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

D3.10: FLAME Platform Architecture and Infrastructure Specification V2

Dirk Trossen, Sebastian Robitzsch, Kay Hänsge (InterDigital Europe) | 15/12/2018

Michael Boniface (University of Southampton) | 15/12/2018

D3.10 provides the second version of the FLAME platform architecture and infrastructure specification. It outlines therefore the blueprint that forms the basis for further specification but more importantly the implementation work towards a deployable FLAME platform within the infrastructures of Bristol and Barcelona. D3.10 builds on V1, published in D3.4, focussing on the key updates to that first version.

Work package	WP 3
Task	Task 3.4
Due date	31/12/2018
Submission date	11/01/2019
Deliverable lead	InterDigital Europe
Version	1.0
Authors	Dirk Trossen, Michael Boniface, Kay Haensge, Sebastian Robitzsch
Reviewers	Aloizio Pereira da Silva
Keywords	NFV, SDN, ETSI MANO, FLIPS, CLMC, orchestration

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	19/09/2017	First structure	Dirk Trossen
V0.2	17/10/2018	Added content to sub-components, changed structure, revised Section 2 and 3, added infra integration to Section 6	Kay Haensge, Dirk Trossen, Sebastian Robitzsch
V0.3	19/12/2018	Page Header and Footer update Minor typos and reference issues updated	Kay Haensge
V0.4	20/12/2018	Updated FLAME Platform architecture to elaborate the CLMC components. Added content for CLMC subcomponents Reviewed document throughout	Michael Boniface
V0.5	07/01/2019	Included reviews	Dirk Trossen
V1.0	11/01/2019	Final quality assurance	Michael Boniface

DISCLAIMER

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

This document reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

Project co-funded by the European Commission in the H2020 Programme

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

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	

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

EXECUTIVE SUMMARY

This deliverable D3.10 provides a second version of the specifications for the FLAME platform and infrastructure, building upon the original blueprint provided in [D3.3] and the experiences and revisions realized in the ongoing development of platform components and deployment at 5G infrastructures in Bristol and Barcelona.

The main outcome of these revisions is a set of revised specifications. Those provide insights into the main FLAME platform components, their relation to media service components provided by media service providers on top of the FLAME platform, and the interaction with the infrastructure provided by stakeholders such as Bristol or Barcelona city council.

The methodology driving the revised specification work focusses on the main changes to the platform architecture with aspects of component-level design, revised inter-component interfaces, intra-component design and infrastructure integration.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

TABLE OF CONTENTS

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

LIST OF FIGURES

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

LIST OF TABLES

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
CLMC	Cross-layer management and control
eSBA	enhanced Service-based Architecture
FLIPS	Flexible IP Services
FMI	Future Media Internet
FQDN	Fully qualified domain name
HTTP	Hypertext transfer protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
MANO	Management and Orchestration
MSP	Media Service Provider
NFV	Network Functions Virtualization
PCE	Path Computation Element
REST	Representational State Transfer
SDN	Software-defined Networking
SE	Service Encapsulation
SF	Service Function
SFC	Service Function Chaining
SFE	Service Function Endpoint
SFF	Service Function Forwarder
SLA	Service Level Agreement
SR	Service Router
TCP	Transmission Control Protocol
TOSCA	Topology and Orchestration Specification for Cloud Applications
tSFF	Transport Derived Service Function Forwarder
YAML	YAML Ain't Markup Language

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

1 INTRODUCTION

This deliverable D3.10 provides a revision or update to the FLAME architecture specification initially provided in [D3.3]. Given the time of developing this specification within the project lifecycle, this revised specification captures the insights obtained from developing and deploying the FLAME platform in real infrastructures and performing initial functional tests on those infrastructures. This understanding will be further developed through experiments and user-centric trials to be conducted from 2019-2020. This deliverable focuses on the architecture, its components interactions as well as the sub-components as being developed in the current prototype. Aspects of the business architecture, the relationships of stakeholders as well as use cases and their derived requirements remain unchanged. We therefore refer the reader to [D3.3] for this information.

With this in mind, we start with our revision to the high-level architecture in Section 2, followed by the main relationships in the overall system, and the various scopes of orchestrations that exist in our system. Section 3 then provides an update of the component-level interfaces, while Section 4 presents an overview of the sub-components as new information compared to D3.3. In Section 5, we follow on the service function chains, as introduced in D3.3, for major interactions with and within the FLAME platform, while providing our approach to infrastructure integration in Section 6.

The specifications in this deliverable are considered very stable and form the basis for the beta release of the FLAME platform in its overall integration lifecycle, as will be detailed in D3.7. This beta release will form the basis for our validation as well as experiments and trials in 2019-2020. Nonetheless, we have planned another update for the release candidate of the FLAME platform towards the end of the project with final revisions based on insights from our experiments.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

2 PLATFORM ARCHITECTURE

This section provides a component-level overview as an introduction for the component-level interfaces and sub-components presented in Section 3 and 4, respectively. It also outlines the relationships in the overall platform and the scopes of orchestration of resources and services that exist in the overall system.

2.1 COMPONENT-LEVEL OVERVIEW

Figure 1 shows the FLAME platform architecture, operating on top of an infrastructure such as the one provided in our deployments in Bristol and Barcelona. In this figure, we focus on the main layers of the overall architecture, including the FLAME platform, while showing main sub-components for explanation of the overall workings. Detailed descriptions on the internal functions of the various layers are deferred here to Section 4.

At the very bottom, we assume the existence of the **infrastructure (provider)**, exposing an ETSI MANO compliant interface [NFV] to the FLAME platform for resource management at the *wholesale level*, i.e. the FLAME platform is reserving platform resources in the compute, storage and networking domain. We assume such infrastructure resource exposure provides the ability to reserve resources for the FLAME platform as part of a longer-lived relationship between FLAME platform provider and infrastructure provider, realized with an infrastructure slice. At the current level of our infrastructure integration (see Section 6), we provide support for OpenStack [Openstack] and OpenFlow [OpenFlow] managed compute, storage and communication resources.

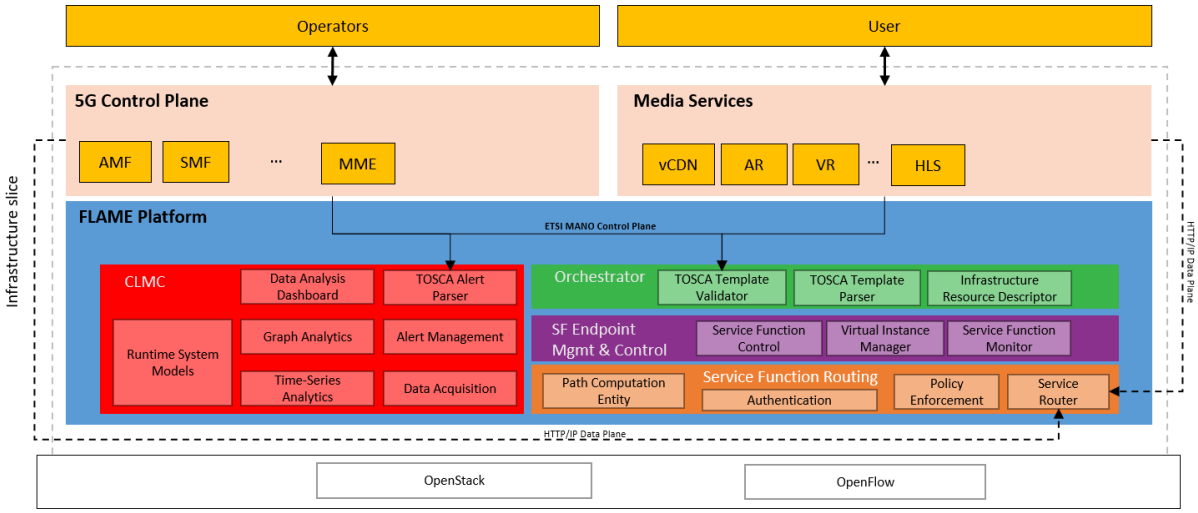

Figure 1: FLAME Platform Architecture

Resources of the infrastructure provided to the FLAME platform are in turn provided as *retail resources* to the media services at the top of the platform through management interfaces exposed to media services. ***In other words, the FLAME platform orchestrates the deployment of media components as well as internal service functions.*** In addition, a monitoring interface allows for information exchange between media services and FLAME platform, which in turn will drive the decisions taken for the management input via the former interface. This combination of management and monitoring interfaces effectively provide the **experiment API** towards media services, allowing for defining and placing the compute, storage and network resources for the specific tests in the experiment, along

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

with monitoring and alert configuration necessary to trigger service function policy actions and understand performance in relation to resource specifications. The experiment API is complemented at the data plane through standard HTTP/IP data plane interfaces of the existing Internet through which media services realize their functionality.

The management interfaces allow for initiating the orchestration of (retail) resources by a media service provider. We see these interfaces and therefore the dialogue between media services and the FLAME platform evolving over time. In an initial realization, we see media services heavily relying on the ETSI MANO compliant interface, providing details on compute, storage and network resources being utilized in their specific experiment deployment. However, empirical insights obtained from the initial experiments will drive the evolution of the FLAME-specific (second) management interface towards one that provides a selection of orchestration templates for specific use cases. Such templates will therefore remove the need for explicitly defining each orchestration template from scratch, while utilizing the ETSI MANO compliant interface to utilize this evolving template knowledge towards the FLAME platform. This evolution is based on the evolving knowledge through our initial use cases on how to best accommodate demand and supply for specific media services. Through our open calls, this richness of knowledge is meant to expand, e.g., through rich control policies that allows for matching runtime-level demand and supply measurements and feeding them into the control elements of the FLAME platform. Again, instead of needing the media service provider to define all those details through own templates and setups, we see an enriched repository of configurations that is being utilized by the FLAME-specific management interface between media services and FLAME platform rather than utilizing the ETSI MANO compliant variant. *Ultimately, we see the relationship between media services and platform as being realized by high-level KPI-driven service-level agreements (SLAs), expressed through the FLAME-specific management interface and realized through an evolving CLMC on the platform side.*

As indicated in Figure 1, we consider the realization of the **media services** outside the scope of the FLAME platform itself. We do assume, however, that a media service is realized through a set of media components, each communicating with each other through an HTTP/IP-compliant data plane interface. Media services will utilize the management and monitoring interfaces to the FLAME platform (see below) to facilitate and enable the deployment of those media components through the FLAME platform, i.e. media components are (computing and storage) resources from a FLAME platform perspective. Towards the end user, we see media components utilizing service-specific interfaces and interaction methods. Media components are implemented utilizing platform resources, such as servers, connectivity components (e.g., switches) and others, while also utilizing resources outside of the scope of FLAME, such as end user devices and Internet-of-thing components. *With that in mind, we position FLAME as a distributed programmable resource platform, which can be used by media services for the fulfilment of a desired experience towards users.* It is up to the policy of the FLAME platform provider if resources are provided exclusively to a media service provider or from a shared resource pool. In the latter case, experiments conducted by the media service providers could run concurrently in the system, while the former case provides an exclusive access to the resources for a single media service provider. The governance and specificity of the experiments as well as deployment will likely drive these policies. For instance, exclusive usage of geographically defined resources might make certain media service deployments mutually exclusive since appropriate resources for another deployment are simply not available.

In addition to media services, we position the FLAME platform as supporting the realization of **5G control plane** services, such as the Session Management Function (SMF), for vertical control planes as envisioned for, e.g., vehicular or industrial scenarios. The applicability of the FLAME platform for this set of use cases has been successfully demonstrated in joint demonstrations with Deutsche Telekom

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

and is enabled through the alignment of the FLAME platform design with the evolving enhanced Service-based Architecture (eSBA) specifications in 3GPP Release 16.

The **orchestrator** component of the FLAME platform interfaces with the infrastructure resource management, through the aforementioned ETSI MANO compliant interface. The orchestrator manages the compute/storage/network resources at the retail level, i.e. towards the media service provider, while it utilizes the surrogate policy control interface towards the **Service Function (SF) endpoint management and control component** to realize the orchestration-level management policies as well as to set suitable shorter-term control policies for (surrogate) service function endpoints. For the realization of the configured service function endpoint policies, the service function endpoint management and control layer utilizes the FQDN registration interface to control the registration and deregistration of the service endpoints towards the **service function routing** component. With this, the 'visibility' of said FQDN towards service requests being routed can be controlled. The service routing layer, in turn, will use the OpenFlow interface (e.g., via suitable platforms such as ODL [ODL]) to suitably configure the switching fabric of the underlying infrastructure.

Particular consideration is given in our platform to the gathering of information across various layers, realized by the **Cross-Layer Management and Control** (CLMC) component. While such data is useful and needed for control-level decisions, such as the activation of service endpoints, it also provides a rich pool of data for media service providers to develop insights into resource specifications, adjusting crucial longer-term strategies such as those for content placement or media adaptation, and dimensioning SLAs expected to govern B2B and B2C relationships. The CLMC brings together time-series and graph analytics to understand demand, resourcing and performance properties of media service function chains deployed within the FLAME platform. Time-series measurements related to individual service functions and links are aggregated and then related through hierarchical topologies of infrastructure, service function endpoints and overall service function chain. For a given orchestration, the infrastructure and service function nodes remain largely static whilst the deployment and state of surrogates varies throughout the lifecycle of the service function chain according to demand and policies. The CLMC can continuously produce temporal graphs that integrate state, performance and resource usage of system elements that can deliver insights such as end-to-end delay for protocols at OSI L6 such as HTTP. The time-series measurements are acquired from existing monitoring frameworks provided by the individual sub-systems (such as those provided directly by switching platforms, obtained through the aforementioned OpenFlow interface towards the underlying infrastructure) or monitoring frameworks offered by media components themselves. The aim is not to replace existing systems but provide a cross-layer knowledge model that can drive QoS-orientated of both content and service configuration.

In Section 3, we elaborate in more detail on the component-level interfaces outlined in Figure 1, including to present an UML-based version. Before that, we introduce the service abstraction being used for the platform-internal interactions as well as discuss the management and control aspects relevant to the FLAME platform.

