FACILITY FOR LARGE-SCALE ADAPTIVE MEDIA EXPERIMENTATION

FLAME Service Delivery Platform

Sebastian Robitzsch

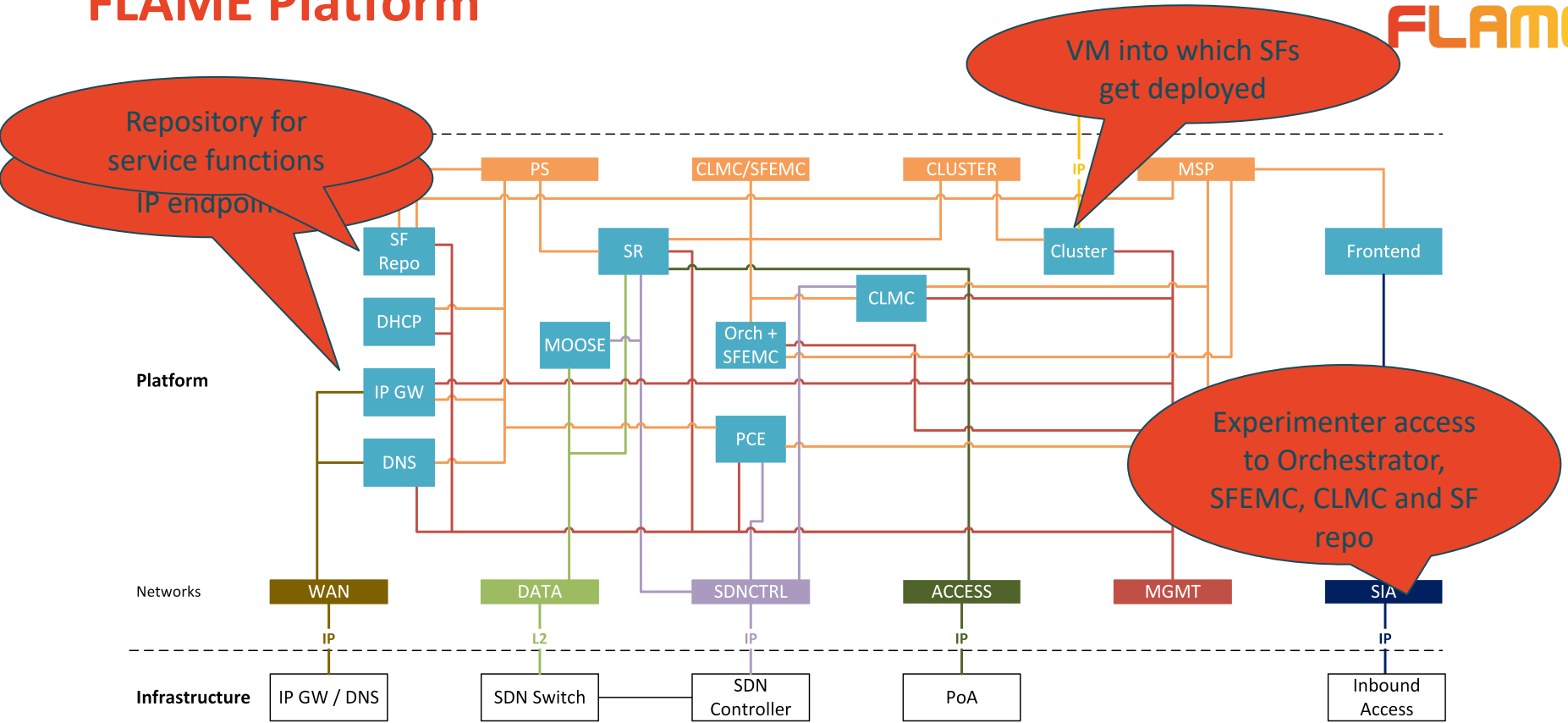
InterDigital Europe

Urban Hacking in 5G

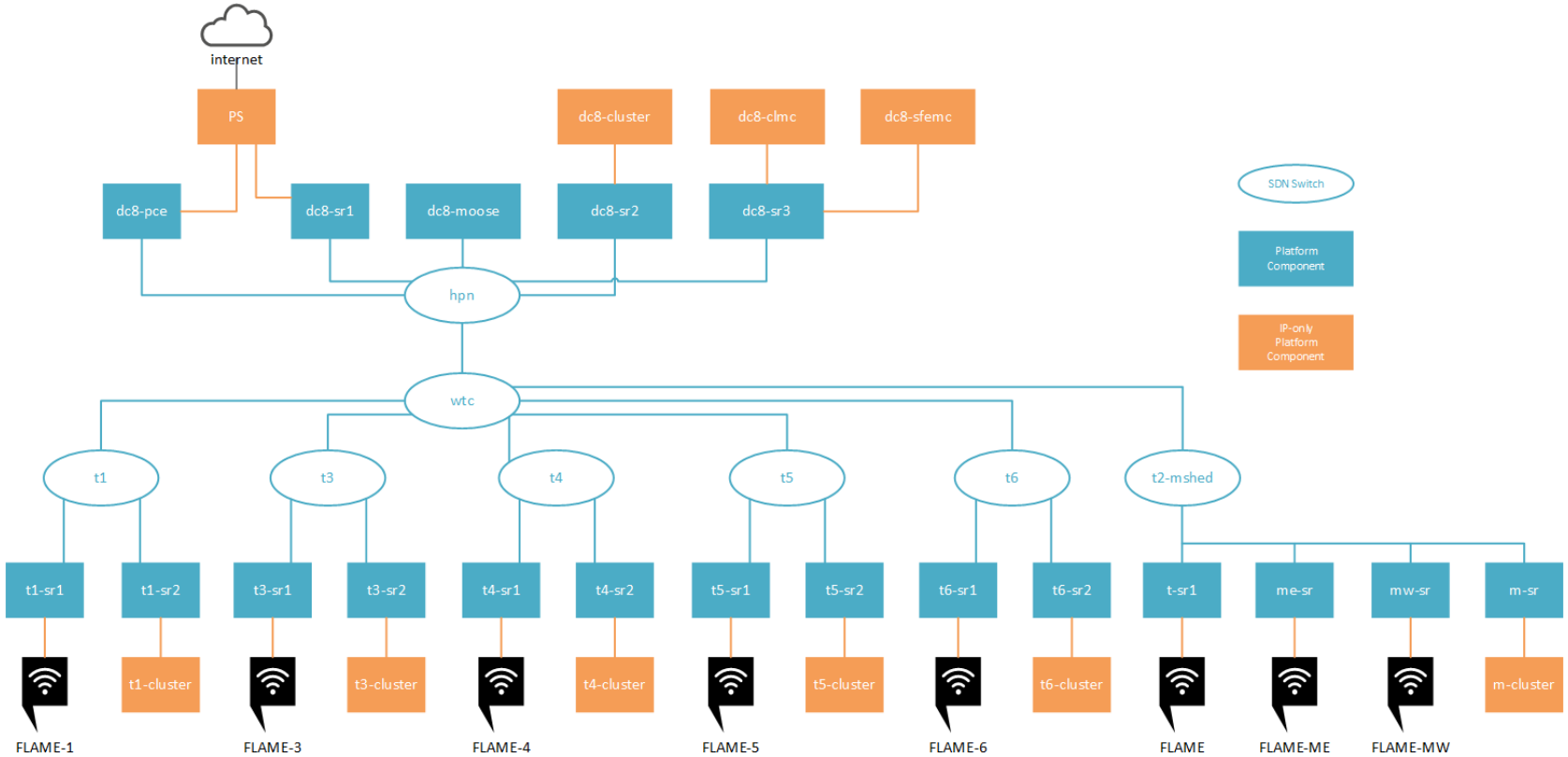
Bristol, 17 June 2019

FLAME Platform

FLAME Platform



FLAME Platform in Bristol



Foundational Media Services

- FLAME identified a set of services which are commonly used by service providers
- FMS are pre-packaged SFs uploaded to the platform's SF repository
- All FMS offer a RESTful API utilising the advances of the platform for HTTP-based services

