FACILITY FOR LARGE-SCALE ADAPTIVE MEDIA EXPERIMENTATION

FLAME Platform Architecture

Sebastian Robitzsch & Kay Hänsge

InterDigital Europe, Ltd

FLAME Summer School

Bristol, 11.07.2018

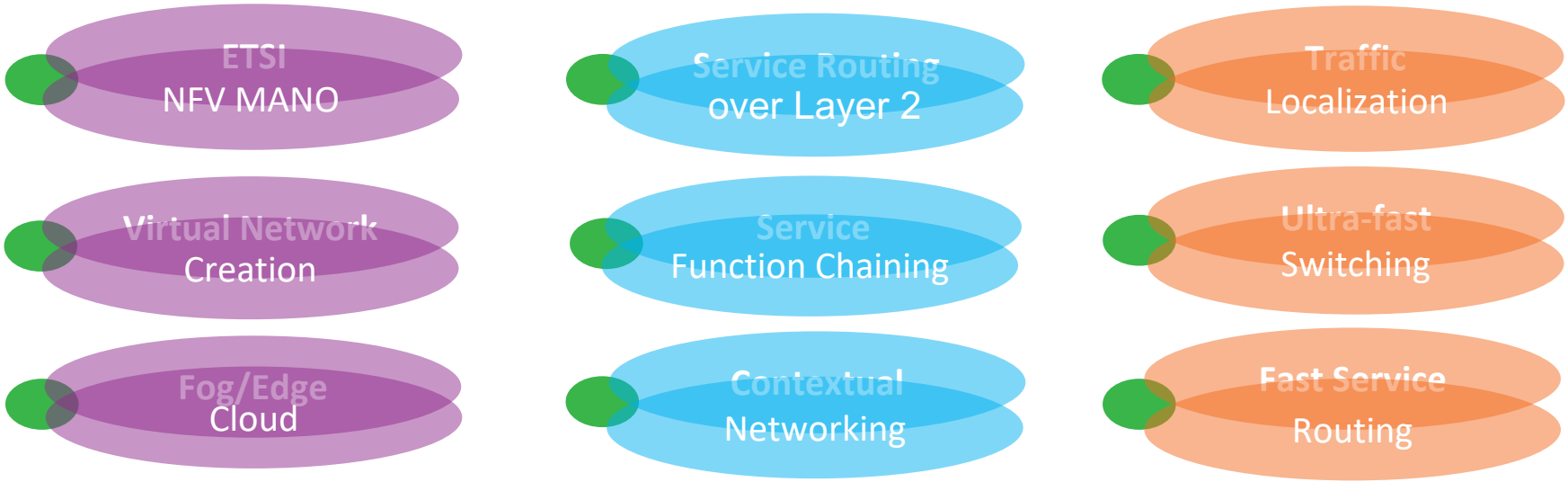
Outline

- Background
- FLAME service delivery platform
- Experimentation API
- Hands-on

Background

5G Networks: Observed Trends

Edge Deployments	Virtualization	Converged Layer2	Service-based Architectures for control & user plane	5mSec	From 100 Mbps To 1 Gbps
Buildings, cars, street furniture, …	Slice network & compute resources	Programmable Networks	Interpret anything as a (Internet) service	E2E Latency	User Experienced Data Rate



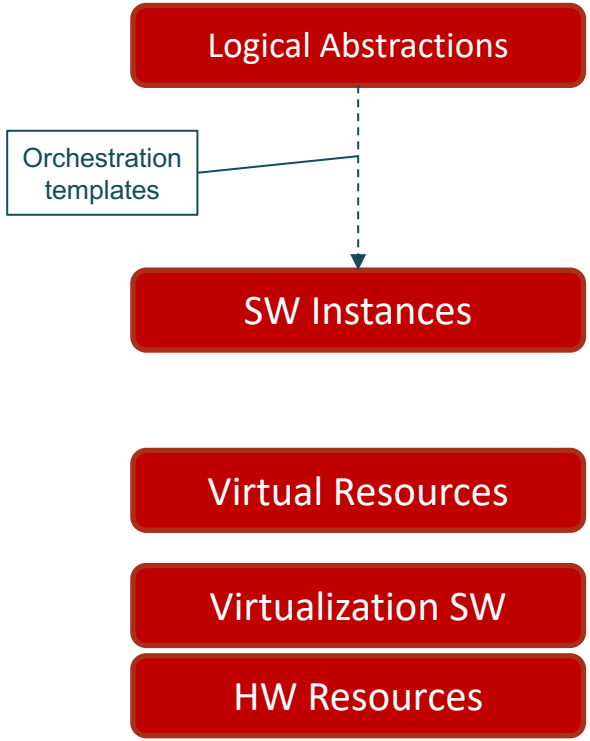
Effects through Virtualization

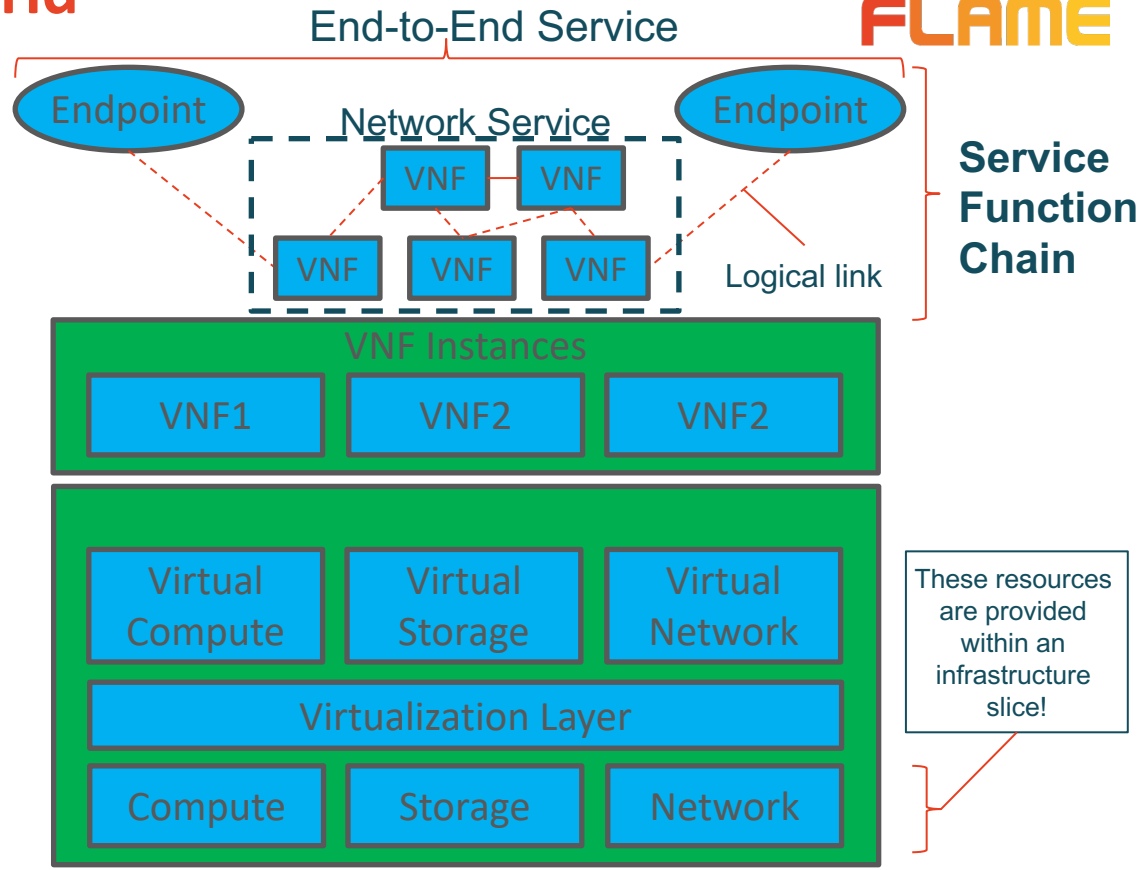
- **Availability:** virtualized compute entities deployed, replicated, collaborating in the edge network
  - Service just one hop away could be achieved

- **Instantiation**: happens in short time scales
  - Use of containers can push this into seconds timescale

- **Synchronization**: service replicas might have **incomplete** information
  - Rely on backend synchronization (e.g., for content, state, …)

The Emerging Telco World





FLAME Service Delivery Platform

Technical Proposition of Our Platform

FAST, ADAPTIVE

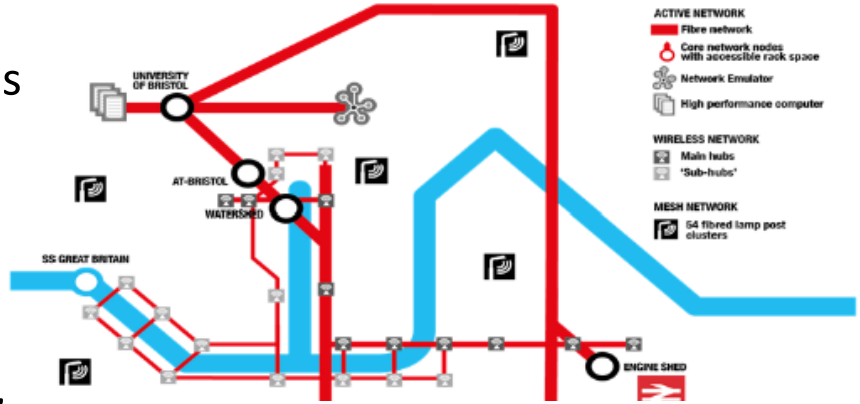
- **Faster response, better engagement**
  - service deployment at the edge of the network (e.g. in a street cabinet)
  - compute located just one hop away (at best) from the users, low latency access
  - compute workload distributed across the network
- **Improved service request routing**
  - fast (between 10 and 20ms) switching time from one service instance to another by not relying on the DNS.
  - overcomes inefficient ‘triangular’ routing of requests in current IP networks
- **Multicast delivery of http responses**
  - multicast-based delivery of HTTP responses to service request transparently to the (otherwise unicast) semantic of HTTP transactions.