2.2 SERVICE ABSTRACTION AND RELATIONSHIPS

This section discusses the main relationships of actors in the overall system, focussing on the service providers, platform operator and infrastructure provider. Figure 2 captures those relationships. The **business logic** of a given media service is captured in a (orchestration) *template* that captures the chain of services being used within said overall media service. One or more of such templates are then orchestrated by the **FLAME Platform**'s *orchestrator* into one or more deployed *Service Function Chains* (SFCs), which in turn captures the interaction of the one or more *Service Functions* (SFs) that represent

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

a specific **service** in the overall chain, such as a CDN or transcoding service. The interaction of services functions within a service function chain is facilitated by the FLAME platform's *service function routing* capability. The one-to-one relationship illustrates the joint usage of user plane resources, e.g., switch buffers, bandwidth, etc, across all deployed service function chains within a single deployment of the overall FLAME platform.

A service function can be realized by one or more Service Function Endpoints (SFEs), which in turn is realized through a virtualized resource, such as a container or virtual machine, whose capabilities are being specified in the orchestration template. The SFE is instantiated using a packaged image, which is controlled by the FLAME platform's SFEMC component (see Figure 1 and Section 4.1.2) and executed on a *physical instance* of the underlying **infrastructure**, which in turn hosts one or more such physical instances a part of the overall deployment in a specific site. If more than one infrastructure is being used across, e.g., two different sites, each exposing their set of physical instances to the FLAME platform, an *infrastructure exchange* can be used for exposing a single resource pool to the orchestrator of the FLAME platform.

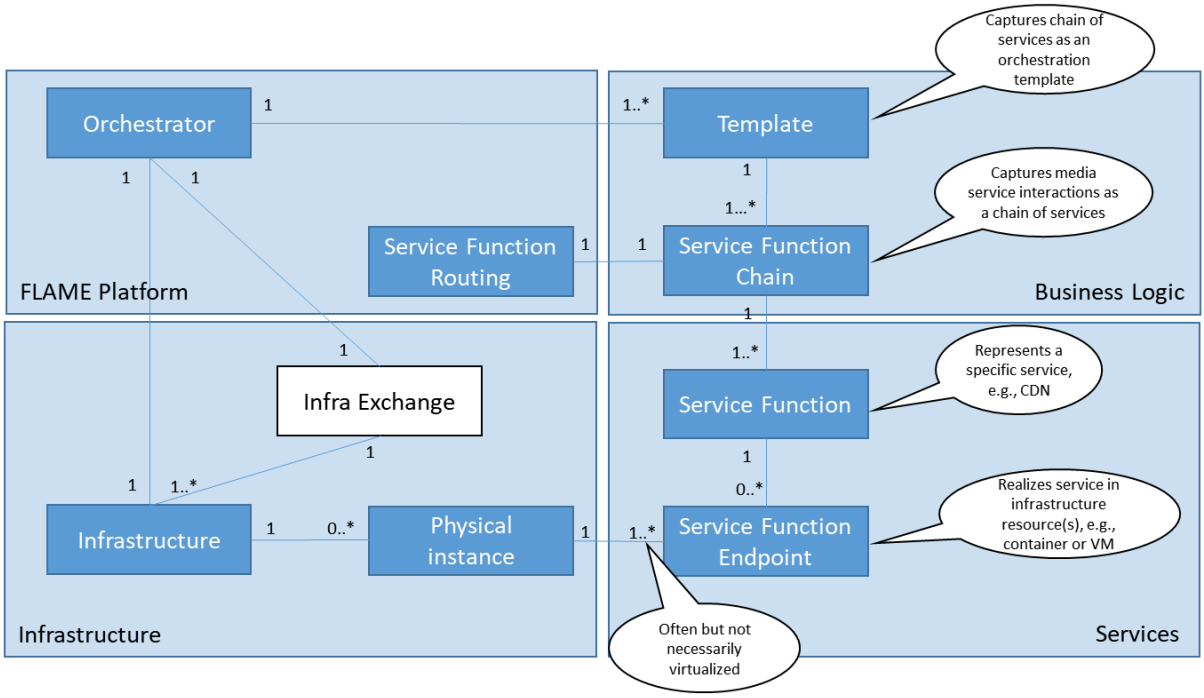

Figure 2: Main Relationships

2.3 SCOPES OF ORCHESTRATION

Orchestrating an overall deployment of a media service in FLAME involves various level of orchestration, as shown in Figure 3. Over a rather longer timeframe of likely hours, days or longer, the infrastructure provider orchestrates the provisioning of a **network slice** based on a long-term wholesale relationship between infrastructure and platform provider.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

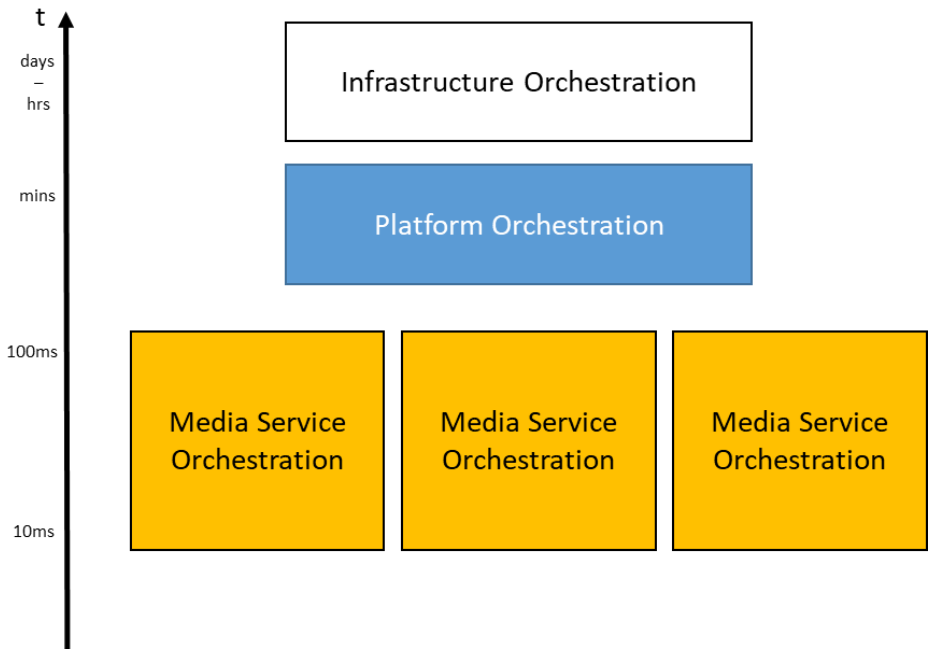

Figure 3: Time Scales of Orchestration

This slice represents the overall resource pool that the platform provider is able to utilize for deploying one or more instances of the FLAME platform. We envision the deployment of such platform instances to be a matter of minutes rather than hours or more, given the existing tenant relationship with the infrastructure, as manifested in the associated slice of resources. Within an orchestrated platform, media services are eventually the fastest (and closest to end users) level of orchestration with deployment times in the seconds or below timeframe by utilizing container techniques for deploying media services.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

3 COMPONENT INTERFACES

This section presents the main component interfaces of the architecture in Figure 1. We will reference these interfaces throughout the SFCs presented in Section 4. For this, we transform the component architecture in Figure 1 into an UML version, presented in Figure 4.

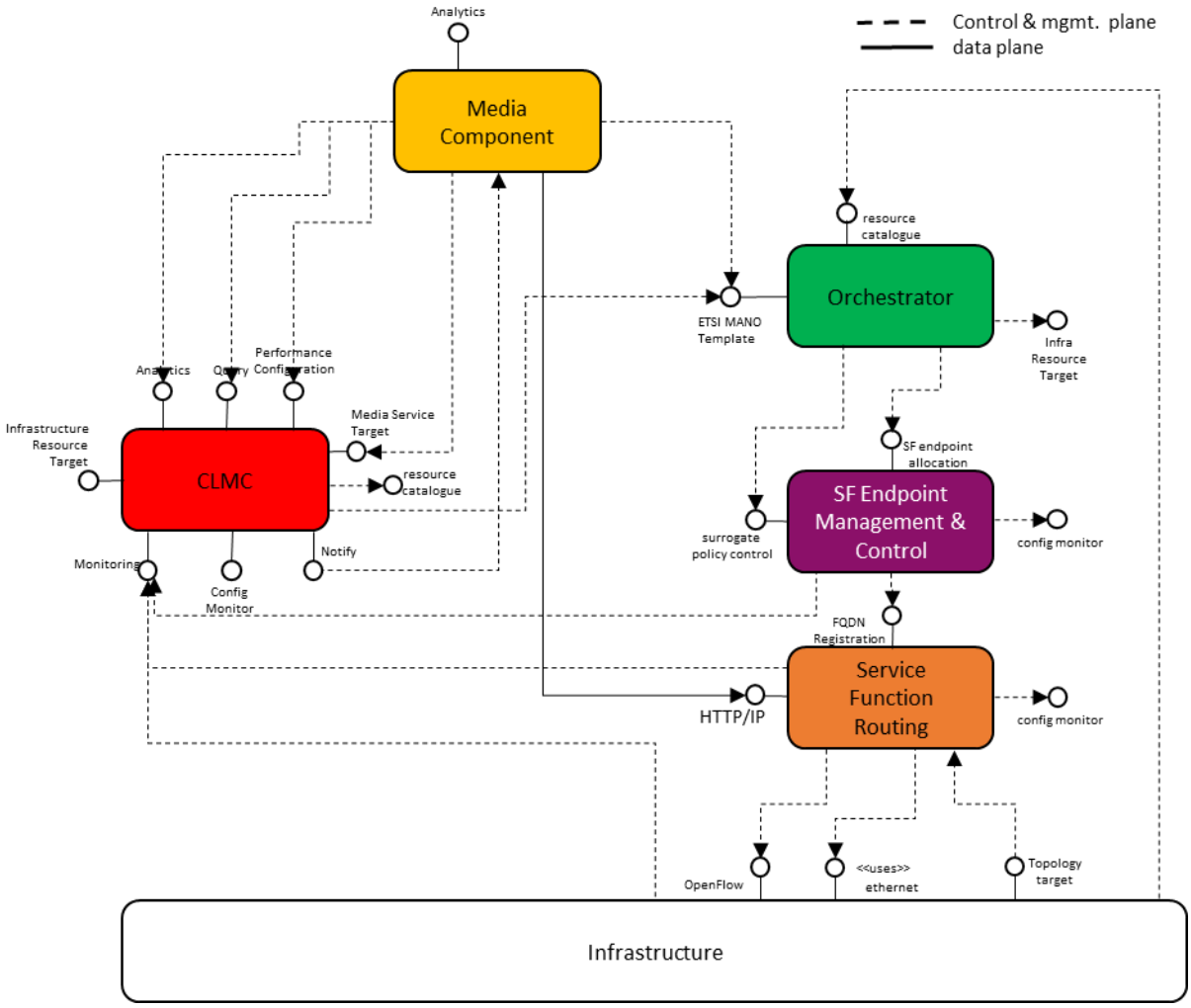

Figure 4: FLAME Platform Architecture – UML version

The following sub-sections outlines the interfaces of each of the components in more detail.

3.1.1 Media Component

A media service is comprised of one or more media components, implementing specific sub-functions of the overall service. Although FLAME does not prescribe the specific abstractions or methods being used for the decomposition nor any specific capabilities being realized in the media component, we do assume data plane traffic of the media component to utilize HTTP or other IP-based protocols, see Figure 4. In addition, we expect media components to expose an optional ***Analytics*** interface, providing access to media-related analytics, such as frame rates, drop rates for video content, usage statistics, and alike. For this, there are a number of widely accepted standard KPI definitions, while we expect an HTTP/REST API realization for this interface.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

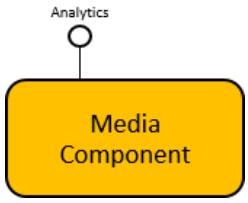

Figure 5: Media Component

No.	Interface	Purpose	Outcome	Relevant Specifications
M1	Analytics	Retrieve media-related statistics and analytics performed by the media component	Persistent set of media statistics and analytics data	Specific to media components

Table 1: Media component Interfaces

3.1.2 Orchestrator

Orchestration is concerned with mapping a desired resource configuration onto the resource constraints as defined by the availability of resources at the infrastructure level. The general capabilities of the interfaces are shown in Figure 6 and further elaborated in Table 2. The further decomposition of this component is presented in Sections 5.1.1 and 5.1.2.

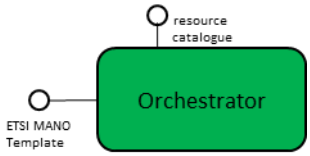

Figure 6: Orchestrator Component

No.	Interface	Purpose	Outcome	Relevant Specifications
O1	ETSI MANO	Provide a template-level description of desired resource configurations for a media service to be deployed. Access control to initiate orchestration is included in this interface.	Persistent set of service function endpoints and their topological relationship	ETSI MANO
O2	Resource catalogue	Provide a template-level description of available resources (and their constraints) at the infrastructure level in which the media service template needs to operate. Furthermore, provide suitable infrastructure-level resource handles allowing for the management of the resource instances.	Resource catalogue derived from the wholesale infrastructure (slice) template as well as infra-level resource handles for management	ETSI MANO

Table 2: Orchestration Interfaces

Templates, both at the O1 and O2 interfaces, are based on ongoing specification work in ETSI MANO [NFVMANO]. The infrastructure provides a resource catalogue in the form of such template, referred to as the **slice template**. This template provides the constraining framework, representing the wholesale resources, within which the set of all media service templates need to operate. A specific **media service template** is provided at the O1 interface to the orchestration component from either

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

the CLMC component or the media component directly. Section 5.1.1 will elaborate on the different usage for these two types of access to the orchestration template.

The resource catalogue, provided by the O2 interface is utilized by the CLMC component for capacity management, i.e., for planning the dimensioning of the various media service templates within the constraints set out by the slice template provided by the O2 interface. The infrastructure-level resource handles provided by the O2 interface are utilized by the orchestrator to initiate the suitable service endpoint management and control by providing such information via the SFEMC1, see below.

3.1.3 Service Function Endpoint Management & Control

This component is concerned with the management and control decisions pertaining to SF endpoints. The management decisions concern the placement and instantiation of suitable virtual instances, while control decisions concern the visibility of such SF endpoints in the overall service path, e.g., realizing use cases for alternative playout points. The general capabilities of the interfaces are shown in Figure 7 and further elaborated in Table 3. The further decomposition of this component is presented in Section 5.1.2.

