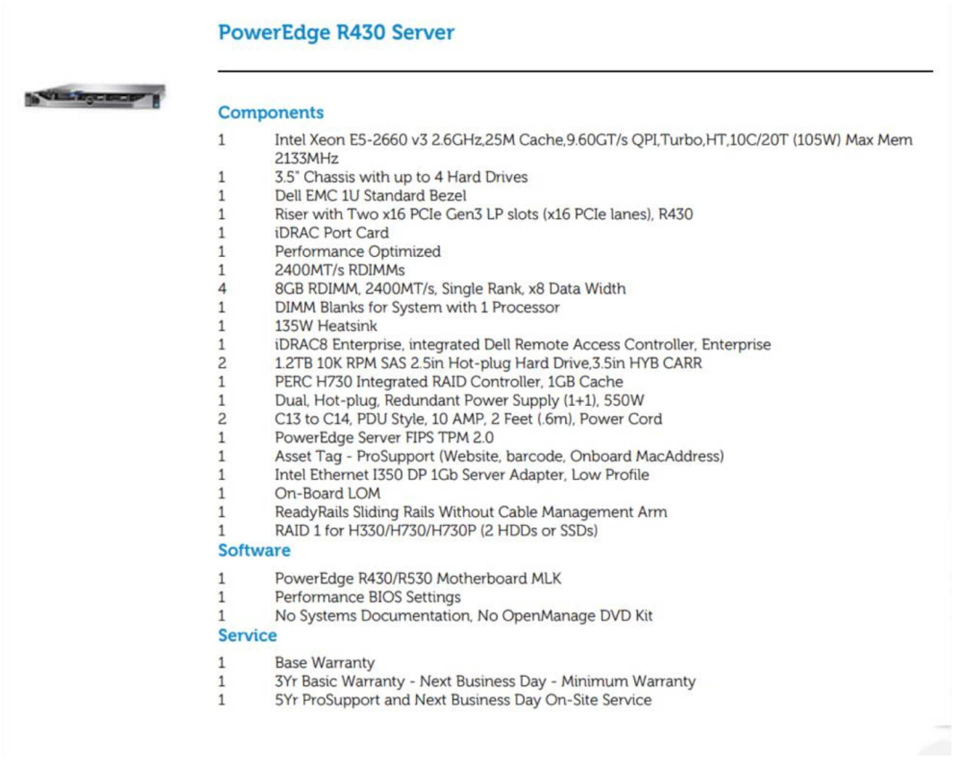
Figure 16: Bristol logical infrastructure configuration

The overall FLAME platform has been deployed at Millennium Square Bristol City Centre on top of 5GUK Test Network. Four compute nodes (one per tower) are deployed at the square.

Resource	**Capacity**	**Availability Constraints**
Compute	4 x OpenStack Compute Node with 15 core available. See Figure 16.	These resources are dedicated to FLAME.
Storage	4 storage nodes with 900GB	These resources are dedicated to FLAME.
Networking	4x EdgeCore SDN switches 4 x Ruckus WiFi access points	These resources are shared across projects.

Table 16: Bristol infrastructure resource specification

Co-funded by the Horizon 2020 Framework Programme of the European Union



PowerEdge R430 Server

Components

1	Intel Xeon E5-2660 v3 2.6GHz,25M Cache,9.60GT/s QPI,Turbo,HT,10C/20T (105W) Max Mem 2133MHz
1	3.5" Chassis with up to 4 Hard Drives
1	Dell EMC 1U Standard Bezel
1	Riser with Two x16 PCIe Gen3 LP slots (x16 PCIe lanes), R430
1	iDRAC Port Card
1	Performance Optimized
1	2400MT/s RDIMMs
4	8GB RDIMM, 2400MT/s, Single Rank, x8 Data Width
1	DIMM Blanks for System with 1 Processor
1	135W Heatsink
1	iDRAC8 Enterprise, integrated Dell Remote Access Controller, Enterprise
2	1.2TB 10K RPM SAS 2.5in Hot-plug Hard Drive,3.5in HYB CARR
1	PERC H730 Integrated RAID Controller, 1GB Cache
1	Dual, Hot-plug, Redundant Power Supply (1+1), 550W
2	C13 to C14, PDU Style, 10 AMP, 2 Feet (.6m), Power Cord
1	PowerEdge Server FIPS TPM 2.0
1	Asset Tag - ProSupport (Website, barcode, Onboard MacAddress)
1	Intel Ethernet I350 DP 1Gb Server Adapter, Low Profile
1	On-Board LOM
1	ReadyRails Sliding Rails Without Cable Management Arm
1	RAID 1 for H330/H730/H730P (2 HDDs or SSDs)

Software

1	PowerEdge R430/R530 Motherboard MLK
1	Performance BIOS Settings
1	No Systems Documentation, No OpenManage DVD Kit

Service

1	Base Warranty
1	3Yr Basic Warranty - Next Business Day - Minimum Warranty
1	5Yr ProSupport and Next Business Day On-Site Service

Figure 17: Bristol compute/storage/network node specification

5.3.1.2 Bristol Production Infrastructure Specification (UOB)

The purpose of the production infrastructures is to support real-life trials and experiments to explore the acceptance, viability and performance of FLAME products in Bristol. The diagram in Figure 18 is a current snapshot showing the locations with mobile edge computing in the four towers and the Ruckus WiFi technology, mirroring the staging environment presented above albeit located in the actual physical deployment location of the Millennium Square in Bristol.