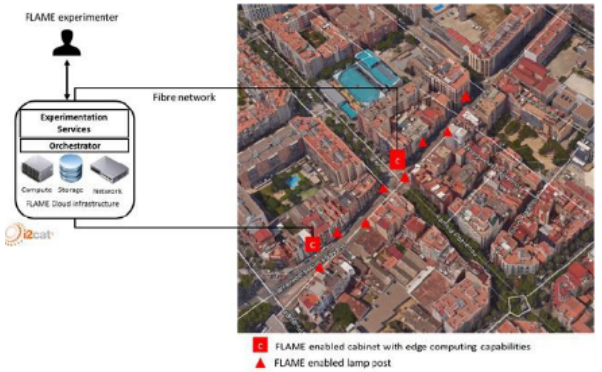
ROBUST, SECURE

- **Net-level indirection**
  - indirection of service requests at the network level allowing error response to redirect the original request to another alternative surrogate
  - nesting operations leads to a net-level ‘search’ among all available surrogate instances
- **Less chance of insecure direct object references**
  - CDNs morph into surrogate service endpoints with the potential to hold the necessary security context when serving the desired content
- **Secure end-to-end access to content**
  - CDNs deployed as properly secured endpoints with the necessary certificate sharing between content
  - Securing content delivery according to the originally intended end user facing contract -more secure for provider and consumer.

Validated through Urban Scale Trials & Experiments

- Validate platform capabilities through trials conducted by ecosystem partners
  - 5 operator infrastructures
  - 25+ customer trials
- New media formats (AR, VR, 360) and distribution channels
- Engagement with media service providers, content providers, infrastructure operators and beyond
- Trials will be conducted in 3 waves from Mar-18 to Dec-19
- Public funding available through H2020 FLAME project





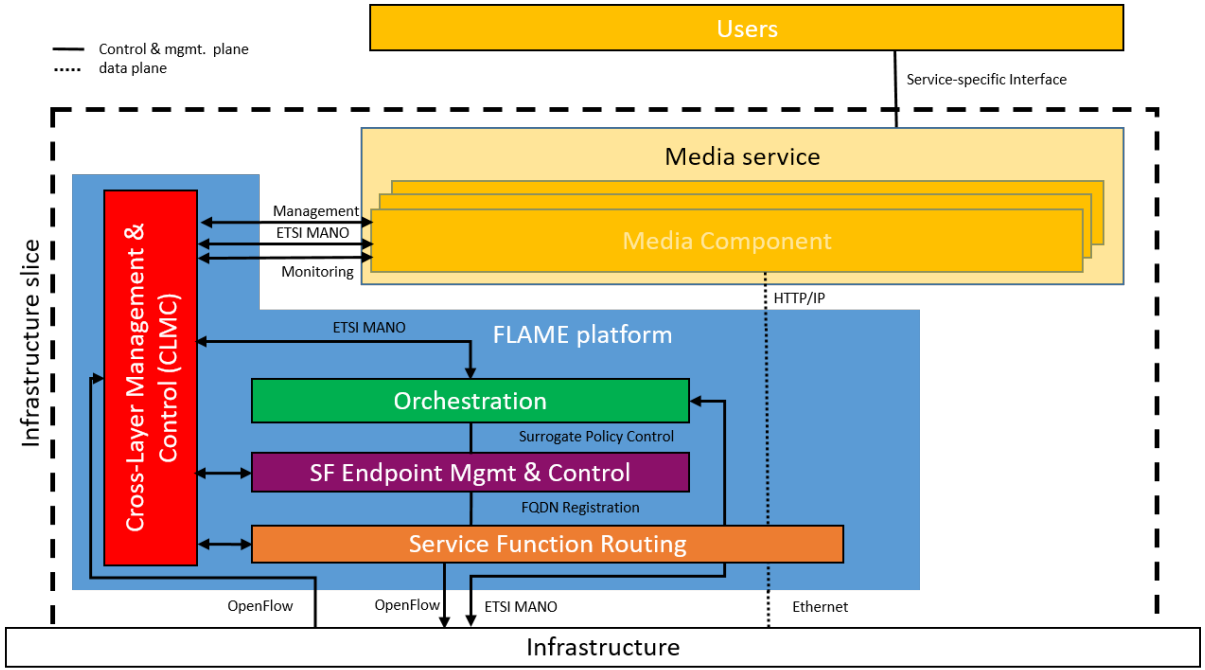
Basic Concepts for Platform Design

- Capture service interactions based on a **service function chain (SFC)**, i.e., serialized execution of service functions
  - For notation utilize concepts introduced in https://tools.ietf.org/html/rfc7665
    - Service functions (SF)
    - Service function forwarders (SFF)
    - Service Encapsulation (SE)

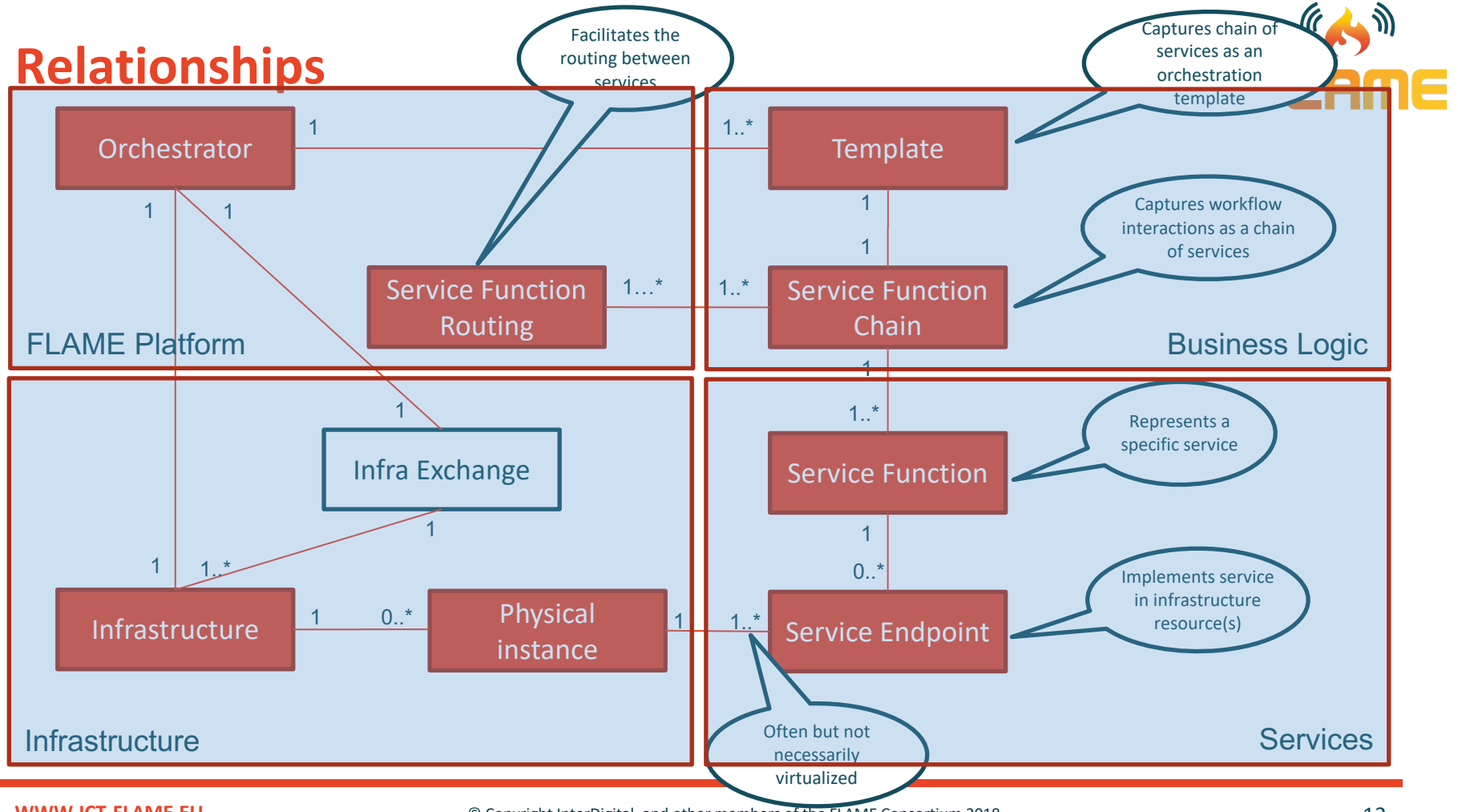
    NOTE: an SFC includes all three aspects above, i.e., is fully defined by the SFs interacting, the message exchange between them (SFF) and the encapsulation being used (SE)
  - SFCs are usually captured through 'templates'
  - Concepts are applied across the entire FLAME platform
- SFCs are deployed through a process of orchestration, deploying and managing compute, storage and connectivity resources