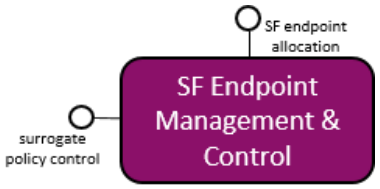

Figure 7: Service Function Endpoint Management & Control Component

No.	Interface	Purpose	Outcome	Relevant Specifications
SFEMC1	SF endpoint allocation	Provides virtual instances of SF endpoints based on received orchestration information, utilizing the received infrastructure-level resource handles.	Persistent set of virtual instance handles to SF endpoints for future management and control decisions	ETSI MANO
SFEMC2	Surrogate policy control	Provides suitable orchestration information for the control of SF endpoint states from PLACED over BOOTED to CONNECTED (see Section 5.1.2 for more information on these states)	Persistent set of SF endpoint specific control policies being monitored and executed by the component	ETSI MANO

Table 3: Service Function Endpoint Management & Control Interfaces

Via the SFEMC1 interface, suitable orchestration information is received based on realizations of the ETSI MANO specifications. The component utilizes the infrastructure-level resource handles received via this interface to realize the functionality of a virtual instance manager [NFVMANO], placing suitable SF endpoints and instantiating the necessary virtualized resources. Via the SFEMC2 interface, control policies are provided, derived either from existing or evolving ETSI MANO service templates, that define the SF endpoint specific control behaviour within the overall media service deployment. While current ETSI MANO compliant templates include little of such control policy, FLAME will investigate other control policy approaches with the final aim to integrate such policies into a coherent overall template description for the media service.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018
Co-funded by the Horizon 2020 Framework Programme of the European Union

3.1.4 Service Function Routing

The Service Function Routing component provides a number of control plane as well as a data plane interface for the purpose of efficiently routing service requests of any SF endpoint to one or more others. The general capabilities of the interfaces are shown in Figure 8 and further elaborated in Table 4. The further decomposition of this component is presented in Section 5.1.4.

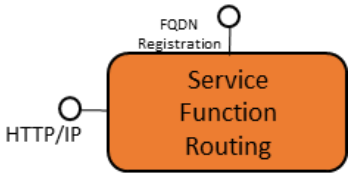

Figure 8: Service Function Routing Component

No.	Interface	Purpose	Outcome	Relevant Specifications
SFR1	FQDN registration	Registration of an SF endpoint with the service routing	Persistent set of internal state information for suitably forwarding any request to the registered FQDN-based SF endpoint	Internal FLIPS specifications with the intention of IETF standardization
SFR2	HTTP/IP	Transmission of IP-based protocols at the HTTP or IP level	Persistent set of internal path information suitable for forwarding the request in the network	IETF specifications for HTTP and IP-based protocols

Table 4: Service Function Routing Interfaces

The data plane interface SFR2 is based on existing protocol specifications standardized by the IETF for protocols such as HTTP, TCP, SCTP or other IP-based protocols. This interface differentiates HTTP as a specific application protocol due to its proliferation in the Internet, with about 60% or more traffic being based on HTTP. HTTP-level SFEs, i.e. represented by a fully qualified domain name (FQDN), are registered through the SFR1 interface and will become 'visible' in the media service path as a result, i.e. any future service request to said FQDN will possibly be routed to the registered SF endpoint.

For the realization of this component, FLAME has selected the FLIPS solution provided by InterDigital, which comes with FLIPS-specific protocol specifications (see Section 4.1.3.6 for an overview of those protocols). However, the component could be realized via standard IP technologies for which existing DNS (for SFR1) specifications would be utilized.

3.1.5 CLMC

The CLMC component offers a series interfaces supporting the monitoring, measurement and analysis of performance, in addition to configuration of processes supporting the integration and organisation of data for analytics. The general capabilities of the interfaces are shown in Figure 9 and further elaborated in Table 5.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

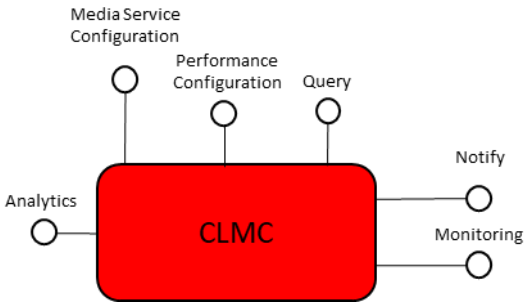

Figure 9: CLMC Component

No.	Interface	Purpose	Outcome	Relevant Specifications
CLMC1	Media Service Configuration	Used to set a Resource Configuration for a Media Service including target allocations and constraints.	Persistent set of infrastructure resource configuration change events	TMF640 Activation and Configuration API, ETSI Mano VFN Configuration
CLMC2	Performance Configuration	Used to configure monitoring and measurement processes for performance assessment and analysis of a Media Service. Configuration will include setting up processes for multidimensional analysis of KPIs from monitoring data including aggregation, classification and stakeholder viewpoints.	Performance monitoring and measurement model for a media service including runtime notification rules.	TM Forum TMF-628, ETSI Mano
CLMC3	Query	Used to retrieve KPI measurements for a given set of criteria	KPI measurement	TM Forum TMF-628, ETSI Mano. Mainly proprietary options for the information model and query language
CLMC4	Notify	Used to notify subscribers of performance measurements through event-trigger-action rules such as current KPI performance	Notifications of configured performance events	Many pub/sub specifications
CLMC5	Analytics	Used by developers of interactive media systems or sub-system to understand performance of service function chains through exploration or requests, responses and workload	Knowledge on how to best achieve KPIs through correlated data, etc.	Mainly proprietary options for the information model and query language
	Monitoring			
CLMC6a	Monitoring (infrastructure)	Used to monitor infrastructure resources (compute, storage and network) including resource allocation, resource usage and resource performance metrics. Resource monitoring data is high frequency (uSec-mSec) data. These data may be aggregated and resampled at points of publication to reduce the frequency and quantity of monitoring data through the network.	Persistent set of transactional data describing the state of infrastructure resources over time related to a known context within a Resource Configuration	Pub/sub protocol supporting a range of collection and sampling models
CLMC6b	Monitoring (Service Function Configuration)	Used to monitor changes in Service Configuration occurring in response to demand	Persistent set of Service Function configuration change events	TMF640 Activation and Configuration API, ETSI Mano VFN Configuration
CLMC6c	Monitoring (Service Function)	Used to monitor SF including service request, service response and service performance. Service monitoring data is high frequency (mSec) data. These data	Persistent set of transaction data describing the state of a Service Function.	Pub/sub protocol supporting a range of collection and sampling models

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

		may be aggregated and resampled at points of publication to reduce the frequency and quantity of monitoring data through the network.		

Table 5: Performance Management Interfaces

There are different standards and specifications relevant to Performance Management. For the ***Performance Configuration***, the TM Forum Performance Management API (TMF-628) [TM17] provides a restful API for managing the lifecycle of production and collection of performance measurements. ETSI Mano includes Network Service, VNF and Resources performance management operations to measure performance synchronously "Get Performance Measurement Results" and asynchronously "Notify". Although it should be noted that the ETSI Mano specification is abstract and architectural and not at the level of an API with message formats. ETSI and TM Forum separate processes for measuring performance KPIs (Performance Management) from contractual processes provided by SLA Management with the former underpinning the latter. For ***Media Service Resource Configuration Monitoring***, TMF640 Activation and Configuration API provides a more general version of the VFN Configuration operations within ETSI Mano. It is likely that the same API can be used to monitor configuration objects and changes associated with both infrastructure resources and service functions. For ***Resource and Service Monitoring*** there are no widely accepted standards but we expect to adopt a pub/sub protocol supporting a range of data acquisition options allowing for resampling, aggregation and buffering of time-series monitoring data.

For the knowledge model, ***KPI Classifications*** have been formally defined by ITU-T, ITIL and many others, covering all aspects of business operations. We focus on KPIs relevant to Service Quality (Response Time, Throughput, Setup, Availability, etc.) and Operational Efficiency, as these are the KPIs most influenced by the Platform's capabilities. A taxonomy of key KPIs will be selected and made available to Platform users when assessing the performance of different interactive media systems and their configurations. Although, from an architectural perspective the monitoring and performance model is generic to any service and selection of KPIs are considered a domain specific specialisation. For ***Multi-Dimensional Modelling and Queries,*** there are no standard approaches to multi-dimensional analysis of data, however recent advances in graph databases and analytics is allowing for powerful connections and patterns between different data sets to be explored, complimenting and often replacing the concept of JOINS. The CLMC utilises a combination of time-series analytics and graph analytics to understand the behaviour and relationships between system components across layers of infrastructure, platform and media services.

3.1.6 Infrastructure

Our view on the infrastructure is outlined in more detail in [D3.3]. From an interface point of view, we focus on the ability to configure forwarding capabilities and utilize the data plane of the infrastructure. These capabilities are outlined in Figure 10 and further elaborated in Table 6.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

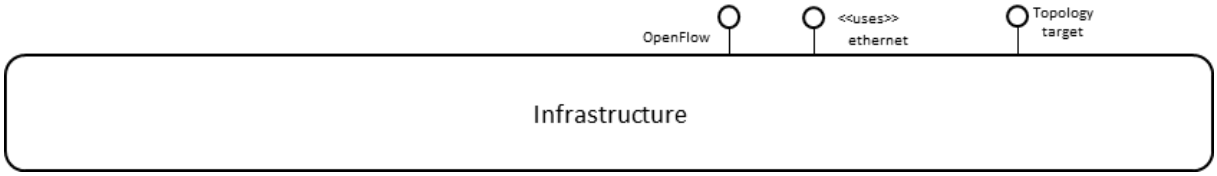

Figure 10: Infrastructure Component

No.	Interface	Purpose	Outcome	Relevant Specifications
I1	OpenFlow	Provide capability to define switch-specific forwarding rules compliant with OpenFlow specifications and associated with the FLAME-specific infrastructure slice	Persistent set of forwarding rules per SDN switch	OpenFlow
I2	Ethernet	Provide data plane access at the Ethernet framing level		Relevant IEEE Ethernet standards
I3	Topology Target	Provide an interface to the SFR component to retrieve suitable data plane topology information to formulate the suitable switch-specific forwarding rules being provided in turn via the I1 interface. The exact nature of this interface depends on the one provided by specific SDN controllers and SDN platforms, such as OpenDayLight.	Topology information	OpenFlow

Table 6: Infrastructure Interfaces

As outlined in previous sections, we assume the infrastructure resources to be provided through the concept of a slice. Within such slice, forwarding rules can be associated to the specific SDN-abstracted forwarding switches associated to the slice. The I1 interface provides such rules to the infrastructure, which in turn will suitably configure the switches. Such configuration is based on topology information provided to the SFR component via the I3 interface, where the nature of the interface will depend on the specific SDN controller platform being utilized.

Any translation onto legacy components, provided through the SDN abstraction to the FLAME platform, is seen as being part of the infrastructure. At the data plane level, a simple Ethernet abstraction of the data plane is assumed at this stage. Other Layer 2 technologies can be supported by extending the supported framing at this interface. However, this is completely subject to the availability of the source code of the firmware and knowledge on the supported framing protocols. Commonly this is not provided by network device vendors.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

4 SUB-COMPONENTS

4.1.1 Orchestrator

The sub-components of the Orchestration are depicted in Figure 11. It consists of a TOSCA Template Validator, a TOSCA Template Parser and an Infrastructure Resource Descriptor.

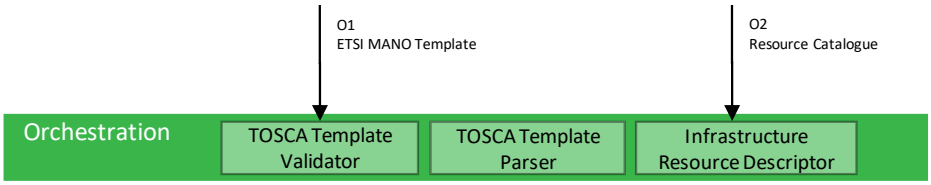

Figure 11: Sub-Components of the Orchestrator

4.1.1.1 TOSCA Template Validator

The Validator performs a validation at syntactical and semantical level against the TOSCA NVF and FLAME specifications. The Validator is also the endpoint of the O1 interface and allows a media service provider to upload and modify a resource specification document. This document describes the SFC with its SFs and its lifecycle management. The upload may be performed via a HTTP/REST call using the POST method or uploading the resource specification file via a form on a WebGUI. The content of theses resource specifications has to be structured in YAML syntax.

4.1.1.2 TOSCA Template Parser

The Parser performs a mapping into manageable data objects and checks against the platform capabilities and available resources. It will only proceed, if the specified resources are available within the platform during runtime on all given clusters.

4.1.1.3 Infrastructure Resource Descriptor

The Infrastructure Resource Descriptor maintains platform information and offers this information via the O2 interface. Furthermore, this information is used from the Validator and Parsers to evaluate the resource requirements of the deployable SFs on each of the clusters of the FLAME platform.

4.1.2 Service Function Endpoint Management & Control

The sub-components of the Service Function Endpoint Management & Control are depicted in Figure 14. It consists of three elements, the Service Function Control, the Virtual Instance Manager and the Service Function Monitor.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

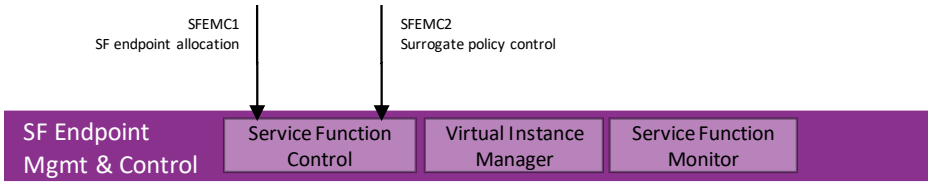

Figure 12: Sub-Components of the Service Function Endpoint Management and Control

4.1.2.1 Service Function Control

The Service Function Control is utilized to allocate resources of SFEs per cluster, based on the given VM/Container image, the resource specifications per SF node from the TOSCA template, as well as on the number of possible replications on the clusters of the FLAME platform.