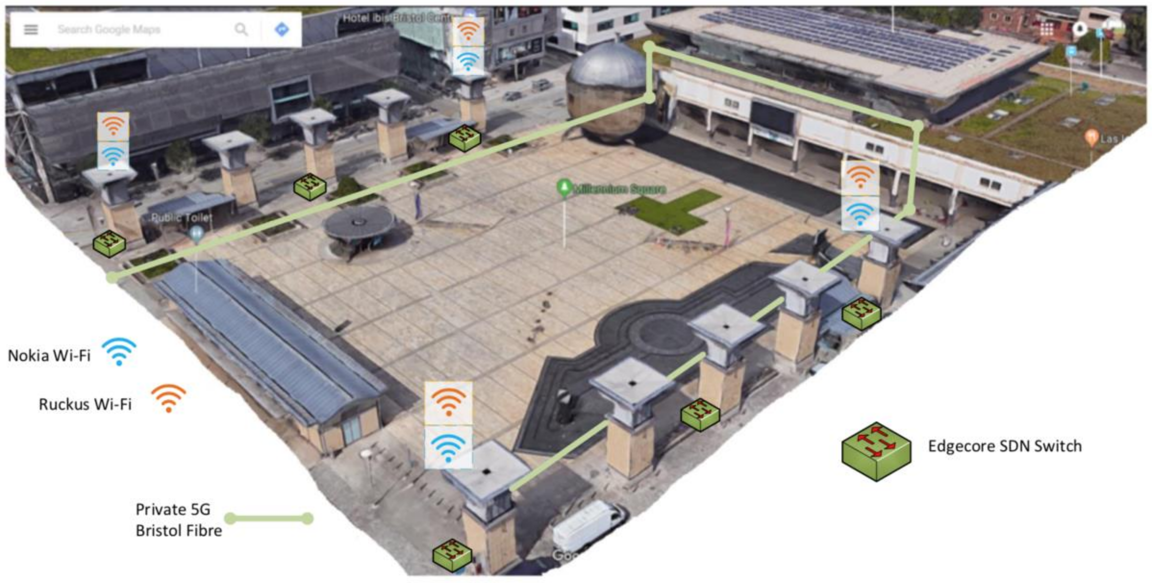


Figure 18: Bristol production infrastructure configuration

Co-funded by the Horizon 2020
Framework Programme of the European Union

5.3.2 Barcelona Infrastructure (i2CAT)

Barcelona Production Infrastructure Specification

The purpose of the infrastructures is to support real-life trials and experiments to explore the acceptance, viability and performance of FLAME products in Barcelona. There exists no separate staging environment with deployment taking place directly into the production environment.