Architecture View

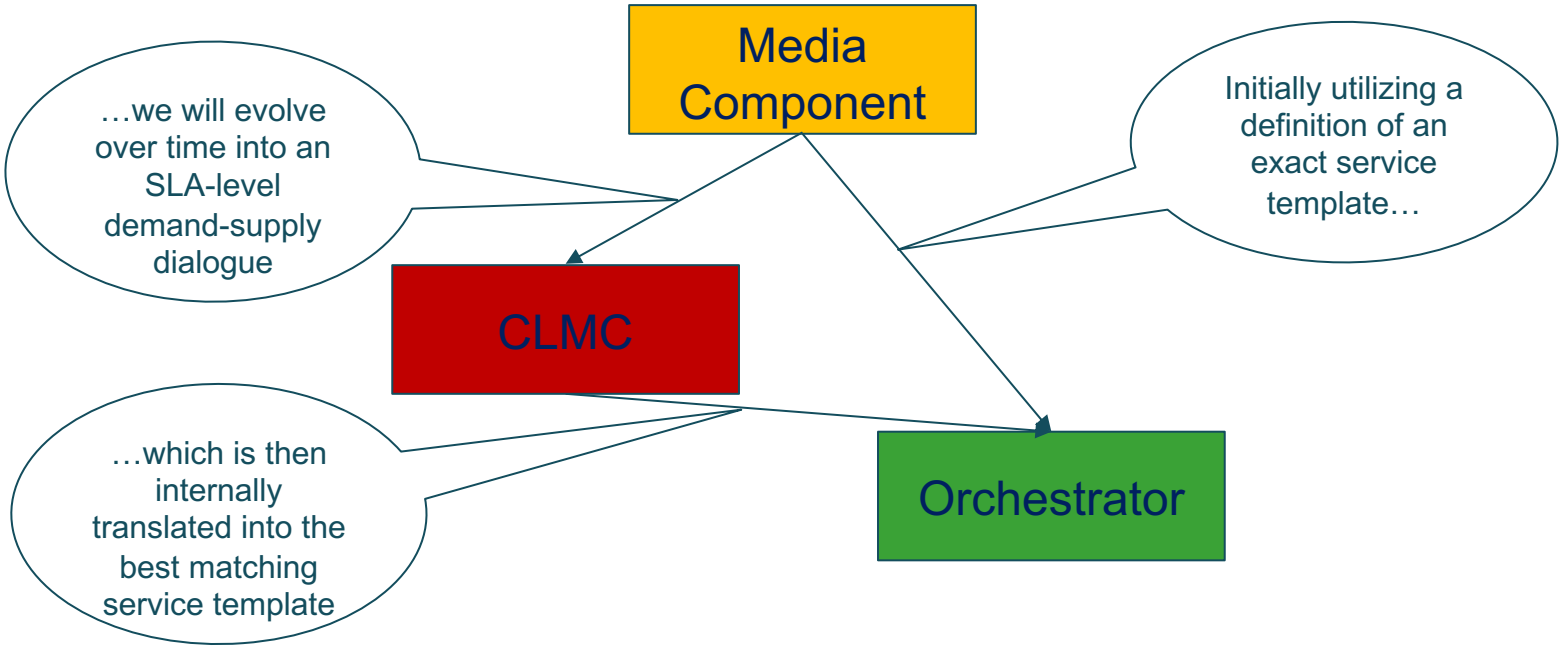
- **A new dynamic service delivery platform**
  - Deployed in minutes over a municipal scale infrastructure
  - Support for (among others)
    - many points of presence of NFV infra
    - fast service-level indirection in millisecond range
    - multicast delivery of HTTP-based video, e.g., in VR/AR
    - direct path mobility
- **Supporting enhanced Quality of Experience**
  - Targets personalised, interactive, mobile and localised media services