4.1.2.2 Virtual Instance Manager

The Virtual Instance Manager maintains the lifecycle of each SFE. Based on given instructions via policies, the Instance Manager sets the SFEs into a targeted lifecycle state. The following state could be set as target per SFE: Non-Placed, Placed, Booted, and Connected. Each SFE passes certain states within this state machine. This is monitored by the Service Function Monitor.

4.1.2.3 Service Function Monitor

The Service Function Monitor keeps track of the resource consumption within the FLAME platform for each cluster. Furthermore, it observes the current state of each SFE.

4.1.3 Service Function Routing

The sub-components of the Service Function Routing component are shown in Figure 13, with a decomposition within a network deployment shown in Figure 14. The components are aligned with the emerging *Service Framework* of the enhanced Service-Based Architecture (eSBA) in 3GPP Release 16 [3GPP_SBA] with the Service Router sub-component mapping onto the *Message Routing* of the service framework and the *Path Computation Entity* sub-component mapping onto the Network Resolution Function (NRF) of the service framework. With this, we ensure an alignment of the SFR component with future cloud-native operator environments realized via the eSBA specifications.

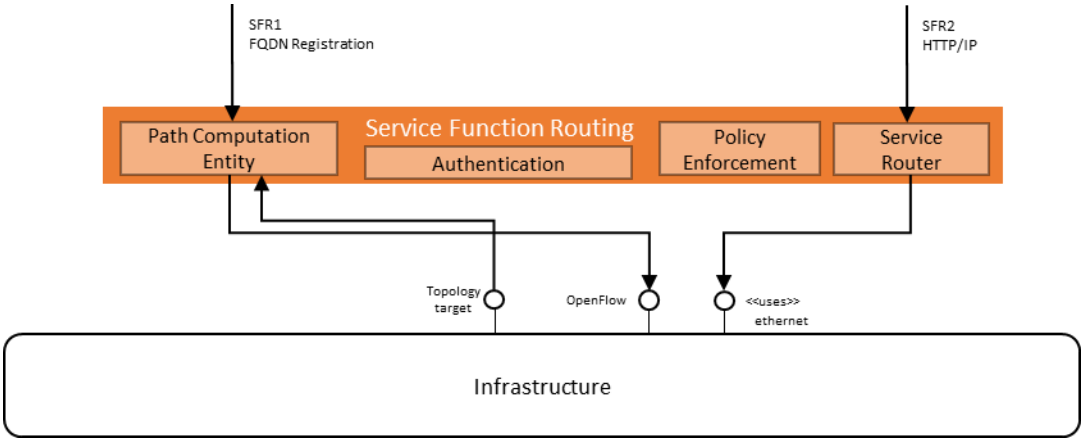

Figure 13: Sub-Components of the Service Function Routing

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

As can be seen in Figure 14, the SR functions are distributed at the ingress/egress to the network with the PCE being realized centrally together with the Authentication and Policy Enforcement sub-components.

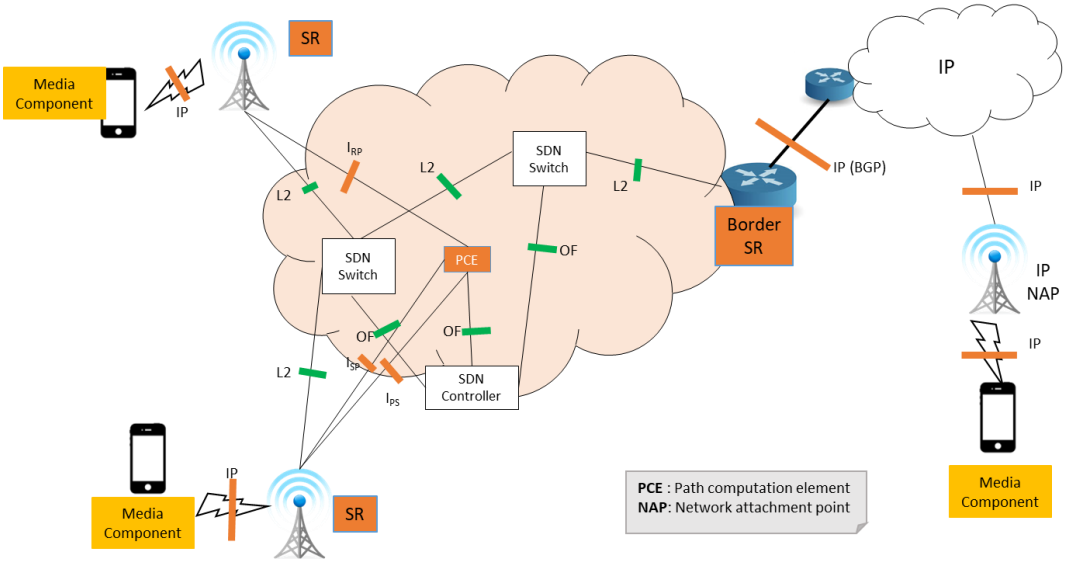

Figure 14: Service Function Routing SF Decomposition as Network Architecture

4.1.3.1 Service Router

Each media component (or any other IP-based device) is connected to at least one SR at the ingress point to the transport network. This SR serves as an edge protocol termination point, i.e. terminating application sessions, transport protocol sessions or below. In the current realization of the SR, HTTP sessions are terminated in a proxy operation, i.e. the TCP sessions is being terminated and the HTTP request is being forwarded towards a suitable other SR in the network. Any other protocol but HTTP is terminated at the IP level, forwarding the resulting IP packets to a suitable other SR in the network. At said suitable other SR in the network, the HTTP or IP level packets are then forwarded to a suitable media component (or generally IP-based service endpoint). The Border SR shown in Figure 14 acts as an SR but connects to a peering autonomous system rather than a media component, for cases that services are being requested from outside the infrastructure domain provided by the FLAME platform. In other words, the Border SR is connected to a Wide Area Network (WAN) service at the IP level, which in turn forwards packets to the Internet. The SR performs a number of functions relating to operations and management, e.g., for registration of service endpoints at the fully qualified domain name (FQDN) level, handling of path updates, mobility support and more. With this, the SR implements the SFR2 interface, as shown in Figure 13.

4.1.3.2 Path Computation Element

In order to configure the forwarding of HTTP or IP-level packets from one SR to another (see above), the PCE SF is being utilized. This PCE works closely with the underlying infrastructure, through suitable management interfaces, specifically the I1 and I3 infrastructure interfaces, to configure the forwarding actions of the SDN-enabled switching fabric and to obtain the topology information from the infrastructure to do so. For this, the PCE assigns every link of the underlying network topology to a unique bit in a pre-defined bitarray (with the size of the bitarray large enough to store all such unique bit positions, i.e. the size of the array equals the number of network links). A forwarding action from point A in the network to another point B can now be represented a unique bitarray in which all bitpositions of those links belonging to the path from A to B.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

Figure 15 shows this simple forwarding operation. In this example, the path from S (originating in a peering network) to A is represented as a simple binary OR of the three links from the border GW to the SR to which media component A connects, i.e. 10010010 is the binary representation of this path. On the other hand, the path from S to B is represented as 10100010 as one link is different here compared to the previous. With that in mind, the transfer of a packet from S to A and B is now easily achieved by a binary OR of the individual path. We refer to this capability as **coincidental multicast**, which is utilized to send a single HTTP-level response to more than one client at the same time (for cases in which the arrival of the client requests is aligned in the right arrival window).

It is the role of the PCE to receive path computation requests in the form of either an IP or FQDN destination as well as a source node identifier (which is assigned to each SR by the PCE during the bootstrapping process). The result of this operation is the bitarray, as shown in Figure 15, and is returned to the requesting, i.e. originating SR. In order to improve HTTP (request/response) operations, the PCE will also return the return path from the destination SR to the requesting SR in the same operation. This supports requests and responses being sent over different paths in the network.

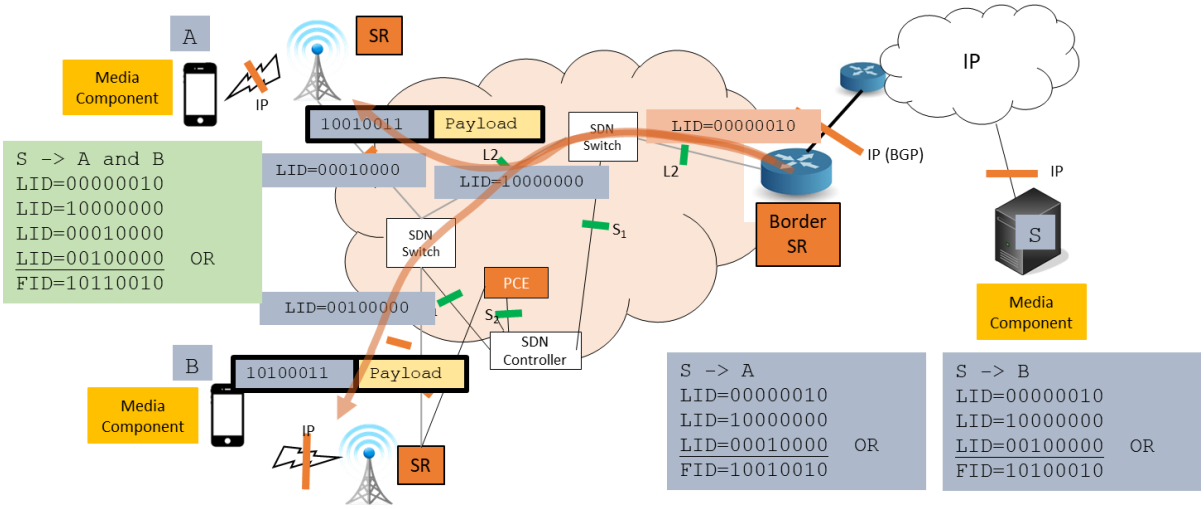

Figure 15: Service Function Routing SF with SDN-based Bitfield Forwarding

In terms of component interfaces, the PCE implements the FQDN registration interface SFR1, suitably configuring the appropriate SRs for receiving HTTP requests sent to those registered FQDNs.

4.1.3.3 Authentication

The authentication sub-component of the SFR handles the certificates being used for HTTPS traffic. Generally, the SFR routes requests without specific knowledge of certificates, only utilizing the FQDN information in the TLS handshake between client and SR. However, in order to utilize the coincidental multicast capabilities outlined in the previous sub-section, the SR will need to partially decrypt and re-encrypt the incoming HTTP request in order to determine a suitable request identifier for possible

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

multicast responses. The certificate is provided via the SFR1 interface to the PCE[1], which in turn stores the certificate information in the authentication sub-component.

4.1.3.4 Policy Enforcement

The path based forwarding and the utilization of the PCE sub-component allows for attaching policies to the computation of suitable paths as well as the selection of suitable SFEs from a set of possible available ones. The Policy Enforcement sub-component implements the handling of provided policies by verifying those policies and providing them as suitable constraints to the PCE sub-component. In the beta release of the FLAME platform, a default policy of 'shortest path' is implemented for the path calculation, selecting the nearest SFE from a set of possible available ones. For the release candidate of the FLAME platform, we foresee exposing a suitable component-level interface to the SFEMC, which will derive policies from provided TOSCA templates.

4.1.3.5 Forwarding at Infrastructure Level

The SFR relies on infrastructure level forwarding operations that utilize the bitfield information of the forwarding path, outlined in Section 4.1.3.2. There are two known realizations of this forwarding operations, namely:

- **SDN-based**: This solution encapsulates the packets into a standard Ethernet frame format. It utilizes the IPv6 source and destination fields of the Ethernet header for storing the aforementioned bitarray. During the bootstrapping of the network during the orchestration of the infrastructure-facing resources (not the media component facing ones), the PCE will interact with the SDN-based infrastructure to configure suitable forwarding rules in each intermediary SDN switch of the infrastructure. For each SDN switch, a number of rules needs to be defined that is propositional to the number of supported hardware output ports (often 48) at said local switch. The forwarding is realized by checking the unique bitposition for each individual output port and, if set to 1, forwarding the incoming packet over said output port. With that, multicast transmission is easily supported, as shown in Figure 15. The required SDN match operation is that of a wildcard (with the wildcard being the bitarray with only the specific output port bitposition set) and is supported with OpenFlow V1.2 [OpenFlow] upwards, i.e., in most commercially available SDN switches. For more information on this forwarding approach, see [Reed16], including memory costs.

- **BIER-based**: The Bit Indexed Explicit Replication (BIER) working group of the IETF has been defining a bit-based forwarding solution similar to that outlined above for SDN. Instead of labelling links, the specific egress, i.e. outgoing, router is specified with a unique bitposition in a given bitarray. The principles of path construction as well as ad-hoc multicast delivery (by combining two or more paths into a single multicast one by a simply binary OR) remain the same. Mappings of the BIER forwarding architecture are provided over SDN, MPLS and other transport networks. [Pur18] outlines the use case for an HTTP-specific SR utilizing such forwarding solution.

[1] At the stage of the beta release, the certificate is still provided via a separate management webGUI but we expect to move this upload capability to the SFR1 interface for the release candidate of the FLAME platform.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

4.1.3.6 Protocols for Realizing the Sub-SF Interactions

Given the distributed nature of the SR and PCE interaction, there are a number of protocols involved in the coordination of those sub-components, dealing with path requests, forwarding updates, certificate sharing and others. Those protocols are currently specified as part of the work in the IETF Service Functions Chaining (SFC) WG with a current version of the standard contribution available at [Trossen18]. We expect future FLAME platform releases to align to those specifications.

4.1.4 CLMC

The sub-components of the Cross Layer Management and Control component are shown in Figure 16.