Deployment diagram of the physical hardware

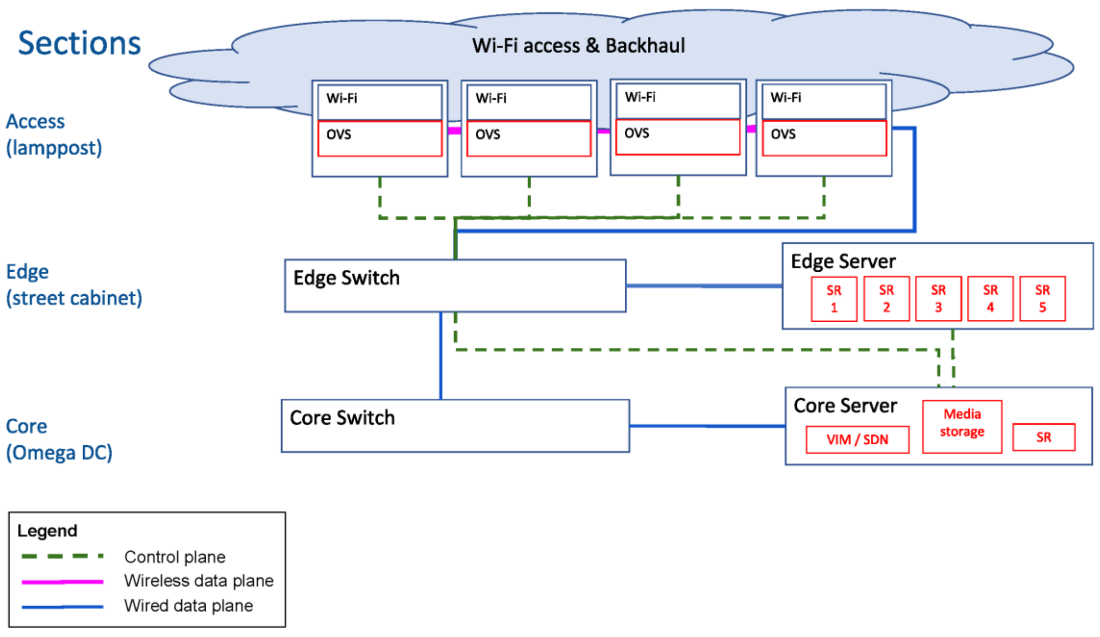


Figure 19: Barcelona production infrastructure configuration

As shown in Figure 19, the lamp posts are connected over a wireless backhaul, forming the experimenter's data plane connection between the far edge (RAN) and any other element deployed in FLAME. Also, each lamp post has a dedicated fibre connection which is used to hook them up to a dedicated SR hosted in the edge cabinet. This way we emulate each lamp post having a dedicated SR.

Resource	Capacity
Compute	Cloud: i2CAT cloud resources subject to the experiment requirements and cloud resource availability. At minimum two medium size servers. Core DC: 2 servers containing 2 x CPUs (6 cores @ 2.4 GHz and 6 cores @ 3.5 GHz) with Hyperthreading enabled, offering a total of 24 vCores, and 2 x 96 GB (total of 192 GB) RAM memory. Edge (cabinet): a server containing 1 CPU (12 cores @ 2.10 GHz) with Hyperthreading enabled, offering a total of 24 vCores, and 128 GB RAM memory.
Storage	Core: Each server provides 1.9 TB of disk space (SSD) (total of 3.8 TB). Edge (cabinet): It provides 1.8 TB of disk space (SSD).
Networking	Cloud: 3xPronto TN3290 switches; 10 Gbps wired connectivity Edge (cabinet): 10 Gbps wired connectivity On-street equipment: 4 WLAN devices installed in lamp posts

Co-funded by the Horizon 2020
Framework Programme of the European Union

	• Access and Multi-hop backhaul network: IEEE 802.11ac • 10 Gbps wired connectivity to the cabinet • 1 Gbps wired connectivity between lamp posts and edge cabinet

Table 17: Barcelona production infrastructure resource specification

The production infrastructure will be deployed in Barcelona during the first stage of the project as a replication of Bristol FLAME infrastructure. Plans to extend the hardware infrastructure in Barcelona is left beyond the scope of the project.

Co-funded by the Horizon 2020
Framework Programme of the European Union

6 CONCLUSIONS

This report has described the updated technical roadmap for the FLAME project. The report describes a set of interdependent software products that together will provide a ground-breaking media service delivery platform exploiting the benefits of highly-distributed software-defined infrastructures.

The Platform product has been elaborated in detail as one key output of the project. A feature analysis is described for the FLAME platform covering Orchestrator, SF Endpoint Management and Control, SF Routing and Cross-Layer Management and Control. The Infrastructure Products are described to provide the target deployment environment for products. The media service product roadmap is also included, identifying the foundation services that will form part of the FLAME offering. The relationship with experimentation and project KPIs is elaborated to explicitly show how features of the Platform address the key objectives of experimentation independent of physical location and reducing the time to perform experiments.

A systems integration and testing plan is defined detailing the DevOps processes including multi-project structure, development workflows, and testing tools. A software-based integration infrastructure is specified that offers the ability to conduct integration tests that cover the expected features of the platform, which are in turn representative of the production infrastructure. This integration infrastructure allows for concurrent integration tests if needed for the different integration activities expected within the project.

Co-funded by the Horizon 2020
Framework Programme of the European Union

REFERENCES

[ETSINFV] ETSI GS NFV-SOL 004 Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification, available at http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.03.01_60/gs_nfv-sol004v020301p.pdf

[FLAME-D3.1] D3.1 FMI Vision Use Cases and Scenarios v1.0, available at https://www.ict-flame.eu/download/d3-1-fmi-vision-use-cases-and-scenarios-v1-0/

[FLAME-D3.2] D3.2 Experimental Methodology for Urban-Scale Media Trials v1.0, available at https://www.ict-flame.eu/download/d3-2-experimental-methodology-for-urban-scale-media-trials-v1-0/

[FLAME-D3.3] D3.3 FLAME Platform Architecture and Infrastructure Specification v1.0, available at https://www.ict-flame.eu/download/d3-3-flame-platform-architecture-infrastructure-specification-v1-0/

[MANO] Open Source Mano, available at http://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano

Co-funded by the Horizon 2020
Framework Programme of the European Union