Relationships

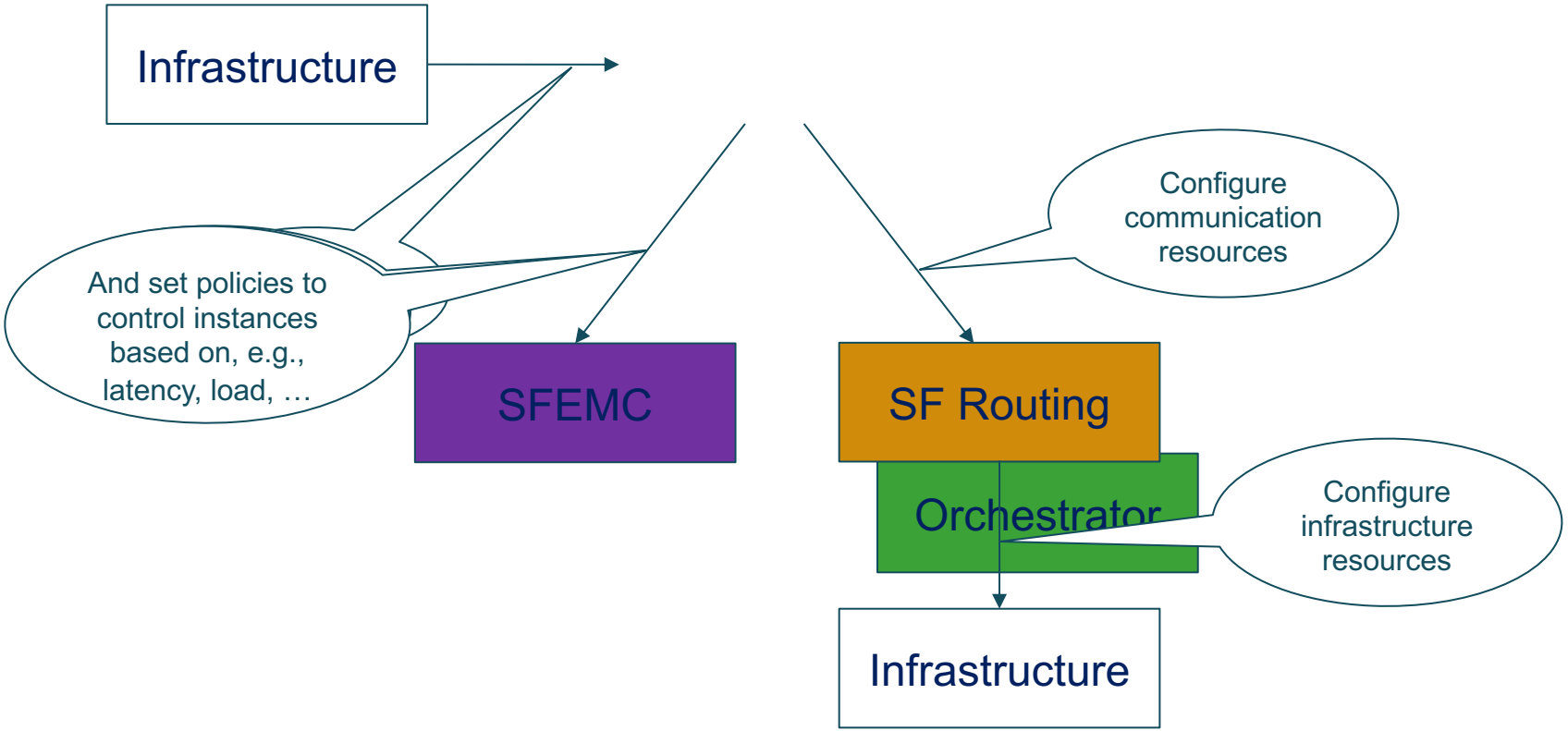


An Increasingly Rich Dialogue between Experimenter & Platform

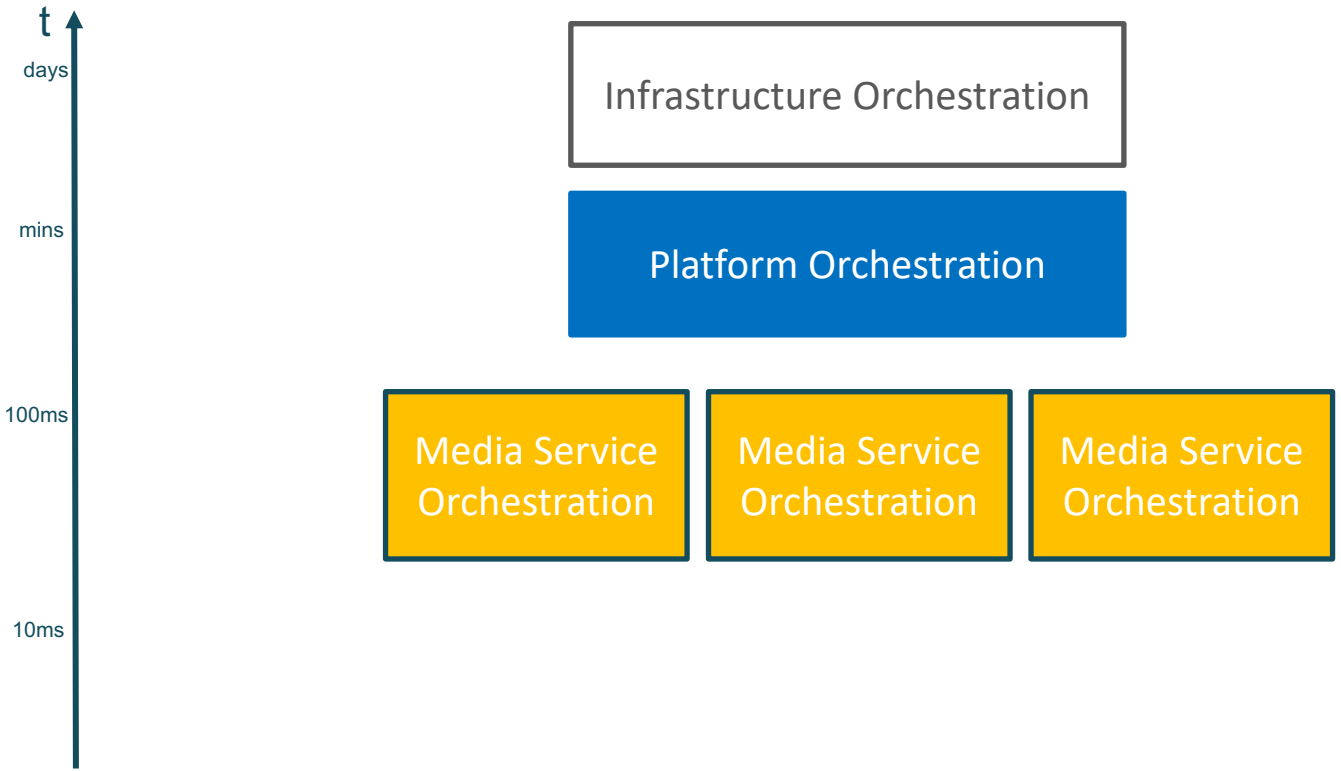


Will show later this part of the Experimentation API

Supported by Flexible Management and Control

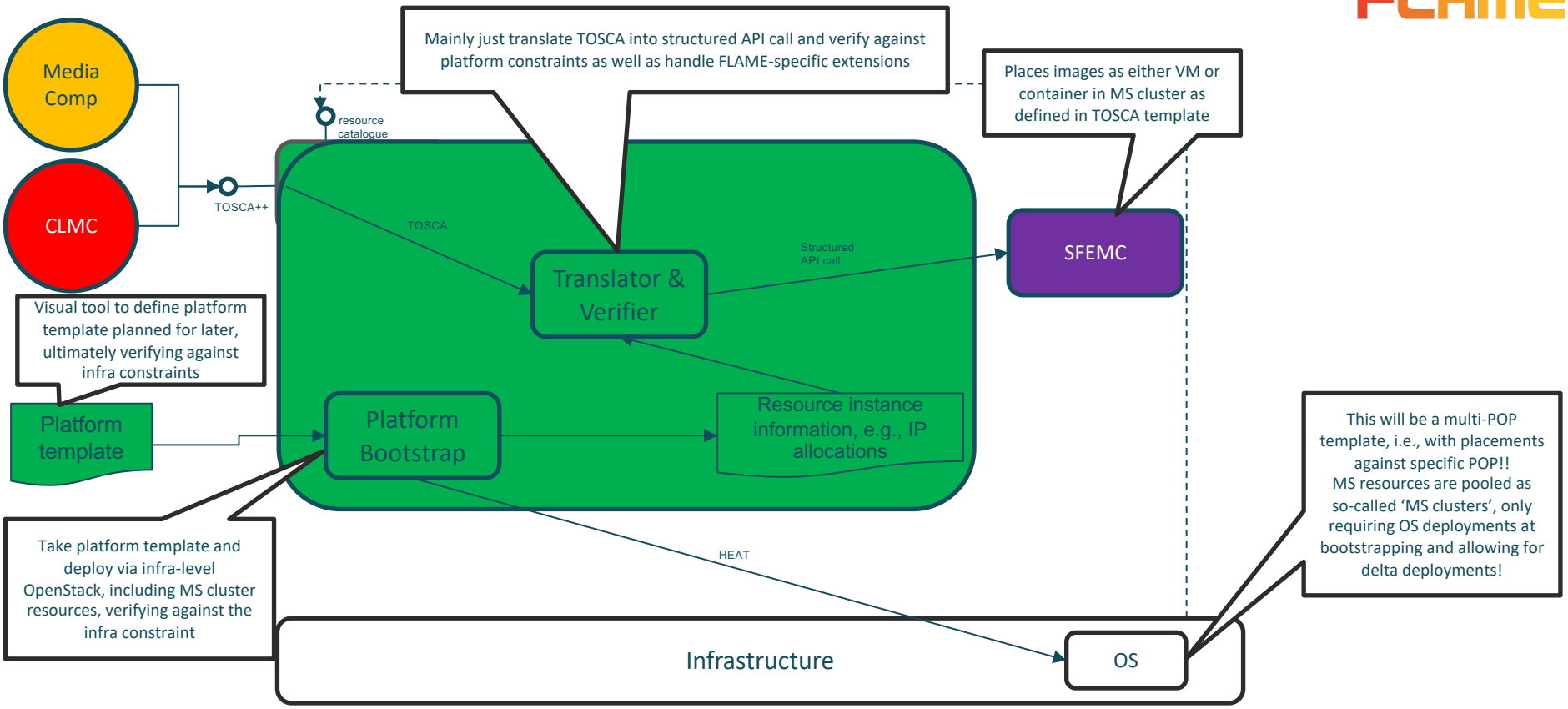


Orchestration Lifecycles of the Overall System

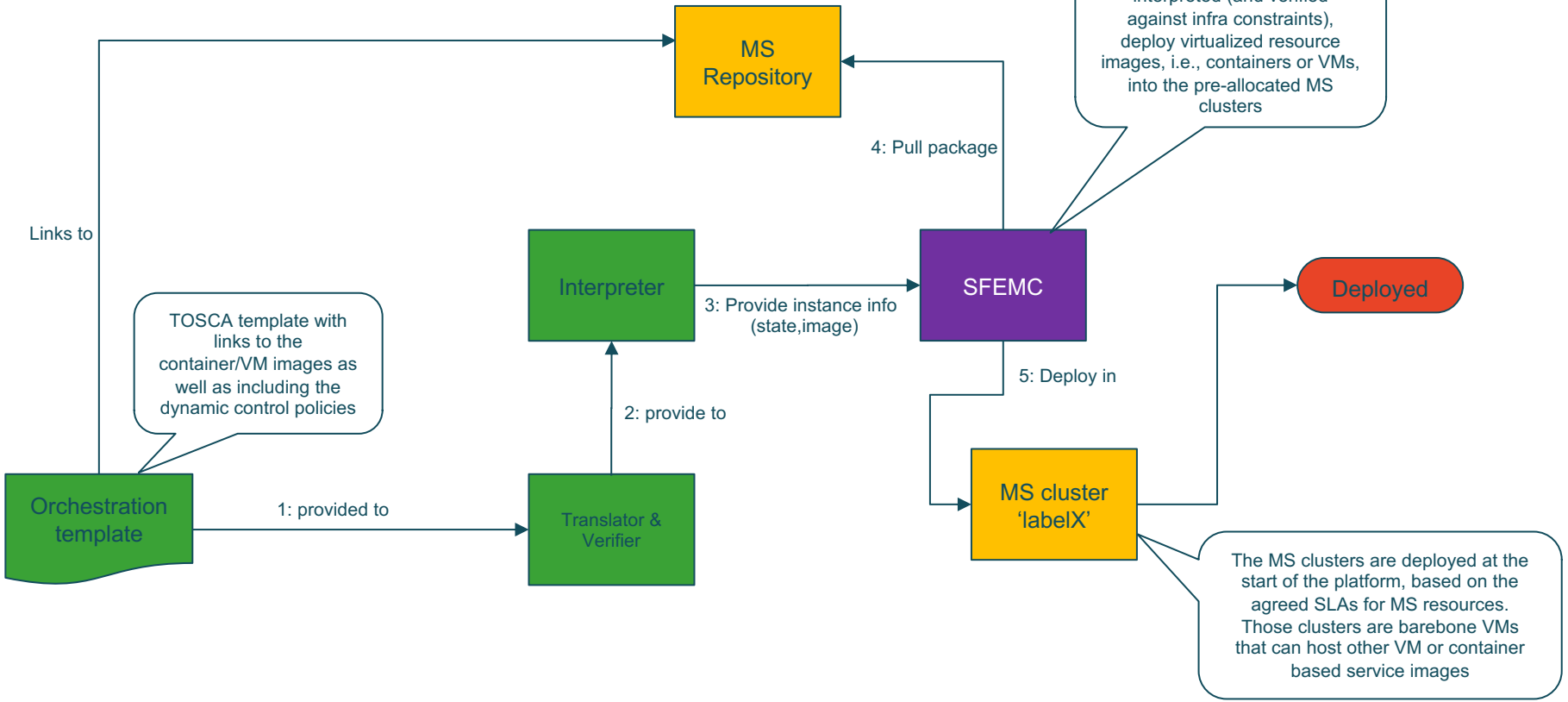


- Provide slice based on long-term relationship (infrastructure provider)
- Provision resource pool for long-term service offering (platform provider)
- Manage AND control service-specific resources (media service provider)