Foundational Media Services

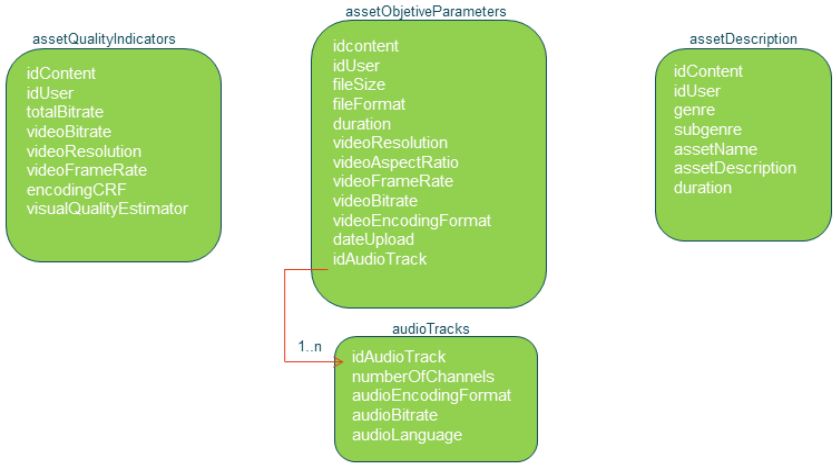
- Set of commonly used services provided to experimenter:
  - Content ingest and storage
  - Adaptive streaming
  - Transcoding, transrating and content conditioning
  - Media quality analysis
  - Proxy storage

Content Ingest and Storage

- Module for ingestion, storage and retrieval of assets
- Common functionality required by media service providers
- It includes a simplified database to keep some information about the stored contents
- Working as content repository, interoperating with the rest of FMS

Metadata Database for Audio & Video Content

- Required by media services to store information about the content
- Schema by default and methods to charge a personalised one
- Technical parameters of the contents
- Spec for content description (e.g., asset name)



Adaptive Streaming

- Offers MPEG DASH and HLS
- Resolution and bitrate is automatically adapted depending on the network and receiver capabilities
- Current implementation conceived for video-on-demand

Transcoding, Transrating and Content Conditioning

- Transcoding: encoding process in a different format to increase compression
- Transrating: encoding process in the same format to increase compression
- Conditioning:
  - Transcoding / transrating at different bitrates
  - Segment alignment to enable switching between qualities
- AVC and HEVC encoding formats