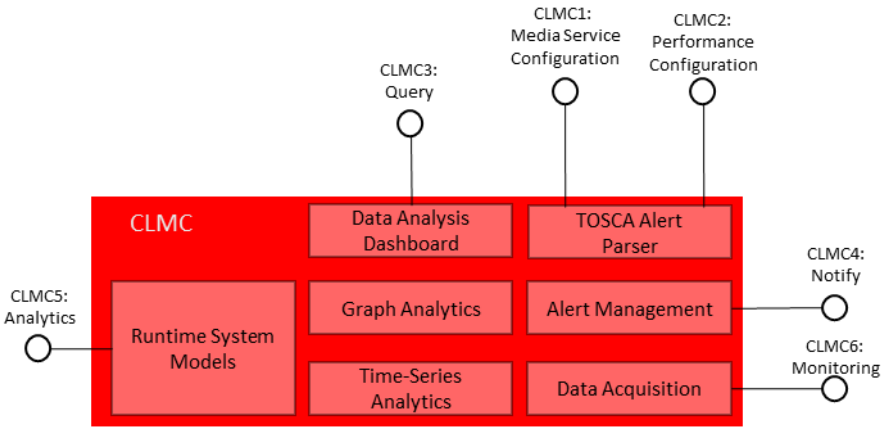

Figure 16: Sub-Components of the Cross Layer Management and Control

Data acquisition is achieved in accordance with the CLMC information model designed to support service management and control decisions. Specifically, the information model supports the exploration and understanding of state and factors contributing to changes in state over time as shown in the primitive below:

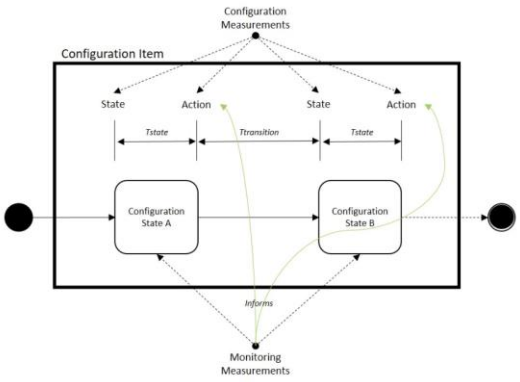

Figure 17: CLMC Configuration State

The overall system (infrastructure, platform and media services) is composed of a set of configuration items that transition between different states during the lifecycle of the system. Configuration items of interest include significant components whose state change influence the response of the system. In general, the information supports the process of:

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

- Identification of significant configuration items within the system
- Assertion of state using configuration measurements
- Measurement of response from monitoring measurements
- Support for taking action in relation to configuration state

Configuration information describes the structure and state of the system over time. Each configuration item has a lifecycle that defines configuration states and events that cause a transition between states. Monitoring measures the behaviour of the system and system components overtime including metrics associated with usage and performance. Measurements are made within the context of a known configuration state. Usage monitoring information can include measurements such as network resource usage, host resource usage and service usage. Performance monitoring information can include measurements such as latency (seconds/request), throughput (requests/second), average response time and error rates.

A media service is as "An Internet accessible service supporting processing, storage and retrieval of content resources hosted and managed by the FLAME platform". A media service consists of one or more Service Functions that together are composed to create an overall Service Function Chain. SFs are realised through the instantiation of virtual machines (or containers) deployed on servers based on resource management policy. Multiple VMs may be instantiated for each SF to create surrogate SFs, for example, to balance load and deliver against performance targets such as latency. Media Services, SFCs, SFs, VMs, links and servers are all examples of configuration items.

Media services are described using a template structured according to the TOSCA specification (http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html). A TOSCA template includes all of the information needed for the FLAME orchestrator to instantiate a media service. This includes all SF's, links between SFs, resource specification and policy information. The TOSCA template provides the initial structure of the Media Service information model through specified service and resource configuration. Within this structure, system components are instantiated whose runtime characteristics are measured to inform management and control processes. Measurements relate to individual SF's as well as aggregated measurements structured according the CLMC information model. Measurements are made by monitoring processes deployed with system components. The configured items provide the context for measurements.

The information model in relation to the high-level media service lifecycle is shown in Figure 18. The lifecycle includes processes for design (packaging) and runtime orchestration, routing and SF endpoint management/control. Each stage in the process creates context for decisions and measurements within the next stage of the lifecycle. Packaging creates the context for orchestration whilst orchestration creates the context for endpoint instantiation, and network topology management.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

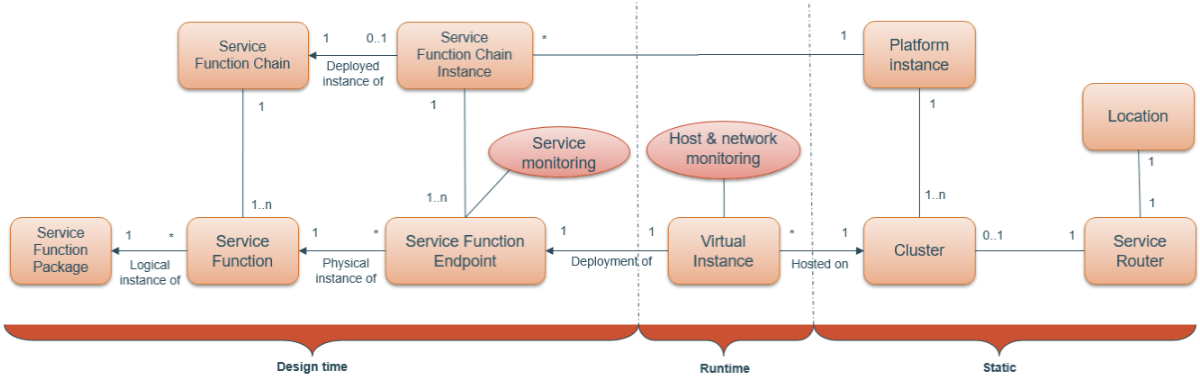

Figure 18: CLMC Information Model

The primary measurement point for a media service is an SF endpoint. An SF endpoint is an instantiation of a Service Function within a Virtual Instance on a Cluster. An SF endpoint exists within two main contexts: media service and virtual infrastructure. The media service context relates to the use of the endpoint within a service function chain designed to process and deliver content. The virtual infrastructure context relates to the host and network environment into which the endpoint is deployed. Deploying monitoring agents in different contexts and sharing information between contexts is a key part of cross-layer management and control.

The diagram highlights the need to monitor three views on an endpoint: network, host, and service. The measurements are captured by different processes running on servers but are brought together by common context allowing the information to be integrated, correlated and analysed. The endpoint can measure a service view related to the content being delivered such as request rates, content types, etc, a VM can measure a virtual infrastructure view of a single endpoint, and the network view can measure an infrastructure view across multiple endpoints deployed on a server. These monitoring processes running on the server are managed by different stakeholders, for example, the platform operator would monitor servers, where by the media service provider would monitor service specific usage.

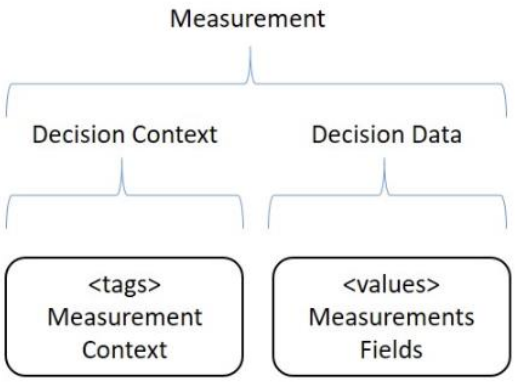

Figure 19: Decision context and decision data

Every measurement has a measurement context. The context allows time-based series to be created according to a set of query criteria, which are then be processed to calculate statistical data over the desired time-period for the series. The FLAME information model supports KPIs calculated from measurement fields and dimensions encoded within measurement tags. This lightweight implementation allow for a broad range of questions to be asked about the cross layer information

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

acquired. The designing of context for measurements allows for integration and processing of between data from multiple monitoring sources. The design principles adopted include:

- identify common context across different measurements
- where possible use the same identifiers and naming conventions for context across different measurements
- organise the context into hierarchies that are automatically added to measurements during the collection process

All of the measurements on a SF endpoint share a common context that includes tag values:

- **flame_sfc** – an orchestration template (a.k.a. Service Function Chain)
- **flame_sfci** – an instance of the orchestration template (a.k.a. Service Function Chain Instance)
- **flame_sfp** – the package a service function is using (a.k.a. Service Function Package)
- **flame_sf** – the service function defined in the TOSCA resource specification (a.k.a. Service Function)
- **flame_sfe** – an authoritative copy of the SF - either VM or container (a.k.a. Service Function Endpoint)
- **flame_server** – a cluster VM inside which service function endpoints are placed (at the current stage, the value of this tag is the same as the location tag)
- **flame_location** – the location of the server - physical machine that hosts the cluster VM

Not all information acquired will be aggregated and stored within the CLMC. The CLMC is not responsible for capturing every measurement point related to transferring bytes over the network. It is also not responsible for capturing every interaction between a user and a service. The key design principle is to acquire information from one context that can be used in another context. For example, instead of recording every service interaction, an aggregate service usage metric (e.g. request rate/s) would be acquired and stored, and the similar aggregation would be needed for infrastructure monitoring.

4.1.5 Data Acquisition

Data Acquisition collects time-series measurements from different layers including infrastructure, platform and media services. Data is collected to support service design, management and control decisions resulting in state changes in configuration items. The link between decisions and data is through queries and rules applied to contextual information stored with measurement values.

The monitoring model uses an agent-based approach with hierarchical aggregation used as required for different time scales of decision-making, as shown in Figure 20.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

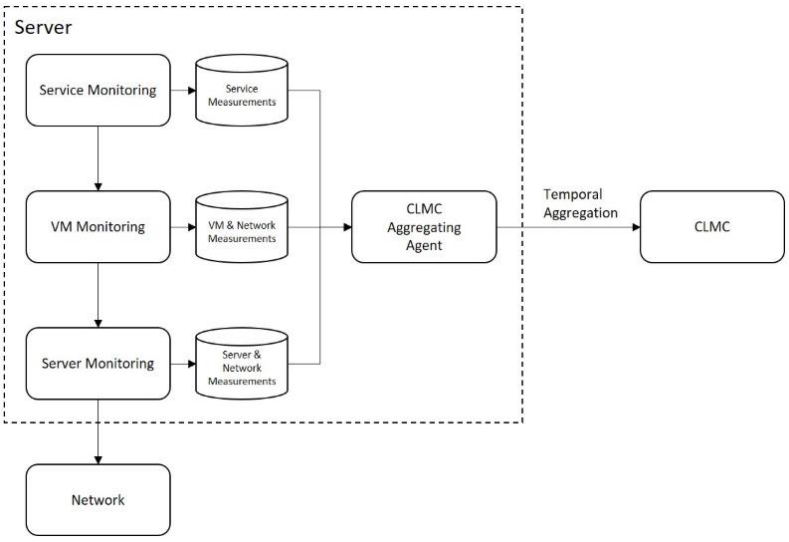

Figure 20: Data Acquisition Architecture

To monitor a SF an agent is deployed on each SF endpoint. The agent is configured with measurement context incrementally at difference stages of its lifecycle. During packaging, the media service developer installs the monitoring agent and configures the SF package name. A **whoami** service is also installed on each SF endpoint to discover runtime context from the platform including the Service Function Chain, Service Function Chain Instance, Cluster, Service Function IDs, and endpoint ids.

4.1.6 TOSCA Alert Parser

TOSCA Alert Parser parses and validates an Alerts Specification Document and configures time-series queries, conditions and notification handlers configured to receive activated trigger events. The alert specification is written in YAML and is a TOSCA-compliant document according to the TOSCA simple profile.

The Alerts Specification Document consists of two main sections - metadata and policies. Each policy contains a set of triggers. A trigger is a fully qualified specification for an alert. The metadata section specifies the service function chain ID to which the alert specification relates. The policies section defines a list of policy nodes, each representing a fully qualified configuration for an alert within the CLMC. Each policy must be of type eu.ict-flame.policies.Alert.

The Alert Specification includes a ***policy_identifier*** that must match with a StateChange policy in the TOSCA resource specification used to define a media service that is submitted to the Orchestrator. The ***event_identifier*** is the name of the event and must match with the constraint event in the TOSCA resource specification. An **event_type** refers to the process used to create the alert. This includes the how data is processed and the conditions under which the alert is triggered, for example, threshold, relative or deadman conditions. A set of configuration defines the conditions for triggering an alert. This includes

- **Metric:** the measurement name and field
- **Threshold:** the critical value compared to the measured value using threshold, relative or deadman
- **Granularity:** period to consider in seconds

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

- **Aggregation_method:** statistical function to apply to data points over period
- **Resource_type**: global tags to uniquely identify the context for the event
- **Comparison_operator**: logical operator for use in comparison between measured value and threshold
- **Implementation**: a list of the URL entries for alert handlers to which alert data is sent when the event condition is true. Alerts can be sent to media services or platform services. If the alert is to be sent to SFEMC, then instead of typing a URL a well-known label flame_sfemc is used.

4.1.7 Alert Management

Alert Management triggers notification events based on conditional policies associated with time-series measurements. Every alert handler registered in a trigger action --> implementation section of the TOSCA Alerts Specification Document receives an alert message when the trigger event condition is true. This alert message is sent using an HTTP POST request to the URL of the alert handler. The alert message includes

- **id**: trigger ID as defined in the alert specification document
- **level**: the level of the alert; currently all alerts that trigger have their level set as CRITICAL
- **previousLevel**: the previous level of the alert
- **details**: the context for the alert in the format "db=<db_name>,sfc=<sfc_id>,sfci=<sfc_instance_id>,policy=<policy_id>"
- **duration:** integer, duration of the alert in nanoseconds
- **time**: timestamp of the point that triggered the alert
- **data**: describes the point(s) that triggered the alert
- **data.series.name**: the name of the measurement
- **data.series.tags**:(OPTIONAL) key-value pairs for all measurement tags used to filter the data series
- **data.series.columns**: list of column names used to trigger the alert
- **data.series.values:** list of list of values, each nested list represents a measurement point and the values for each column

4.1.8 Graph Analytics

Graph Analytics connect related time-series data sets and allows complex pattern matching using graph structure and node properties. Graph analytics is used to understand system level properties, for example, end-to-end response time of a service function chain.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

The topology of a system graph is constructed from smaller topologies representing different aspects of the overall system. Each topology has a different lifecycle and dynamic characteristics that need to be considered. Each topology consists of Nodes representing configuration items with properties defining state, along with Edges between nodes defining relationships which themselves can have properties (e.g. network latency between Clusters).