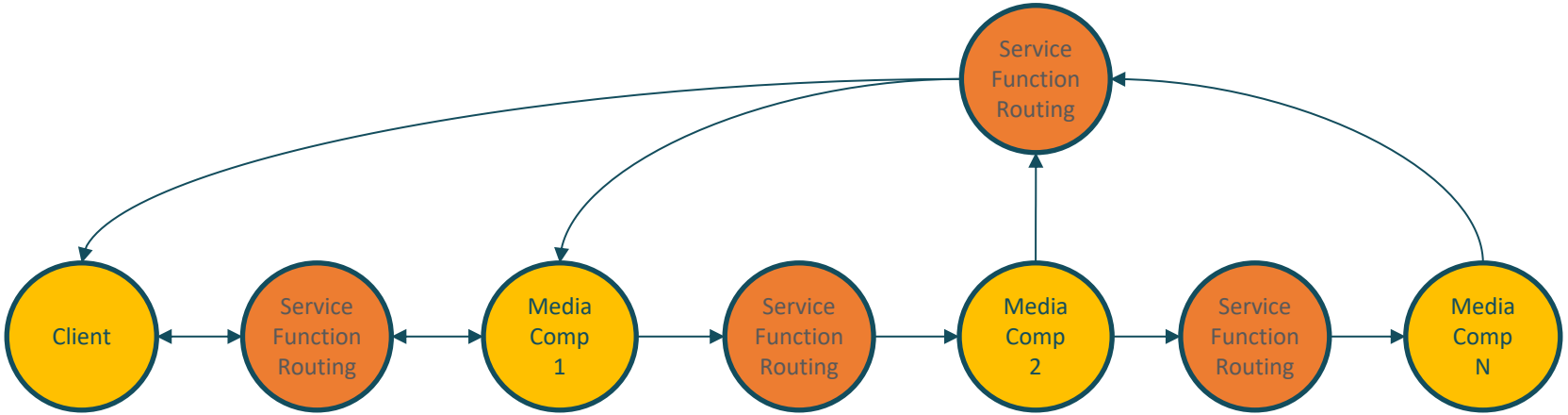
Orchestrator Design



Media Service Orchestration

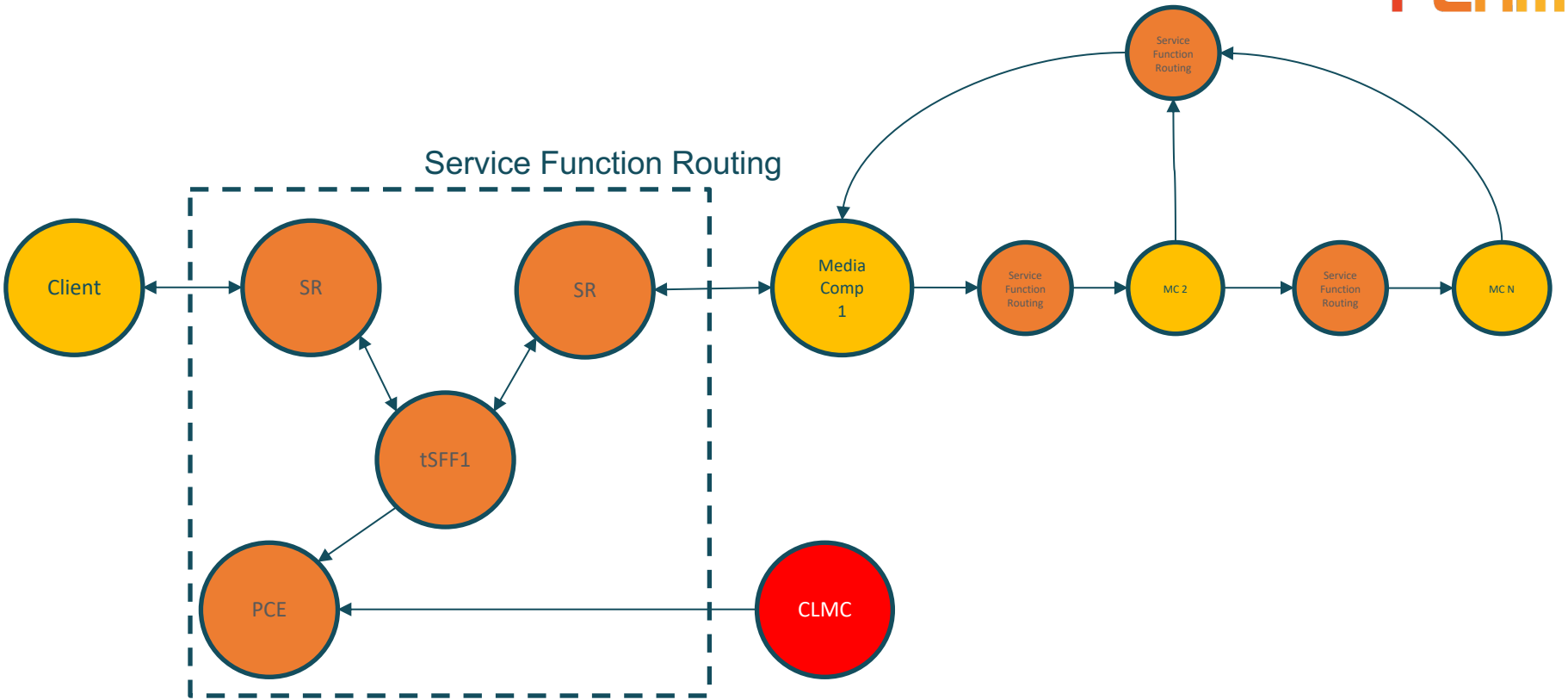


Main FMI Service Flow SFC

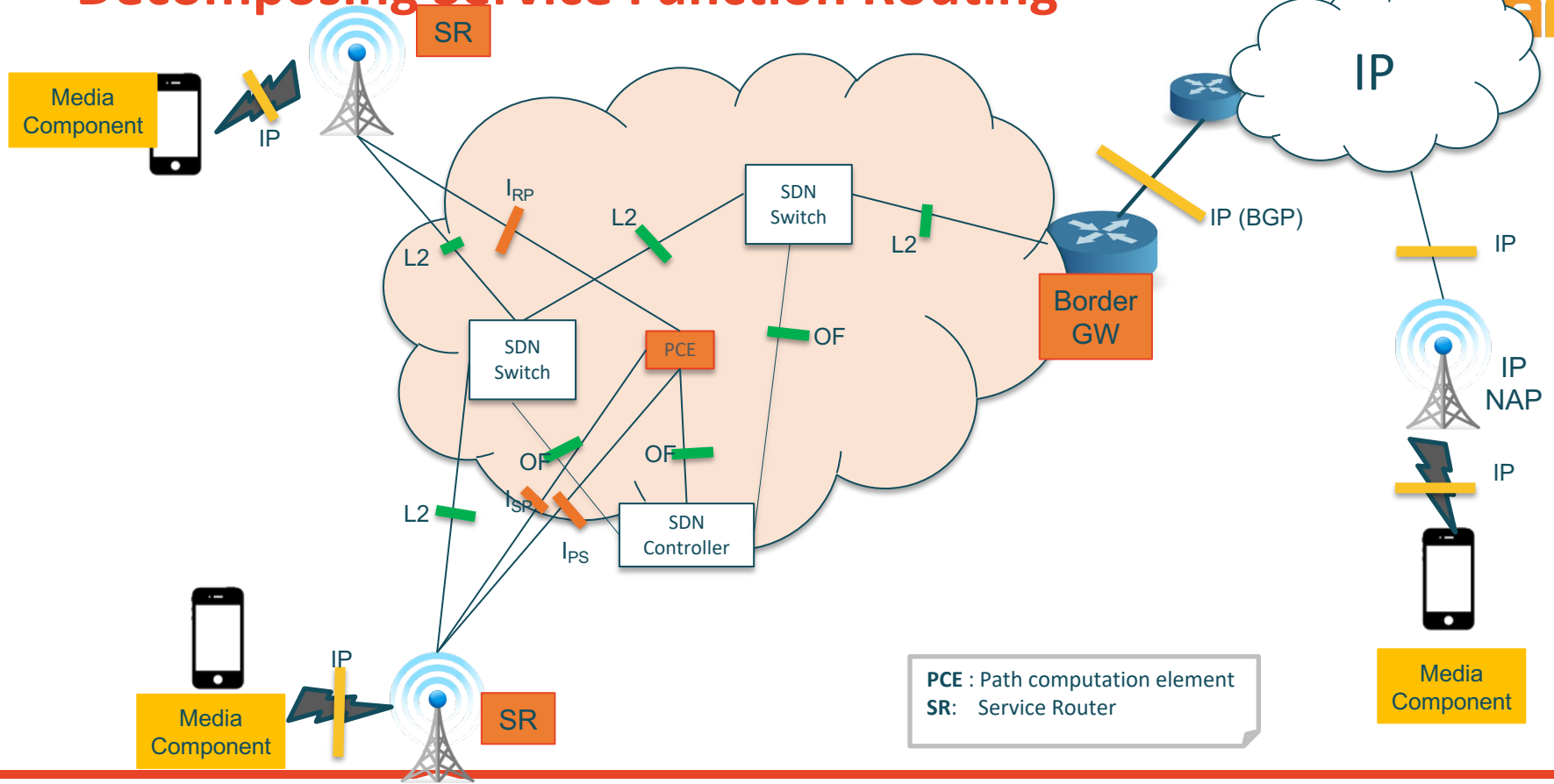


- Clients are connected to media functions through frontend routed network, while media functions are chained via service function routing function, the latter being realized as ‘fast switching’ SDN-based solution

Service Function Routing SFC



Decomposing Service Function Routing



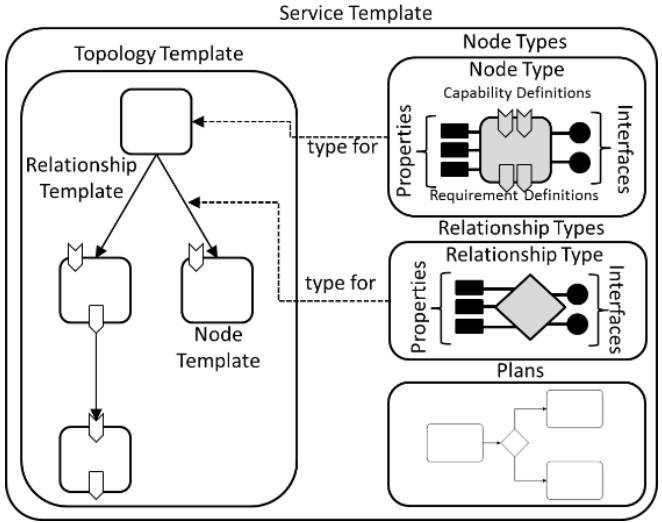
Experimentation API

Introduction to TOSCA

- Abbreviation for
  Topology and **O**rchestration **S**pecification for **C**loud **A**pplications and provides a type system describing the blocks for constructing service templates
- TOSCA offers the possibility to define virtual instance templates and their nodes to be deployed using an Orchestrator.
- Characteristics of these machines can be defined in the template:
  - nodes with properties and their relationships,
  - Hardware and software requirements,
  - policies,
  - machine status,
  - deployment plans.