Media quality analysis

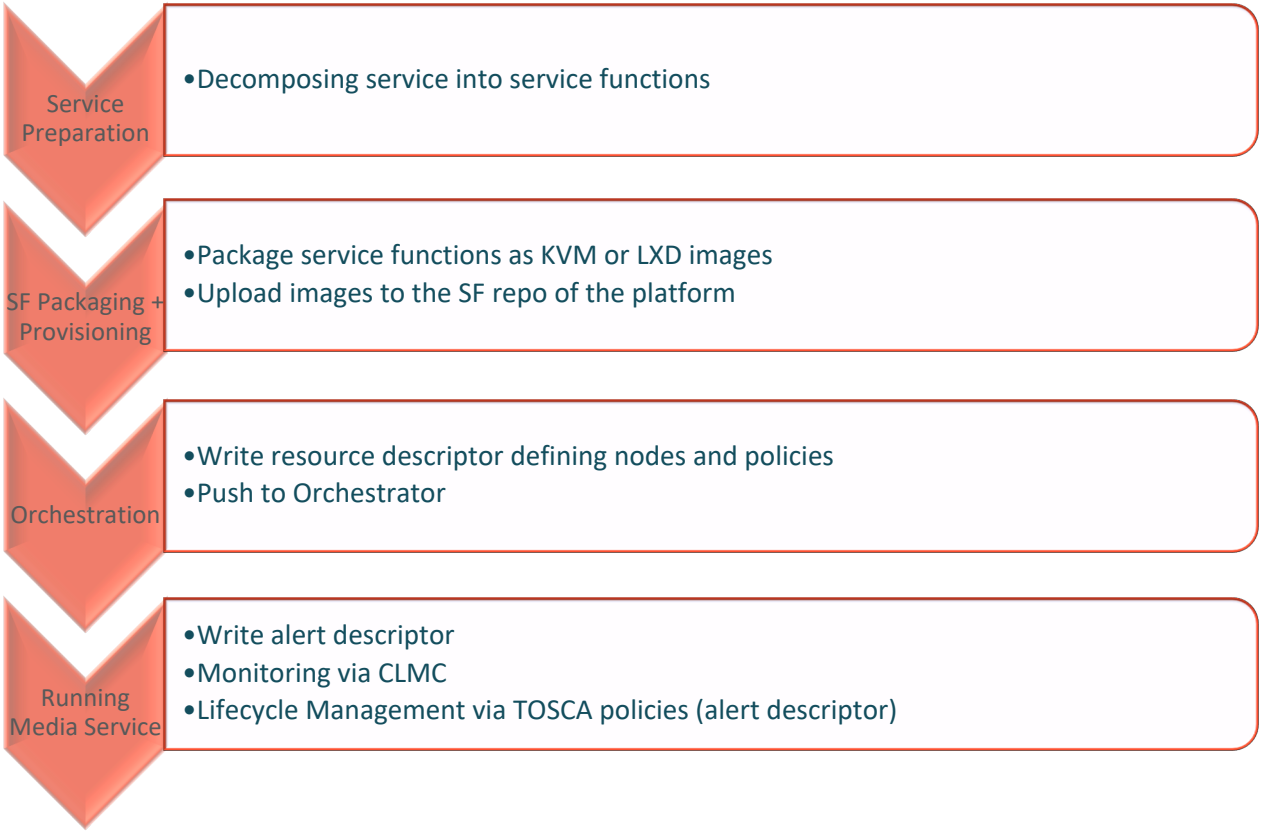
- Extraction of technical parameters
- First approach for quality estimation (e.g., video bitrate)
- Based on ffmpeg (ffprob)
- Consistent with metadata database