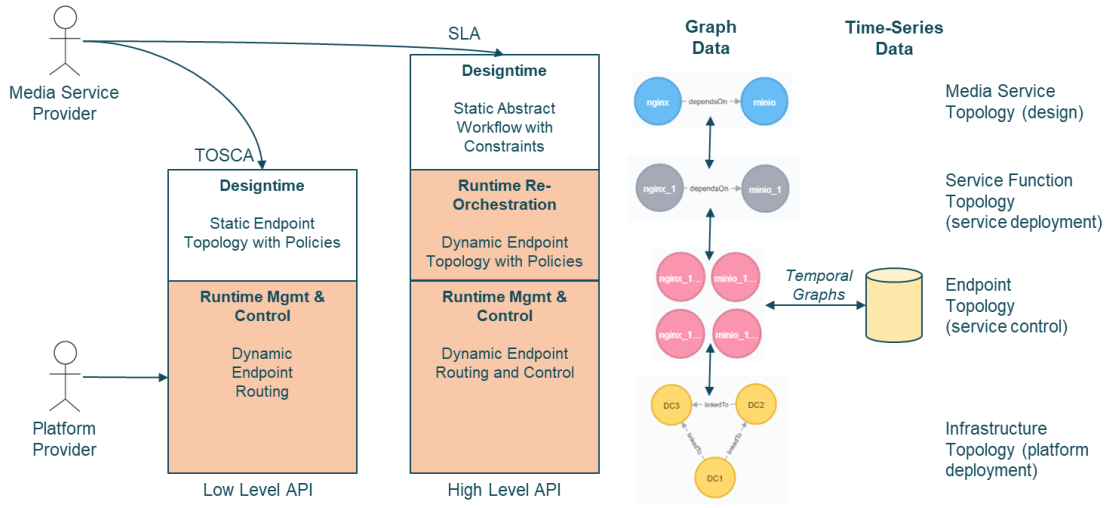

Figure 21: Relationship between topologies

The Infrastructure Topology is an infrastructure slice defined in terms of Clusters and Network Links between them. The topology is derived from the infrastructure slice provided to the Platform Provider on deployment. The topology is considered relatively fixed as it represents the wholesale capacity allocated to the Platform Provider. The Media Service Topology is the design-time Service Function Chain defined in terms of SFs and connections between them. The topology is derived from the TOSCA Resource Specification and it provides an abstract definition of the media service workflow. The Service Function Topology is the set of SFs created when a media service is deployed. SFs are still abstract as they represent a one or more Service Function Endpoints deployed within the platform. Finally the SF Endpoint topology is derived from the deployed SF endpoints and represents all surrogates deployed as part of the media service on the infrastructure slice.

Graphs are created dynamically to support specific analytics and queries over the system properties. The overall structure of the graph is consistent across all Nodes and Edges as this represents the information model of the CLMC. The infrastructure properties are also consistent across all graph queries including the relationship between SF Endpoints and Clusters as the infrastructure model is homogeneous. The properties of specific SF Endpoints varies depending on the type of service, although the common KPI taxonomy allows for general abstractions, aggregation and normalisations can to be defined (e.g. Response Time) even though the measurement fields from data acquisition use different naming conventions or sampling periods.

Figure 22 shows the process of graph building and analytics. The process is initialised to build the initial graph from the infrastructure and media service topologies, along with configuring the continuous queries to acquire, aggregate and normalise the desired SF Endpoint properties over a specified time-period. The continuous queries execute periodically to add new nodes to the graph representing state of a SF Endpoint over the period. The graphs are created automatically from the measurement context data reported by the monitoring agents. A subsequent graph query is executed continuously to determine system measurements and stores these as monitoring data in time-series database for visualisation or further higher-level analytics.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

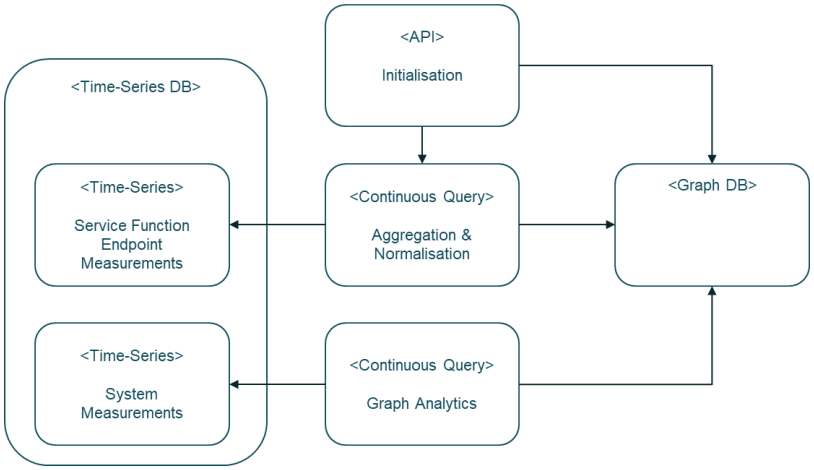

Figure 22: Graph building and analytics processes

4.1.9 Data Analysis Dashboard

Data Analysis Dashboard provides a human interface to time-series data sets allowing the exploration of available measurements including resource usage and performance. The dashboard allows monitoring views to be configured that are specific to stakeholders or media services.

4.1.10 Runtime System Models

Runtime Systems Models simulate media service risk, cost and performance using expert knowledge and empirical data derived from runtime systems fast enough to re-orchestrate media service resource configuration state at runtime. RSMs can be initiated by media service providers exploring the initial conditions for TOSCA resource specifications or can be triggered in response to Alerts. Many different techniques exist to explore system behaviour (e.g., constraint based optimisation, Monte Carlo simulation, semantic reasoning, etc.). FLAME is not prescriptive on the approach, however, the inclusion of RSMs in the architecture is necessary to link operational management and decisions to service performance, especially in highly dynamic systems where changes to infrastructure slices and service function chains are to be expected.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

5 SERVICE FUNCTION CHAINS

The following subsections present various SFCs for the main internal interactions between core components of the platform architecture, as shown in Figure 1. We divide the interactions into various areas. We will start with the *experimentation SFC*, which connects with the *management & control SFC* for driving the management policies of the experiment into the FLAME platform. We then outline the *main (FMI) data plane SFC* for the exchange of media service interactions over the data plane provided by the FLAME platform. Due to the core contributions of the service function routing solution to the benefits of the FLAME platform, as outlined in D2.1, we further decompose the service routing component of our platform into the specific *service function routing SFC*. As a crucial enabler for the controllability and observability of our platform, we further decompose the CLMC part of our platform as a dedicated SFC.

Throughout the presentation of the various SFCs, we will reference the specific component interfaces being utilized. There will, however, be several SFC-internal interfaces and protocols, whose exact details are not within the scope of D3.10 but will be covered throughout the various realizations of the SFCs in WP3. We will also utilize the colour coding of Figure 1 to indicate which component realizes which service function.

5.1.1 Orchestration SFC

Figure 23 shows the orchestration process of SFs into a specific Media Service cluster. The MSP describes its service in a TOSCA resource specification file (orchestration template) and provides this to the orchestrator's validator unit (1a). The CLMC may optimise the resource specification in a dialog with the MS provider (1b). Both steps utilize the *O1* interface of the Orchestrator component. The resource specification also includes package pointer of the MS repository the SFEMC should pull from. Such images may contain full VM package or only containerized MS packages.

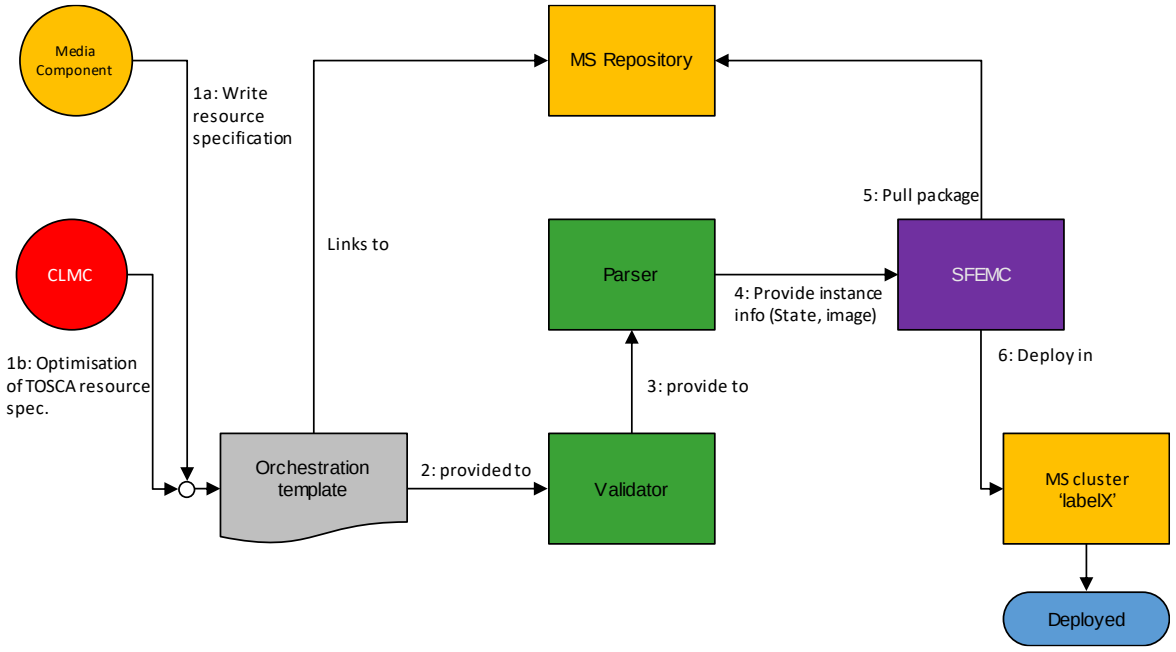

Figure 23: Orchestration of SFs

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

After the template is provided, the validator validates at syntactical and semantical level against the TOSCA NVF and FLAME specifications (2). When successful, it will be forwarded to the parser element to interpret the content and create data objects for the orchestrator (3).

The parser then forwards the data objects towards the SFEMC (4) via the *SFEMC1* interface. These data objects include the pointer for the MS package, the type of virtualization and the target state (see Section 5.1.2) of the deployed instances.

The SFEMC performs a package pull and deploys an instance on the targeted MS cluster with the given SF state (5 and 6). The targeted state must not be **Non-Placed** to perform this action. After the pull and initial deployment are performed the SFEMC sets the SF instances into the targeted state per cluster.

The SFs are then deployed on the targeted MS cluster with the targeted state.

5.1.2 SFE Management & Control SFC

SFs are initially deployed by the FLAME platform orchestrator as SFEs and the SFE's state is controlled by the SFEMC. Via State Policies on orchestration level, the SFEs may be located on defined clusters with a finalised SFE's state. Every possible SFE's state is reflected in the SFE's State Machine as depicted in Figure 24. A SFE state target can be **Non-Placed**, **Placed**, **Booted** or **Connected**. Furthermore, the state machine contains **transitional states**. However, such states are not selectable as a target state. They offer a finer grained view of any change from one state to another. Before entering and leaving a transitional state and event is raised via the Service Function Monitor (see Section 4.1.2) on FLAME platform level. The events are labelled as function calls between the states.

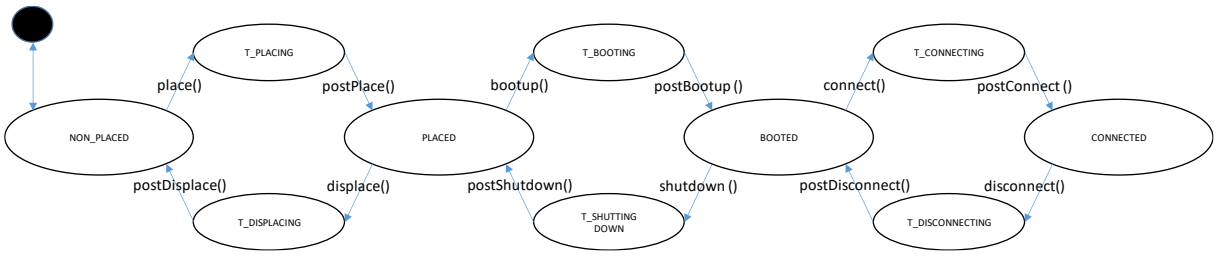

Figure 24: Service Function State Machine with Transition States

The lifecycle state of such SF may change during the lifetime of the deployed service. This may be caused by external influences such as monitoring data, time schedules, network topology changes or failures on hard- or software level. Such an external influence may come from the FLAME's platform CLMC. Figure 25 shows the interaction between the Orchestrator, CLMC, SFR, and the SFEMC for deploying and controlling of SFEs. After parsing the orchestration template, the Orchestrator provides the suitable SF information to the Service Function Control sub-component of the SFEMC via the *SFEMC1* interface. This in turn instructs the Virtual Instance Manager to initiate the appropriate SFE in the state specified in the template, while instructing the PCE of the SFR component via the *SFR1* interface to register the FQDN of the SFE.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

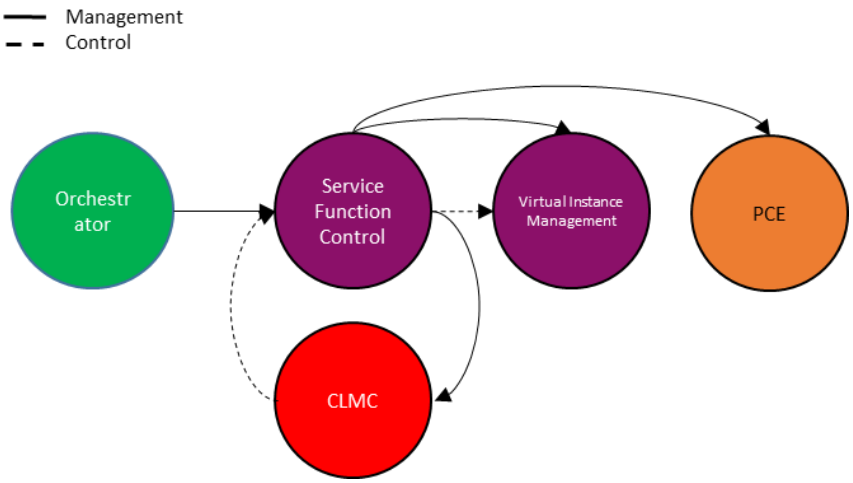

Figure 25: Interaction between CLMC and SFEMC

Furthermore, the Service Function Control sub-component of the SFEMC instructs the CLMC, via the Config interface, to configure the suitable trigger as specified in the orchestration template for the SFE.