Basic Concepts of TOSCA

- Specification defines a metamodel for defining IT services; it defines the structure of a service and how to manage it.
- The Structure of a service is defined by the Topology Template (or Topology Model).
- Relationships define how Nodes may interact.
- Plans define the process models that are used to create, manage during its whole lifetime and terminate a service inside the runtime environment.



TOSCA in FLAME

- The idea is to define all the topics related with the network virtualization in terms of equipment, protocols, servers and others, available to be located in a data center or network.
- We use specified node and policy types to define our service deployment and lifecycle templates.
- We do not change the TOSCA standard, but we define our own types. Hence, we still standard compliant.

What we need in FLAME – and what not (so far)

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

FLAME Definitions in TOSCA - Nodes

- We have defined an own type of SF Endpoint type and applies for every node which can be managed/orchestrated by FLAME's orchestrator or the SFEMC:

eu.ict-flame.nodes.ServiceFunction

- Within this element, properties such as addressable FQDNs, hypervisor-type or the packaged image_url have to be specified.

FLAME Definitions in TOSCA - Policies

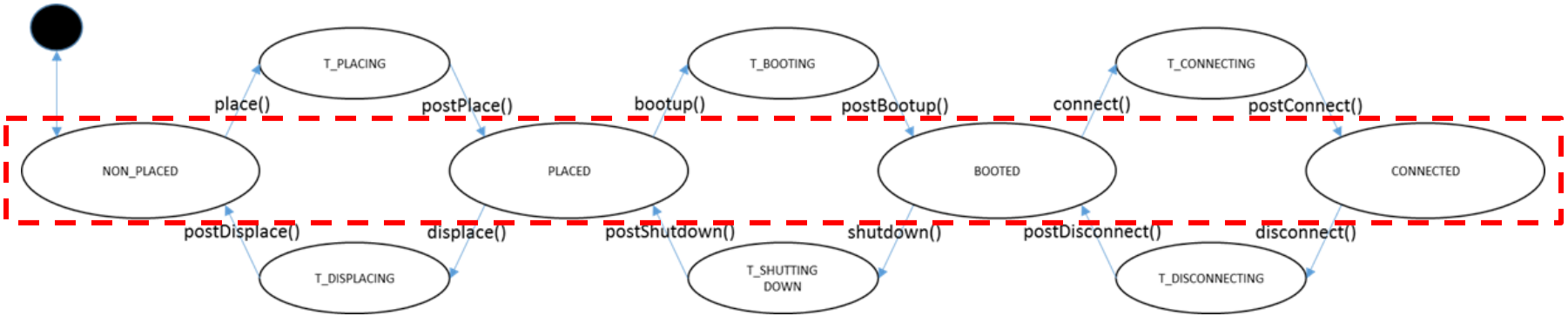
- TOSCA Policies are a type of requirement that govern use or access to resources which can be expressed independently from specific applications [...][1].
- We have defined two types of policies so far:
  - Initial Policy: A special policy type and is defined as **eu.flame.policies.InitialPolicy** and 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).
- The field *parent* may contain hierarchy information may reduced conflicting policies its result.

FLAME Definitions in TOSCA - Policies

eu.ict-flame.policies.{…}	Description
InitialPolicy	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.
StateChange	This policy represents the FLAME-specific lifecycle management. Within this type of policies, we allow the change of the lifecycle of a node on a given cluster/location.
InternetAccessPolicy	This policy specifies whether certain FQDNs are permitted (whitelisted), or prohibited (blacklisted). They apply on all nodes.

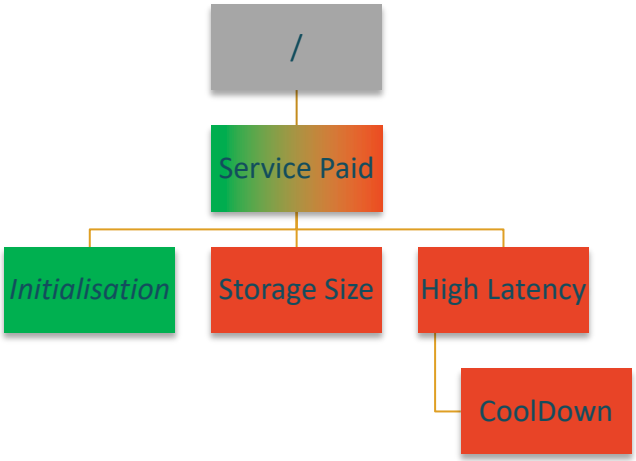
Lifecycle Management

- 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,
  - eu.ict-flame.sfemc.state.lifecycle.non_placed: Push the Endpoint to NON_PLACED state.



FLAME Definitions in TOSCA - Policies

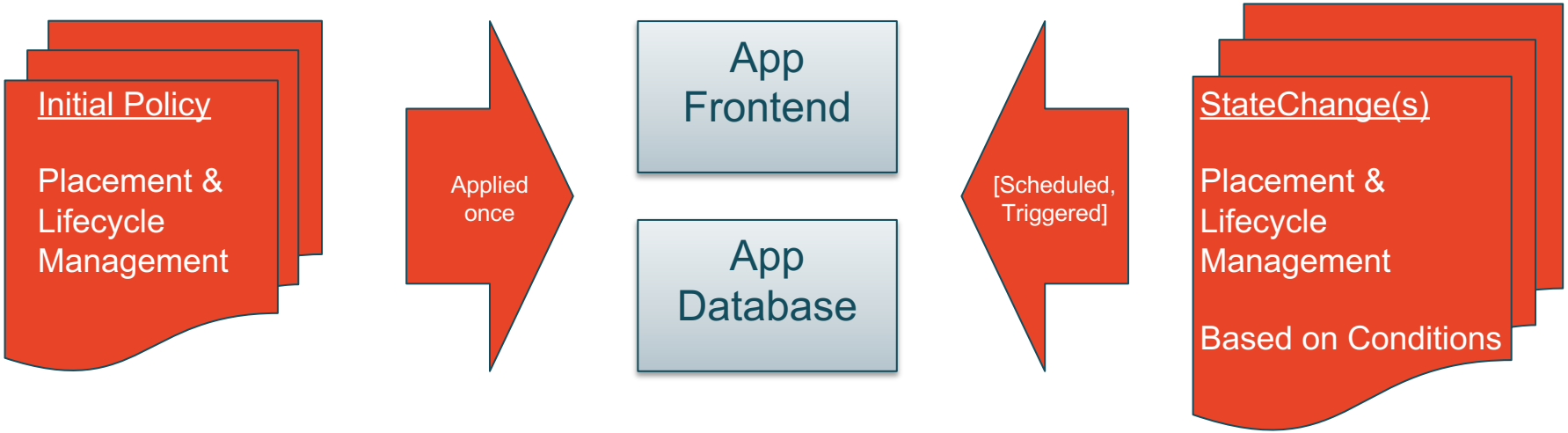
- Inside the policy description, the optional field *parent* contains hierarchy information.
- It reduces possible conflicting policies and their result.



SFEMC Deployment State:

Policies and their influence

EXAMPLE: APPLYING ON TWO MEDIA SERVICE NODES



Good to know

We have a git-repository for TOSCA-based Orchestration:

https://gitlab.it-innovation.soton.ac.uk/FLAME/flame-tosca

- /definitions – the FLAME TOSCA definitions,
- /documentation – Detailed documentation for the templating,
- /examples – examples to show certain use cases,
- /validator – validation software to test written templates against the definition file.

Stable versions are tagged: For any development, please use the latest tagged version.

Templating – General structure

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: |
  Template for deploying a service based on policies.
  When the delay of the deployed services is too high,
  another service endpoint will automatically be set
  to the CONNECTED state.
  However, only if the payment of the service has been received!

metadata:
  template_name: Flame Use Case High Latency
  template_author: Kay Haensge <kay.hansge@interdigital.com>, Sebastian Robitzsch <sebastian.robitzsch@interdigital.com>
  template_version: 0.1.6
  servicefunctionchain: high_latency_uc_sfc

# Import own definitions of nodes, capabilities and policy syntax.
imports:
  - flame_definitions.yml
```

Templating Nodes

```
database:
  type: eu.ict-flame.nodes.ServiceFunction
  capabilities:
    endpoint:
      type: tosca.capabilities.Endpoint
      properties:
        ports: # uses a map of PorSpecs
          mysql_port:
            protocol: tcp
            target: 3306
    host:
      properties:
        num_cpus: 2
        mem_size: 4096 MB
        disk_size: 10 GB
  properties:
    hypervisor: kvm
    image_url: http://cdimage.debian.org/cdimage/openstack/current-9/debian-9-openstack-amd64.qcow2
    fqdn:
      - db.app.ict-flame.eu
```

Templating Policies (Initial Policies)

```
- init:
    type: eu.ict-flame.policies.InitialPolicy
    description: Start the nodes initially
    properties:
     parent: service_paid
    triggers:
     inital_trigger:
      condition:
       constraint: initialise
      action:
       frontend:
        -
         fqdn: frontend.app.ict-flame.eu
         lifecycle_actions:
          London: eu.ict-flame.sfe.state.lifecycle.connected
        database:
        -
         fqdn: db.app.ict-flame.eu
         lifecycle_actions:
          London: eu.ict-flame.sfe.state.lifecycle.connected
```

Arrays of complex elements/structures

Templating Policies (State Change Policies)