Proxy Storage

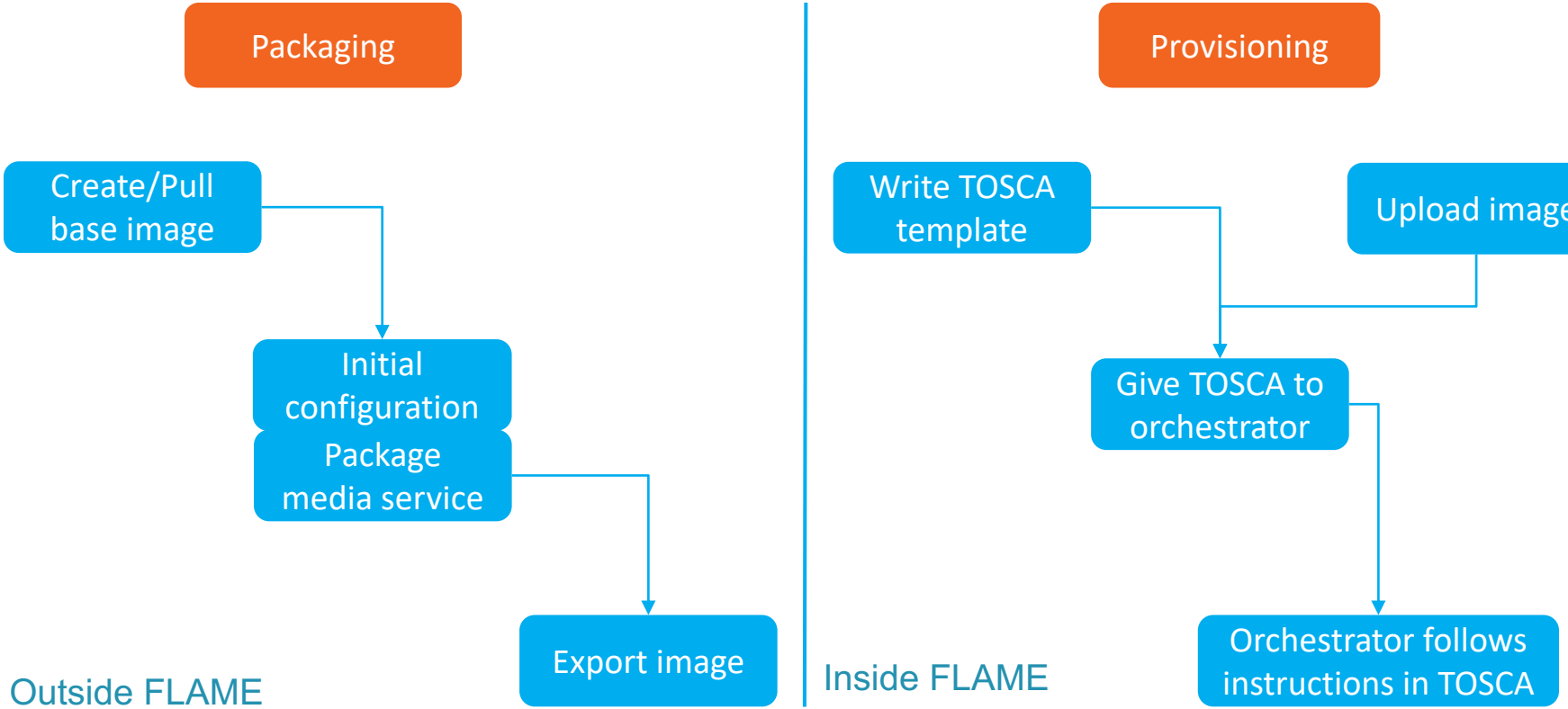
- Complements storage FMS
- 1 master storage and n proxy caches at edge
- If web resource is not available it is fetched from master
  - Otherwise, resource is served from proxy cache
- Comes with default caching strategies

Experimentation API

Workflow



Packaging and Provisioning Workflow



TOSCA in FLAME

- TOSCA is used to communicate desired *resource* orchestration
- FLAME TOSCA is TOSCA compliant: only new types are defined on top of TOSCA NFV standard
- Node and policy types specified to define service deployment and lifecycle of service function endpoints

Adoptions from TOSCA

Item	Relevant for FLAME
Node Templates	✓
Groupings	✓
Relationships	✗
Policies	✓
(Build-)Plans	✗

FLAME Definitions in TOSCA - Nodes

- FLAME defined their own type of SF Endpoint type
- Applies for every node which can be managed by SFEMC:

eu.ict-flame.nodes.ServiceFunction

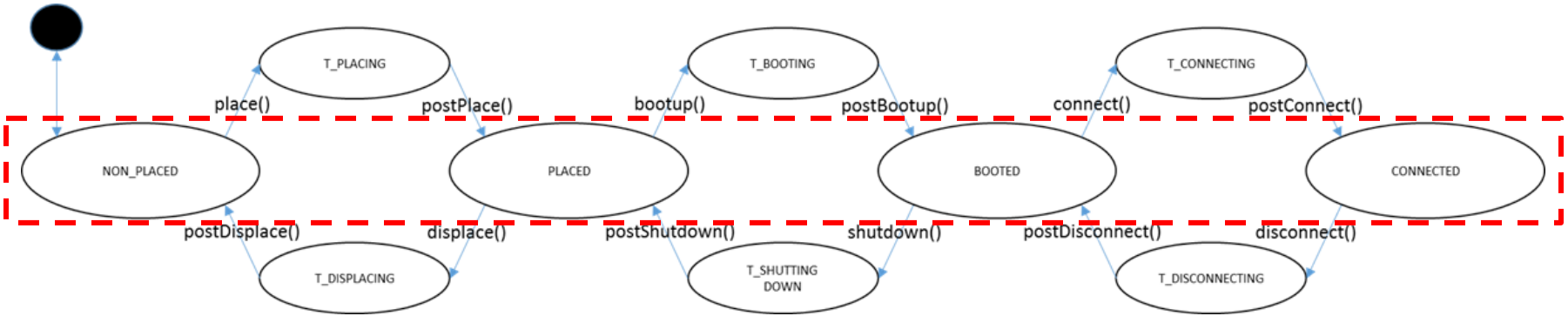
- Within this element, properties are specified such as
  - Addressable FQDNs (SF Identifiers)
  - Hypervisor-type (KVM or LXD)
  - URL from where the packaged SF can be retrieved
  - SF cloud properties (vCPUs, RAM, disk)