At the control level, the suitable trigger information is provided from the CLMC to the Service Function Control sub-component via the *SFEMC2* interface, providing the necessary event information which may trigger a lifecycle state change of SF instances within a specific cluster. However, the event source is not limited to the CLMC. Other data analysis and trigger procedures may call this interface. The general handling based on events is specified in the TOSCA resource specification as a policy and related trigger elements respectively.

5.1.3 Main (FMI) Data Plane SFC

The main interactions for a media service at the data plane is shown in Figure 26 as being realized through the service function routing component of the FLAME platform via the *SFR2* data plane interface that connects said media components at the data plane level. As indicated in Figure 1, the media components itself are not part of the FLAME itself albeit their chaining at the data plane is being realized by the FLAME platform. By preserving the data plane abstractions of well-known protocols, such as HTTP, RTP or generally any IP-based protocol, the SFC-based decomposition of the FLAME platform is not exposed to the media components and therefore the media service developers, allowing for utilizing any available composition technique for this part of the overall system.

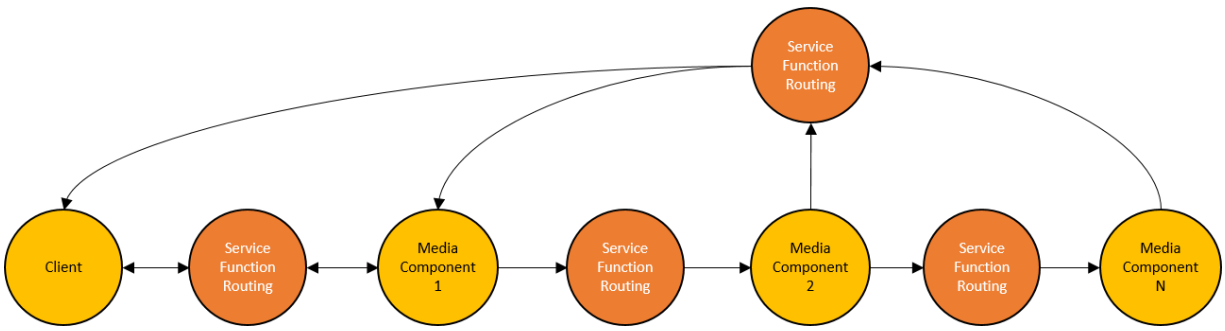

Figure 26: Main (FMI) Data Plane SFC

The routing function is realized by the Service Function Routing component of the FLAME platform. From an SFC architecture [SFC] perspective, this SF enables flexible and dynamic chaining of the

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

connection service functions, here the media components, including the flexible redirection to virtualized instances. This capability is outlined in [Trossen18] and its realization is presented in the following subsection.

5.1.4 Service Function Routing SFC

Figure 27 further decomposes the Service Function routing SF, introduced in Figure 26, realizing the data plane transfer of media service interactions over the underlying infrastructure. In IETF SFC WG parlance [SFC], the transfer of the infrastructure is realized as a *transport-derived Service Function Forwarder* (tSFF). The composition is shown as a sub-component architecture in Figure 14. In the following sub-sections, we further describe the operations of these sub-SFs.

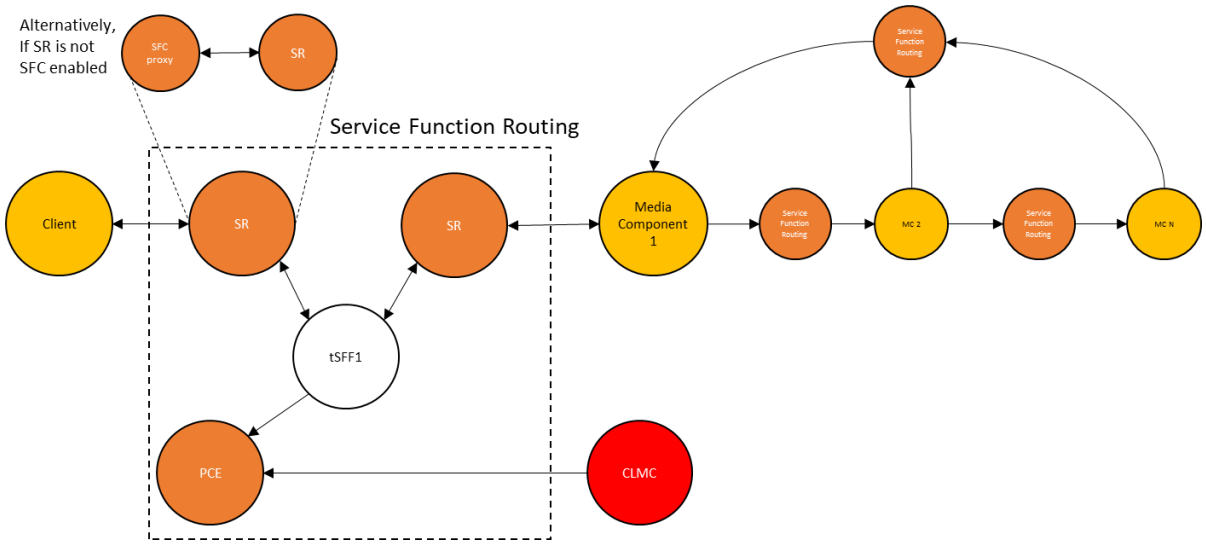

Figure 27: Service Function Routing SF Decomposition

The SFC decomposition shown in Figure 27 shows a general realization over an SFC-compliant service chaining architecture. With this, media components could be connected via several Layer 2 links to the Service function routing SF, e.g., within a cable network.

In most of our use cases, however, we can assume that media components will be connected to the SRs of the SFR via a link-local, e.g., single Ethernet/WiFi, link. Even in the case of the media component being hosted in a data centre, the latter would appear as the IP-level endpoint to the FLAME platform (with the data centre networking to the actual media component instances being opaque to the FLAME platform). For this reason, we can simplify the SFC to the one shown in Figure 28, where the forwarding between media components and the SRs are realized by another $tSFF_2$, representing Ethernet-based link-local IP-level communication.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

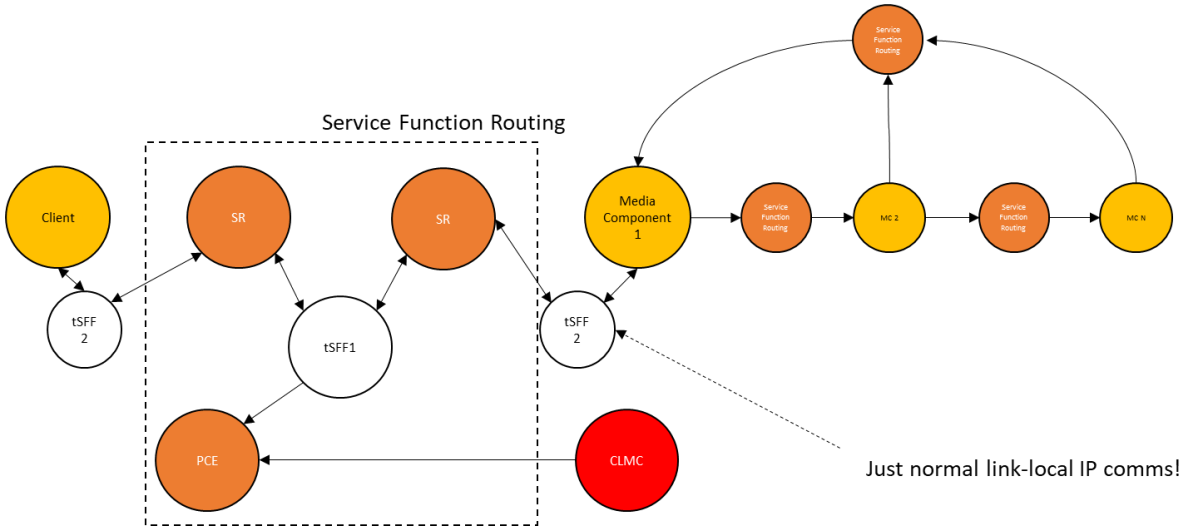

Figure 28: Service Function Routing SFC with link-local clients and media components

5.1.5 CLMC SFC

The SFC decomposition of the CLMC is shown in Figure 29. The SFC describes a stream-processing pipeline that delivers stream-based graphs for cross-layer knowledge that can be queried and analysed at runtime by media and platform components, or explored by aiming to design templates.

The SFC is designed to support multiple sources of monitoring information from information providers (***CLMC6***) at the platform and media service levels distributed to the distributed transactional time-series data storage. Examples of this data include service request events, service response events, resource usage from service functions and hosts. This data is high frequency and high volume data that is delivered using a Pub/Sub SF. The transactional data sources are stored in a various Transactional Data Storage SF sources relevant to the components being monitored. For example, the data stores used to store network monitoring may be different from those used to store user interaction and media service usage due to factors such as technology, ownership and control.

The transaction data storage provides an evidence base for system behaviour. The evidence base is persistent with appropriate retention and access policies defined according to the needs of queries underpinning service management and control decisions. The evidence provides quantifiable data supporting hypothesis testing, allows evidence for effectives of the platform to be established, and subject to permission, allows multiple stakeholders to explore openly the consequence of collaborative decision making in interactive media systems. Where appropriate, parts of the evidence base will be made available as open research data.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

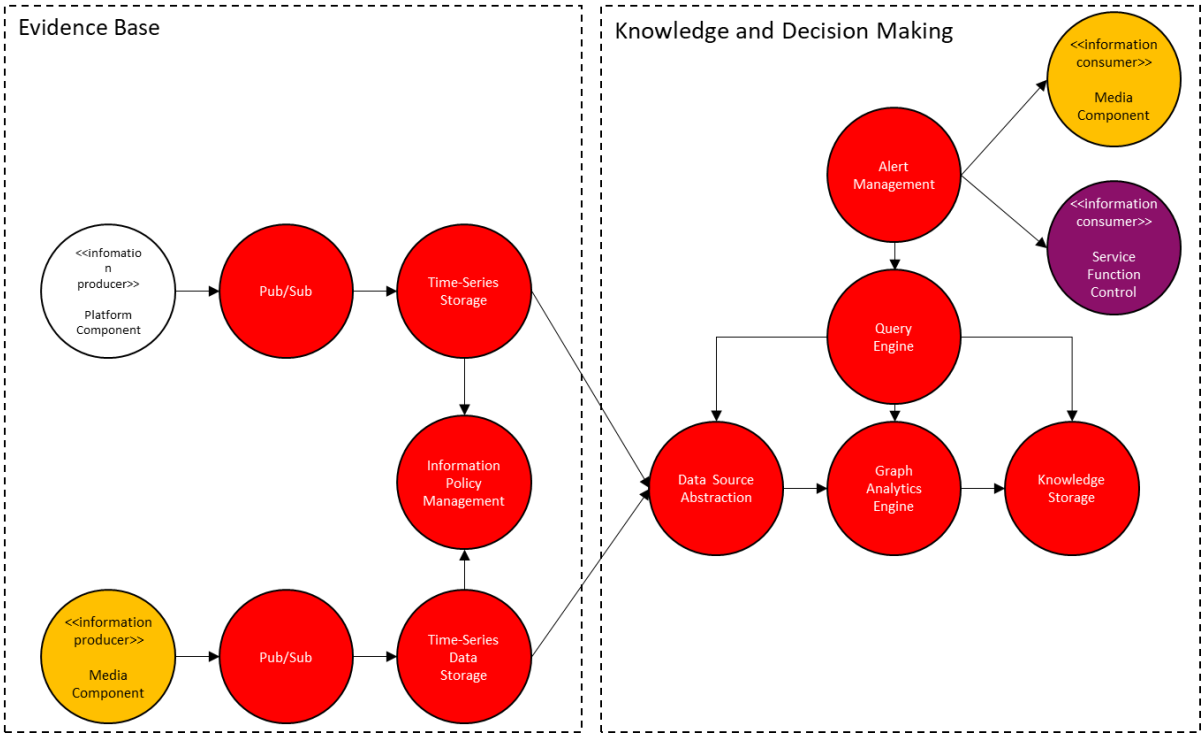

Figure 29: CLMC SFC

The data sources are then processed by a Data Source Abstraction SF to create data abstractions representing the dimensions relevant to specific cross-layer management queries and analytics. The creation of abstractions from different data sources can be computationally intensive activity considering the scale and velocity of monitoring data. Various abstraction algorithms can be considered from simple numerical aggregation to more complex machine learning and graph analytics.

The selection of the dimensions is driven by queries posed by Alert Management SF (***CLMC2***). The queries are executed by a Query Engine SF (***CLMC3***) that selects the time-series dimensions (data abstractions) needed and accesses graphs stored within the Knowledge Storage SF. The query language for time-series analytics is based on InfluxQL[2] and RESTful APIs. The generation of the graphs is achieved using the Analytics Engine SF as part of the CLMC service, which requests data abstractions from time series data and maintains currency of graphs according to query specifications. The graph database is uses Neo4J[3]. The analytics SFs allow the evidence base to be explored in ways that translate the evidence into knowledge and decision-making. This knowledge can then be encoded in templates describing the configuration of services and resources necessary to achieve specific performance outcomes.

Access to the data is governed by Information Policy Management SF. Information policy defines the authorisation rules for access to data sources. It should be noted that all data stored within the CLMC is aggregated and anonymised data, and as such the CLMC must not be used to store personal data.

[2] https://docs.influxdata.com/influxdb/v1.7/query_language/spec/

[3] https://neo4j.com/

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

6 INFRASTRUCTURE INTEGRATION

FLAME separates the infrastructure from the platform following an owner <> tenant relationship model. For this purpose, certain information about the infrastructure must be passed to the platform tenant. Furthermore, it is desired that the platform tenant does not have the rights to create and modify certain infrastructure resources, which requires some sort of read-only information exchange.

At the beginning of the FLAME project, it became apparent that OpenStack is used in the infrastructure of Barcelona and Bristol and was on the verge to becomes the industry standard for a Telco-NFV environment. Thus, OpenStack has been chosen to orchestrate the FLAME platform. The realisation of the workflow described here is OpenStack centric but with the aim to keep it as NFV solution agnostic as possible.