```
- high_latency_check:
    type: eu.ict-flame.policies.StateChange
    description: Check Latency and peform a connect of another node when the latency is too high.
    properties:
     parent: service_paid
    triggers:
     check_trigger:
      description: Check high latency on relationships
      condition:
       constraint: clmc::service_latency_exceeded
      action:
       frontend:
        -
         fqdn: frontend.app.ict-flame.eu
         lifecycle_actions:
          Bristol: eu.ict-flame.sfe.state.lifecycle.connected
```

Templating Policies (Whitelisting external FQDNs)

- allowed_fqdns:
  type: **eu.ict-flame.policies.InternetAccessPolicy**
  description: Allow some typical external resources.
  properties:
    method: **permit**
    fqdn:
      - www.google.com
      - www.github.com
      - gitlab.it-innovation.soton.ac.uk

Validation - Installation

The validator allows to test test written templates against the definition file(s). The validator runs in a container to reduce the system impact.

Prerequisites:

- Preferable a Linux machine (e.g., Debian/Ubuntu distribution)
- Installed Software:
  - Docker to run the validator (e.g. for Ubuntu Linux: https://docs.docker.com/install/linux/docker-ce/ubuntu/)
  - Text-Editor, with YAML highlighting (Atom, Studio Code, or any higher IDE)

Validation – Download and First Run

Download:

- git clone https://gitlab.it-innovation.soton.ac.uk/FLAME/flame-tosca/

Run

- cd ./validator/
- ./test.sh -d <definition file> -t <template file>

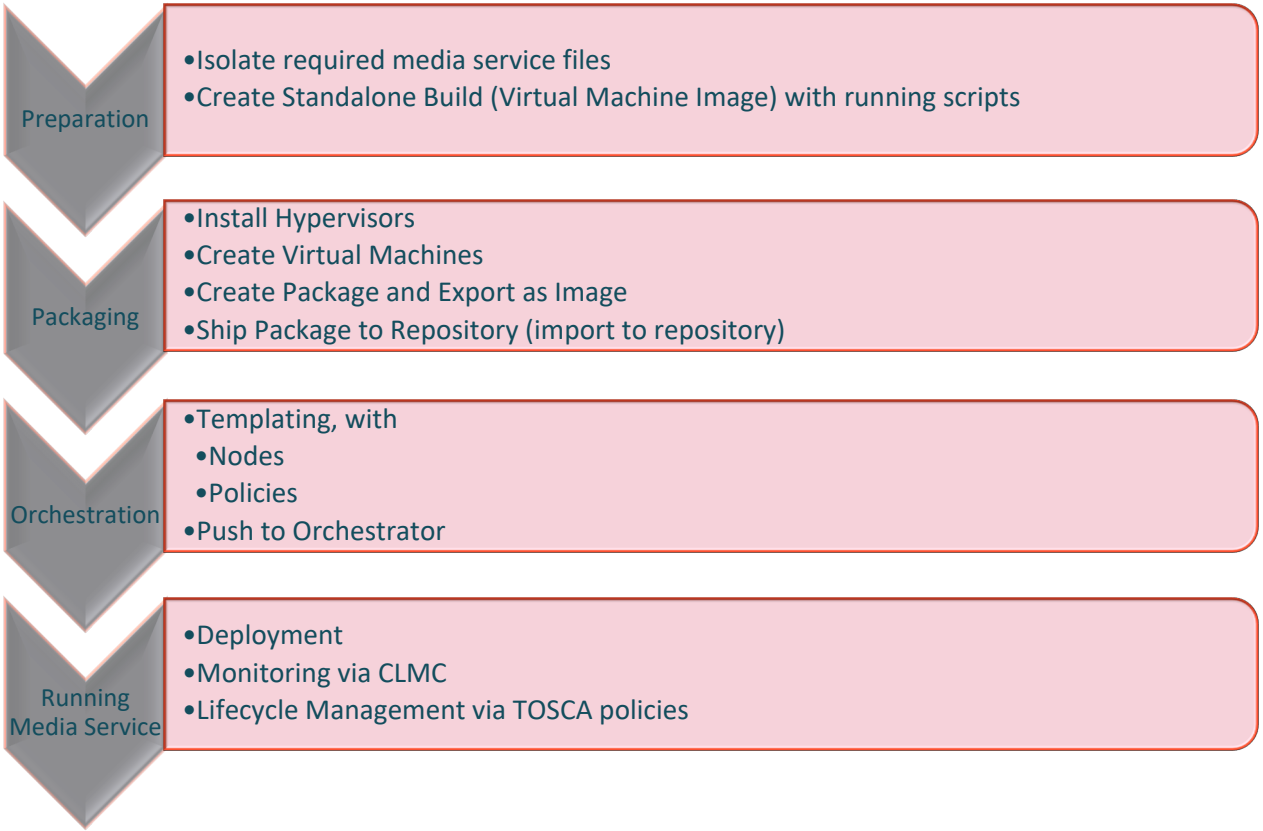
Validating the TOSCA templates

validator: ./test.sh -d ../definitions/flame_definitions.yml -t ../examples/high_latency.yml

validator_1 | Try to parse.../templates/template.yml
validator_1 |
validator_1 | version: tosca_simple_profile_for_nfv_1_0_0
validator_1 |
validator_1 | description: Template for deploying a service based on policies.
validator_1 | When the delay of the deployed services is too high,
validator_1 | another service endpoint will automatically be set
validator_1 | to the CONNECTED state.
validator_1 | However, only if the payment of the service has been received.
validator_1 |
validator_1 | nodetemplates:
validator_1 | eu_flame_app_db
validator_1 | eu_flame_app_frontend
validator_1 | policies:
validator_1 | init:
validator_1 | trigger(s):
validator_1 | name: inital_trigger
validator_1 | high_latency_check:
validator_1 | trigger(s):
validator_1 | name: check_trigger
validator_1 | cooldown_policy:
validator_1 | trigger(s):
validator_1 | name: time_trigger
validator_1 | service_paid:
validator_1 | trigger(s):
validator_1 | name: not_paid_trigger
validator_validator_1 exited with code 0

Hands-On

Workflow



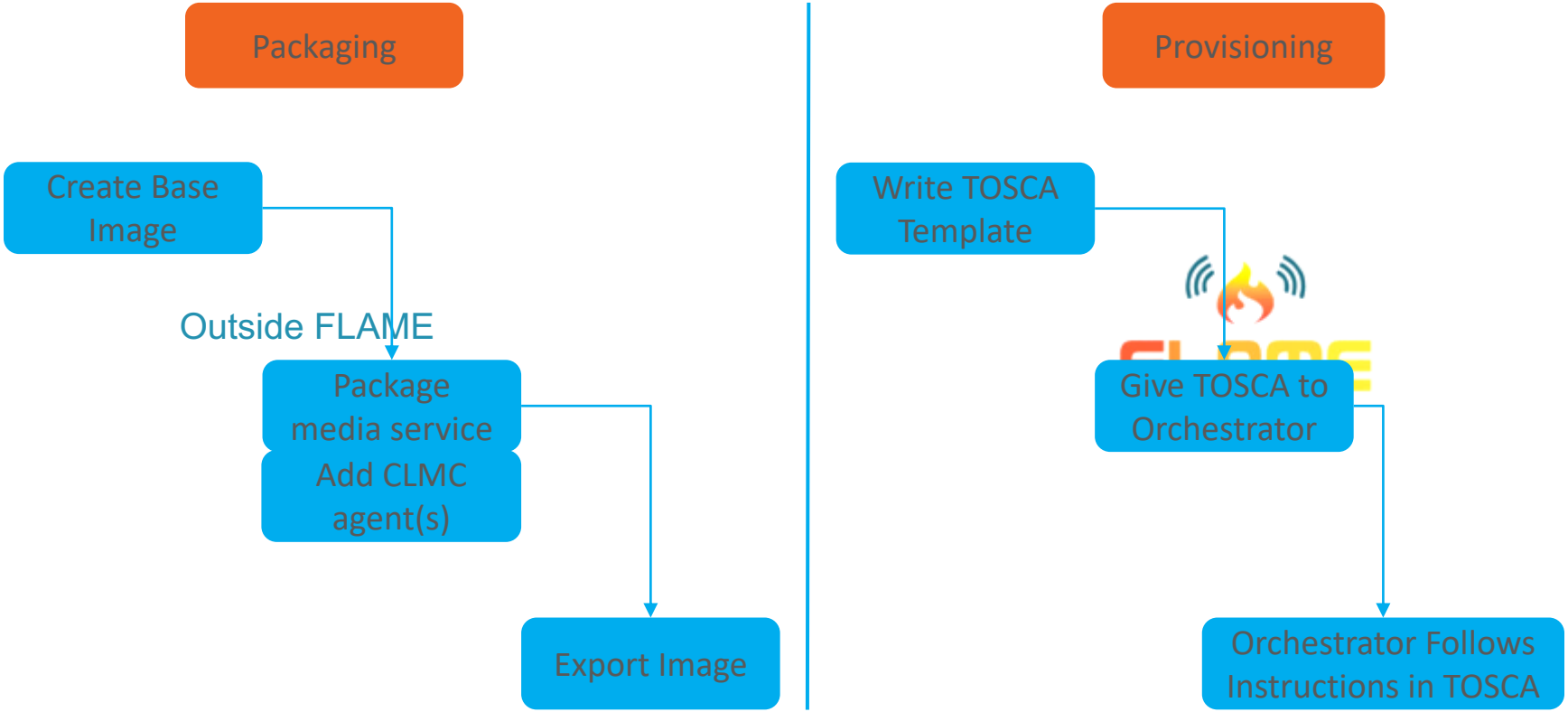
Preparation

- Isolate required media service files
- Create Standalone Build (Virtual Machine Image) with running scripts

Packaging

- Install Hypervisors
- Create Virtual Machines
- Create Package and Export as Image
- Ship Package to Repository (import to repository)

MS Packaging and Provisioning Workflow



Properties of Image

- Disk
  - Format: qcow2
  - Bus: virtio
- Network
  - One interface
  - On slot 0x05
  - DHCPv4 daemon enabled in media service image
    - Gateway and DNS server configured by DHCP server of FLAME platform

Create Image

- Base image created for a particular Linux variant
  - If base image already exists, script clones it
  - If base image does not exist, script creates it first
- The script finishes when the desired image is created using the FQDN of the media service as its name, e.g. video.msws.flame.eu
  - All dots in instance name are replaced by hyphens
- All images are stored to /var/local/kvm/img
- All KVM definitions are stored to /var/local/kvm/xml

Create Image: Hands On