FLAME Definitions in TOSCA - Policies

- TOSCA Policies are a type of requirement that govern use or access to resources which can be expressed independently from specific applications
- We have defined two types of policies so far:
  - Initial Policy: A special policy type and is defined as **eu.flame.policies.InitialPolicy** and these labeled policies shall be processed only at the beginning of the run-time of the deployment. It determines the first targeted state for the deployed nodes.
  - State Change Policy: The second policy is defined as **eu.flame.policies.StateChange** and represents the FLAME-specific policy templates. Within this type of policies, we enhanced the policy's elements with a time element which allows scheduling of the policy (i.e., when the policy is active).

Lifecycle Management

- States are self-defined lifecycle keywords for the work with the State Machine inside the SFEMC. Allowed target states are:
  - eu.ict-flame.sfemc.state.lifecycle.connected: Push the Endpoint to CONNECTED state,
  - eu.ict-flame.sfemc.state.lifecycle.booted: Push the Endpoint to BOOTED state,
  - eu.ict-flame.sfemc.state.lifecycle.placed: Push the Endpoint to PLACED state



Resource Descriptor Example

```
topology_template:

policies:
    - init:
        type: eu.ict-flame.policies.InitialPolicy
        description: Start the nodes initially
        triggers:
          inital_trigger:
            condition:
              constraint: initialize
            action:
              protest-service:
                dc8-sr3-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.connected
                m-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
                t1-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
                t3-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
                t4-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
                t5-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
                t6-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.placed
```

Resource Descriptor Example

```
policies:
    - init:
        type: eu.ict-flame.policies.InitialPolicy
        description: Start the nodes initially
        triggers:
          inital_trigger:
            condition:
              constraint: i
            action:
              protest-serv
                dc8-sr3-cl
                m-sr1-clu
                t1-sr1-cl
                t3-sr1-cl
                t4-sr1-c
                t5-sr1-c
                t6-sr1-c
- high_latency_check:
        type: eu.ict-flame
        description: Check
        triggers:
          check_trigger:
            description: C
            condition:
              constraint:
              period: 600
            action:
              protest:
                m-sr1-clus
```

```
- high_latency_check:
        type: eu.ict-flame.policies.StateChange
        description: Check Latency and peform a connect of another node …
        triggers:
          check_trigger:
            description: Check high latency on relationships
            condition:
              constraint: clmc::service_latency_exceeded
              period: 600 # integer required, unit: seconds
            action:
              protest:
                m-sr1-cluster1-cluster: eu.ict-flame.sfe.state.lifecycle.connected
```

Alert Descriptor Example

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: TOSCA Alerts Configuration document
imports:- flame_clmc_alerts_definitions.yaml
metadata:
  servicefunctionchain: companyA-Vr

topology_template:
 policies:
    - high_latency_poli
        type
        trigg
          serv
            des
            even
            metri
            condit
              thres
              granul
              aggrega
              resource
                flame_l
              comparis
            action:
              implement
                - flame
                - http:
```

```
- high_latency_policy:
      type: eu.ict-flame.policies.Alert
      triggers:
        service_latency_exceeded:
          description: This event triggers when the mean network latency …
          event_type: threshold
          metric: network.latency
          condition:
            threshold: 45
            granularity: 120
            aggregation_method: mean
            resource_type:
              flame_location: m-sr1-cluster1-cluster
            comparison_operator: gt
          action:
            implementation:
              - flame_sfemc
              - http://companyA.alert-handler.ict-flame.eu/high-latency
```

FLAME

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