With four sites where the FLAME platform has been deployed so far (testing environments such as IT Innovation and InterDigital Europe, as well as in the cities of Barcelona and Bristol), the repeatability of such task required an automated and well-structured workflow to deploy the FLAME platform in a new infrastructure. The outcome is an Automated platfoRm DEploymeNt Toolchain (ARDENT).

6.1.1 Purpose and Workflow

Figure 30 illustrates the workflow to deploy the FLAME platform. The four circles represent the major steps to achieve this goal and the responsibly is split between the infrastructure and platform providers, as indicated.

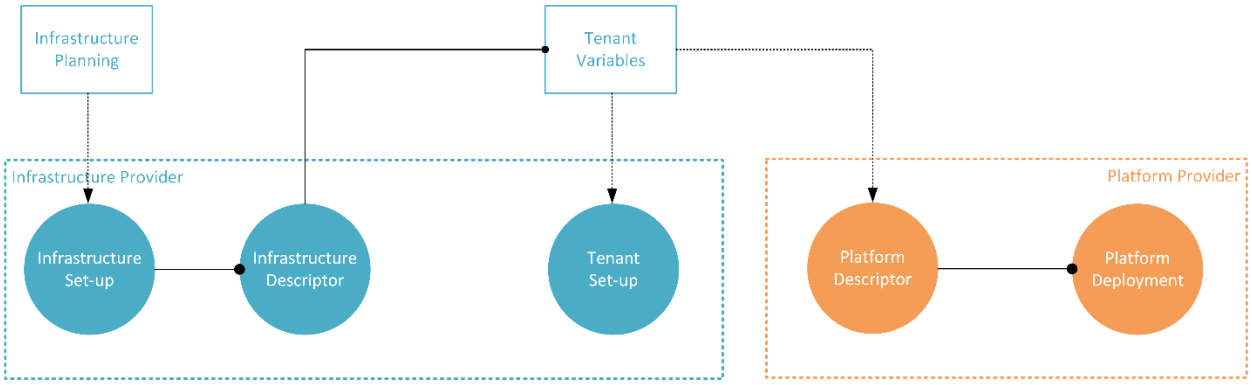

Figure 30: Workflow for Deployment of Platform

The very first step for an infrastructure provider is to **plan** the resources that should be given to the infrastructure tenant, i.e., compute, storage and networking. The creation of the various data plane as well as management plane networks is part of this procedure and results in the **infrastructure set-up**.

The **infrastructure descriptor** is a graphviz-based descriptor communicating the components of the infrastructure topology (compute nodes, infrastructure services, SDN switching fabric) as well as their physical connection and the names of the provider networks.

From the infrastructure descriptor the **tenant variables** are derived and populated in a knowledge base which is shared with the next two tasks, i.e. setting up the tenant, creating the platform descriptor and eventually deploying the platform.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

The **tenant set-up** is a collection of bash scripts with several variable files which comprise infrastructure provider-independent and infrastructure provider-specific values. The scripts allow the infrastructure provider to a) set up the FLAME tenant using an automated procedure and b) to share all necessary values which describes the NFV platform and allows the platform provider to write the platform descriptor and to deploy the platform eventually.

Given that OpenStack is the chosen NFV realisation the **platform descriptor** is a HEAT-based YAML file describing where certain platform instances are deployed and which networks they attach to. The information required to write the platform descriptor is taken from the infrastructure descriptor and the variable files written in the tenant set-up activity.

The last step is the **platform deployment** where the platform descriptor is given to OpenStack via its CLI.

6.1.2 Networks, Subnets and Security Groups

Figure 31 illustrates the provider networks including their purpose on the infrastructure level.

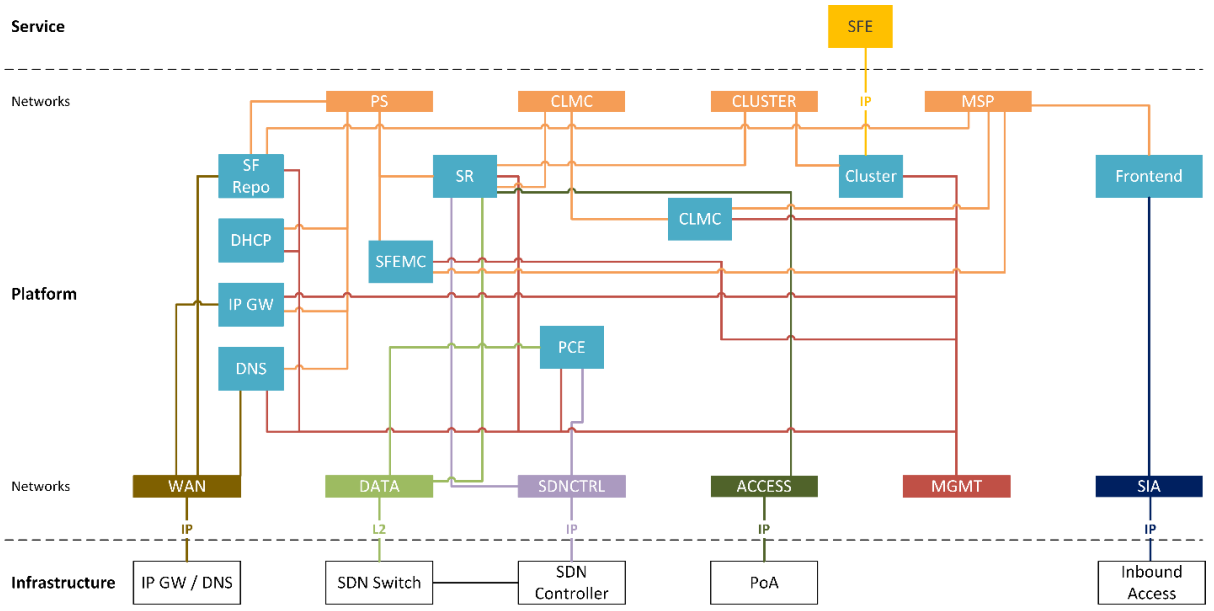

Figure 31: Infrastructure Tenant Networks and their Purpose

The platform networks allowing access to infrastructure resources are:

- WAN: The network to access the internet through one or more IP gateways of the infrastructure provider and a dedicated DNS (if provided). If the infrastructure provider does not maintain their own DNS, a public DNS is used (e.g., OpenDNS or Google). This network must be configured with DHCP, a gateway and a DNS server

- SDNCTRL: The network to reach the SDN controller from any deployed platform instance. This network must be configured with DHCP but no gateway or DNS server

- DATA: The network for interconnecting the compute nodes via the underlying SDN switching fabric of the infrastructure provider. Note, most infrastructure providers create dedicated DATA networks between an SDN switch port and a compute node often using VLANs. Therefore, the number of DATA networks very often equals the number of SDN ports

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

configured for the platform. This network is treated by the platform as an L2 link and does not require any IP configuration in the NFV platform. If the infrastructure provider does not own its own SDN switching fabric a single DATA network must be created though to allow platform instances to communicate with each other.

- ACCESS: The network that allows the platform to handle traffic from end devices connected to the infrastructure via point of attachments (WiFi, cellular, cable). As for the DATA network, the ACCESS network is most likely configured using VLANs and one network per PoA is therefore created to connect the PoA and specific compute node. The IP assignment of IP endpoints attached to this network is done via a platform service rather than an infrastructure one. The platform provider refers to this network as the LAN.

- SIA: This network provides a secured inbound access (SIA) towards the FLAME frontend platform instance that allows service providers to deploy and maintain their service function chains.

For managing the platform instances a MGMT network is required which allows the platform provider to manage and maintain its deployed platform instances.

Furthermore, the following platform networks are required:

- CLMC: The network that allows to let the CLMC and its SR to communicate via IP

- CLUSTER: The network that allows a cluster instance to communicate via IP with its SR

- PS: The network that allows the PCE/SFEMC instance and all platform service instances to communicate with their SR

- MSP: The network allows a media service provider to access the orchestrator, CLMC and the service function repository to create, maintain and configure their deployed service function chains.

Table 7 lists the various networks, subnets and security groups previously described and their properties and also when they are created during the work-flow, as described in Section 6.1.1.

Table 7: Networks, Subnets and Security Groups for FLAME Platform

Network Type	**Created When**	**Subnet**		**DHCP**	**IP GW**	**DNS**	**Security Group**	
		Configured	Created when				Configured	Created when
WAN	Infrastructure Set-up	Yes	Infrastructure Set-up	Yes	Yes	Yes	Yes	Infrastructure Set-up
SDNCTRL	Infrastructure Set-up	Yes	Infrastructure Set-up	Yes	No	No	Yes	Infrastructure Set-up
DATA	Infrastructure Set-up	No	---	---	---	---	No	---
SIA	Infrastructure set-up	Yes	Infrastructure set-up	Yes	No	No	Yes	Infrastructure set-up
ACCESS	Infrastructure Set-up	Yes	Platform Deployment	Yes	No	No	No	---

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

MGMT	Tenant Set-up	Yes	Infrastructure Set-up	Yes	No	No	Yes	Tenant Set-up
CLUSTER	Platform Deployment	Yes	Platform Deployment	Yes	Yes	Yes	No	---
PS	Platform Deployment	Yes	Platform Deployment	Yes	No	No	No	---
MSP	Platform Deployment	Yes	Platform Deployment	Yes	No	No	Yes	Tenant Set-up

6.1.3 Infrastructure and Platform Descriptors

As previously mentioned, ARDENT requires an infrastructure descriptor as an input which allows the infrastructure provider to communicate the essentials about its environment to the platform provider, i.e.:

- SDN switching fabric and their topology
- NFV compute nodes and their available resources to the platform provider (CPU, RAM, Disk)
- Infrastructure services such as DNS and infrastructure IP GWs to be used by the platform
- Infrastructure networks and security groups for the platform provider

Once the infrastructure tenant has been set up successfully through ARDENT a platform descriptor is generated which describes the desired orchestration of the platform. The platform descriptor is a YAML-based HEAT template that is tuned towards OpenStack's orchestrator and how OpenStack abstracts networks and security groups. The generation platform descriptor is based on some best current practises when it comes to compute nodes, their compute capabilities and where to place which platform functionality.

Even though the infrastructure in Barcelona and Bristol look severely different when it comes to network topology and compute capabilities, some important similarities can be discovered which helps the automation of generating the platform descriptor, i.e.:

- The closer a compute node is located to the edge, the smaller its capabilities are
- Each location in the city has one compute node
- Each compute node is placed into its own availability zone

Based on the assumptions and best current practices described above ARDENT is placing the core platform nodes such as PCE/SFEMC, ICW GW, CLMC and all platform services (DNS, DHCP and IP GW) to the Tier 1 compute node (data centre). All remaining resources (CPU, RAM and disk) in the Tier 1 compute node are given to the cluster instance.

Compute nodes with ACCESS networks are treated as Tier 3 edge computing nodes where one SR is placed serving the UE attaching to the ACCESS network and one SR for the Tier 3 cluster instance. The remaining resource is allocated to the cluster instance itself.

If there is a compute node which is not a Tier 1 or Tier 3 one ARDENT treats it as a Tier 2 environment and placed a single SR with a cluster there where the cluster again maxes out the available resources (minus the SR).

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

7 CONCLUSIONS

This document provides an update to the specifications for the FLAME platform and infrastructure, built on the first release in D3.3. We have removed the cornerstones for the architecture development, such as ecosystem, use cases and requirements due to their unchanged nature, focussing the update on the component-level architecture, its sub-components and the interfaces between them. We also included a separate infrastructure integration section.

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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

REFERENCES

[3GPP_SBA] 3GPP, "New SID for Enhancements to the Service-Based 5G System Architecture", S2-182904, 2018

[D3.3] D. Trossen et al. "D3.3: FLAME Platform Architecture and Infrastructure Specification V1", FLAME deliverable D3.3, 2017

[KYLIN] http://kylin.apache.org/

[LENS] https://lens.apache.org/

[MPEGDASH] http://mpeg.chiariglione.org/standards/mpeg-dash

[NFV] ETSI, Network Functions Virtualisation, available at http://www.etsi.org/technologies-clusters/technologies/nfv

[NFVMANO] ETSI GS NFV-MAN 001, "Network Functions Virtualization (NFV); Management and Orchestration", V1.1.1, available at http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf, 2014

[OASIS13] Topology and Orchestration Specification for Cloud Applications (TOSCA). Version 1.0. 23 November 2013. OASIS Standard. available at http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf

[OpenFlow] OpenFlow, available at https://www.opennetworking.org/sdn-resources/openflow

[Openstack] Openstack website, available at https://www.openstack.org/

[POINT] H2020 POINT project, available at https://www.point-h2020.eu/, 2017

[Pur18] D. Purkayastha, A. Rahman, D. Trossen, "Applicability of BIER Multicast Overlay for Adaptive Streaming Services", IETF draft, available at https://tools.ietf.org/html/draft-purkayastha-bier-multicast-http-response-01, 2018

[Reed16] Martin J. Reed, Mays Al-Naday, Nikolaos Thomos, Dirk Trossen, George Petropoulos, Spiros Spirou, Stateless multicast switching in software defined networks, In proceedings of ICC 2016, Kuala Lumpur, Maylaysia, 2016

[SDN] Open Networking Foundation, Software-defined Networking (SDN) Definition, available at https://www.opennetworking.org/sdn-resources/sdn-definition

[SFC] IETF Service Function Chaining Working Group, available at https://datatracker.ietf.org/wg/sfc/about/

[SFC7665] IETF, "Service Function Chaining (SFC) Architecture", RFC 7665, October 2015

[TM17] TM Forum, "Performance Management API REST Specification (TMF628) R14.5.1", 2017, available at https://projects.tmforum.org/wiki/display/API/Performance+Management+API+REST+Specification+%28TMF628%29+R14.5.1

[Trossen18] D. Trossen, D. Purkayastha, A. Rahman, "Name-Based Service Function Forwarder (nSFF) component within SFC framework", IETF draft, available at https://tools.ietf.org/html/draft-trossen-sfc-name-based-sff-01, 2018

© Copyright InterDigital Europe, University of Southampton and other members of the FLAME Consortium 2018

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