```
$ cd flame-wp4/packaging
$ ./create-vm.sh <LINUX_VARIANT> <FQDN>
```

- Hostname: test-flame-eu
- User: flame
- Enable password-based SSH access

```
$ ssh flame@<VM_IP>

$ su
$ sed –i “g/prohibit-password/yes/” /etc/ssh/sshd_config
$ systemctl restart ssh
```

Package Service

- Use SSH to configure instance
- Use SCP to push files/binaries into instance

Export Instance

- Call ./export-vm.sh <FQDN>
- Compressed TAR ball created with image in /var/local/kvm/img

Make Image available

- Orchestrator requires image to be available via web server on Port 80
- TOSCA template has a URL where the image can be retrieved

Orchestration

- Templating, with
  - Nodes
  - Policies
- Push to Orchestrator
  - An Interface between the Media Service Provider and the Orchestrator is currently under evaluation

Templating – get the file: https://goo.gl/jsz8Y7

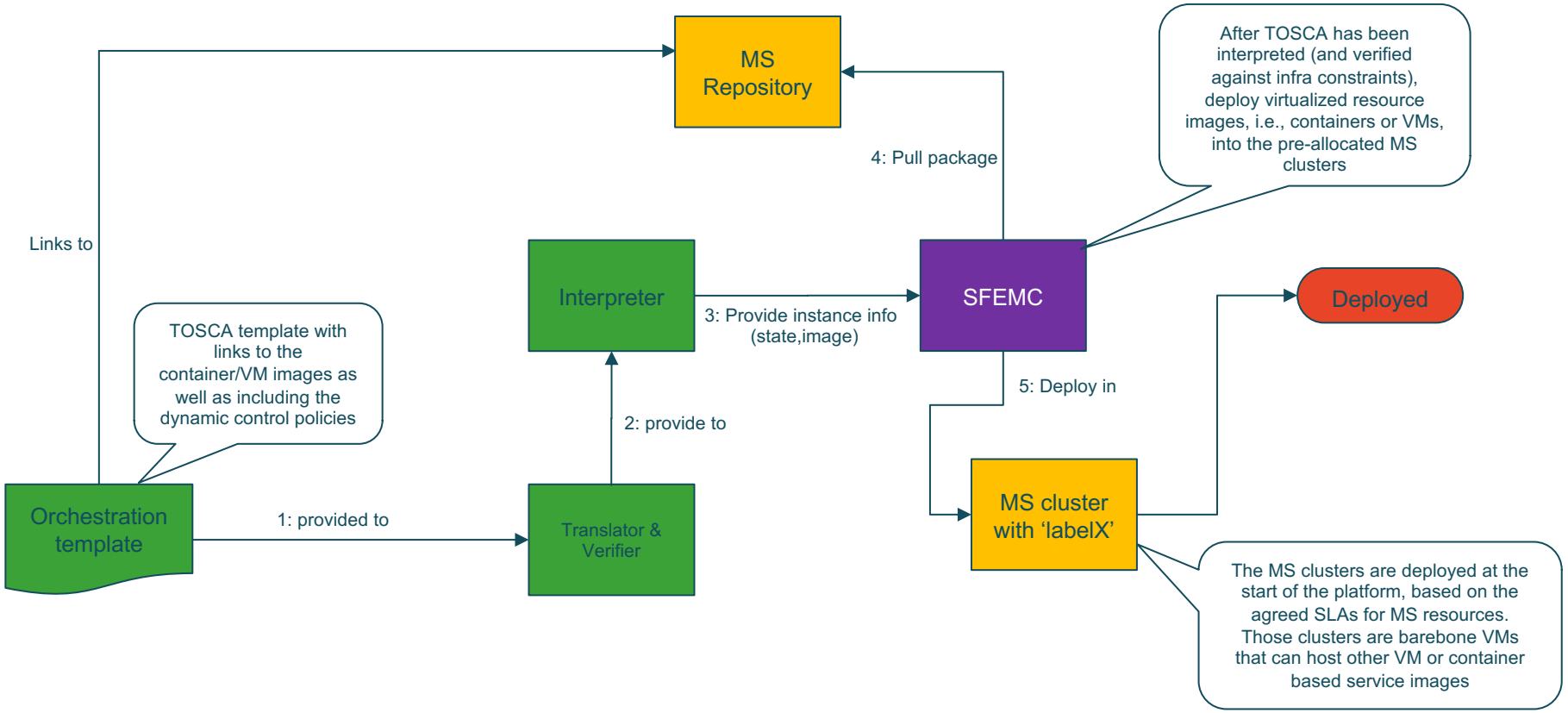
```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: |
  Template for deploying a service based on policies.
  The scenario includes the TCP-Port information for the nodes.
metadata:
  template_name: "Flame Use Case: Allow certain FQDNs to be accessible via Internet"
  template_author: Kay Haensge <kay.hansge@interdigital.com>, Sebastian Robitzsch
<sebastian.robitzsch@interdigital.com>
  template_version: 0.1.4
  servicefunctionchain: allow_fqdns_sfc
imports:
  - flame_definitions.yml
topology_template:
  node_templates:
    database:
      type: eu.ict-flame.nodes.ServiceFunction
      capabilities:
        endpoint:
          type: tosca.capabilities.Endpoint
          properties:
            ports: # uses a map of PorSpecs
              mysql_port:
                protocol: tcp
                target: 3306
        host:
          properties:
            num_cpus: 2
            mem_size: 4096 MB
            disk_size: 10 GB
      properties:
        hypervisor: kvm
        image_url: http://cdimage.debian.org/cdimage/openstack/current-9/debian-9-openstack-
amd64.qcow2
        fqdn:
          - db.app.ict-flame.eu
    frontend:
      type: eu.ict-flame.nodes.ServiceFunction
      capabilities:
        endpoint:
          type: tosca.capabilities.Endpoint
          properties:
            ports: # uses a map of PorSpecs
              http:
                protocol: tcp
                target: 80
              https:
                protocol: tcp
                target: 443
        host:
          properties:
            num_cpus: 2
            disk_size: 10 GB
            mem_size: 4096 MB
      properties:
        hypervisor: kvm
        image_url: http://cdimage.debian.org/cdimage/openstack/current-9/debian-9-openstack-
amd64.qcow2
        fqdn:
          - frontend.app.ict-flame.eu
          - www.app.ict-flame.eu

## Policies
policies:
  - init:
      type: eu.ict-flame.policies.InitialPolicy
      description: Start the nodes initially
      properties:
        parent: service_paid
      triggers:
        inital_trigger:
          condition:
            constraint: initialise
          action:
            frontend:
              -
                fqdn: frontend.app.ict-flame.eu
                lifecycle_actions:
                  London: eu.ict-flame.sfe.state.lifecycle.connected
            database:
              -
                fqdn: db.app.ict-flame.eu
                lifecycle_actions:
                  London: eu.ict-flame.sfe.state.lifecycle.connected
  - allowed_fqdns:
      type: eu.ict-flame.policies.InternetAccessPolicy
      properties:
        method: permit
        fqdn:
          - www.google.com
          - www.github.com
          - gitlab.it-innovation.soton.ac.uk
```

Running Media Service

- Deployment
  - The Orchestrator deploys available images in the given locations
  - For nodes with a “connected” target, the FQDNs are registered at the Service Router
- Monitoring via CLMC
  - Monitoring information of the running instances can be pushed to CLMC
- Lifecycle Management via TOSCA policies
  - Upon thresholds of monitored data, lifecycle management of deployed nodes will be performed

Recap: Media Service Orchestration / Deployment



Conclusions

- Virtualization of compute & connectivity allows for (secure) deployment of service instances:
  - Move away from insecure content replication to secure service replication.
- Fast replication requires fast indirection:
  - Move away from inefficient DNS-based service routing to efficient Layer 2 based service routing.
- Experimentation API at two levels of complexity:
  - Hands-on showed the ‘exact’ TOSCA template approach.

FLAME Online

DISCOVER OUR PRESENCE ONLINE AND GET INVOLVED!

FOLLOW US ON TWITTER!
https://twitter.com/ICT_FLAME

OUR WEBSITE!
www.ict-flame.eu

FOLLOW US ON LINKEDIN!
https://www.linkedin.com/groups/8579978

CONTACT US!
info@ict-flame.eu

SUBSCRIBE OUR NEWSLETTER!
https://www.ict-flame.eu/newsletter/

FLAME

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

THANKS FOR YOUR ATTENTION!

ICT-FLAME.EU

@ICT_FLAME

Sources

- http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/
- https://gitlab.it-innovation.soton.ac.uk/FLAME/flame-tosca
  - /definitions
  - /documentation
  - /examples
  - /validator