SUPERFLUIDITY

A SUPER-FLUID, CLOUD-NATIVE, CONVERGED EDGE SYSTEM

Research and Innovation Action GA 671566

DELIVERABLE I7.1:

PRELIMINARY DEFINITION FOR THE USE CASE SCENARIOS FOR DEPLOYMENT AND VALIDATION

Deliverable Type:	Report
Dissemination Level:	PU
Contractual Date of Delivery to the EU:	31.12.2016
Actual Date of Delivery to the EU:	05.01.2017
Workpackage Contributing to the Deliverable:	WP7

Editor(s):	Philip Eardley (BT)/Francisco Fontes (ALB)
Author(s):	Stefano Salsano (CNIT), Claudio Pisa (CNIT), Lionel Natarianni, Bessem Sayadi (Nokia FR), Erez Biton (Nokia IL), Philip Eardley (BT), George Tsolis (Citrix), Michael McGrath (Intel), John Thomson (OnApp), Francisco Fontes, Carlos Parada, Isabel Borges (ALB), Rufael Mekuria (USTR), Radu Stoenescu (UPB), Costin Raiciu (UPB)
Internal Reviewer(s)	

Abstract: This internal deliverable provides a preliminary description of the validation use case scenarios along with their technical and administrative challenges and the performance and functional assessment key indicators.

Keyword List:

VERSION CONTROL TABLE			
VERSION N.	PURPOSE/CHANGES	AUTHOR	DATE
0.1	First integrated version, for partners revision	Francisco Fontes/Philip Eardley	19/Dec/2016
0.1	First complete version, for partners revision	Francisco Fontes/Philip Eardley	22/Dec/2016
1.0	Release to EC		5 Jan 2017

INDEX

List of Figures

List of Tables

Glossary

(TO BE FILLED OUT IN THE FINAL VERSION OF THE DELIVERABLE)

SUPERFLUIDITY DICTIONARY	
TERM	DEFINITION
C-RAN	
MEC	
RFB	

Table 1: Superfluidity Dictionary.

1 Introduction

The Superfluidity project will establish an integrated demonstrator for the deployment and validation of the project's developments, during the last part the project. This is intended to showcase, integrate and evaluate project results, with a special focus on the Reusable Function Blocks (RFB) that we develop. This internal report describes the plans for the demo. Its purposes are:

- To help guide the development and integration
- To plan for the evaluation and validation of Superfluidity technology
- To help improve the demo, through feedback (on this interim version of the deliverable) from project members, colleagues in the partner organisations, the project officer and so on. This may lead to, for example, identification of additional aspects that it would be good to add or valuate.

The report builds on the earlier work of the project, in particular:

- The use-cases of D2.1
- The KPIs identified in D2.1 and the related, higher-level 5G-PPP targets
- The four existing demos, which are stand-alone or show a lower-level of integration. Since these were not yet described in a deliverable, the Appendix briefly summarises them.

D2.1 described a wide range of use cases where we expect that Superfluidity will be relevant and the associated technical and business requirements. The use cases fall into several overlapping areas:

- The wireless access part of the network
- Monitoring
- Services delivered from the edge
- Security

We want our demonstration to integrate aspects of all these areas, and so to be broader than any one of the D2.1 use cases. Since the primary purpose is to bring together most of our technology developments, at this stage we have not tried to develop a novel, exciting user-centric story. Feedback is welcome on whether it is important to fix this in the next iteration of the document.

D2.1 also provided some initial identification of relevant metrics or Key Performance Indicators, such as:

- Scalability – the ability to scale up /out (for example)
- Orchestration – the speed of deployment and service migration supported (for example)
- Analytics – the ability to collect and analyse metrics
- Service agility – the time to develop, deploy or tailor a service
- Security – the ability to detect and isolate anomalous events

Again, the demonstration will integrate aspects of these KPIs.

The 5G-PPP community has proposed the following parameters as indicative new network characteristics to be achieved at an operational level:

- 1000 times higher mobile data volume per geographical area
- 10 to 100 times more connected devices
- 10 times to 100 times higher typical user data rate
- 10 times lower energy consumption
- End-to-End latency of < 1ms
- Ubiquitous 5G access including in low-density areas

In addition, other 5G-PPP goals include:

- Retaining at least 35% of the global market share in Europe for equipment
- European industry driving the development of 5G standards and having at least 20% of the 5G Standards Essential Patents.
- Supporting lower cost access to a wider spectrum of applications and services
- Reduction of the network management OpEx by at least 20% compared to today
- New lightweight but robust security and authentication metrics.

Superfluidity's technology advances have been documented in the deliverables from work packages 3, 4, 5 and 6 and are summarised schematically in Figure 1:

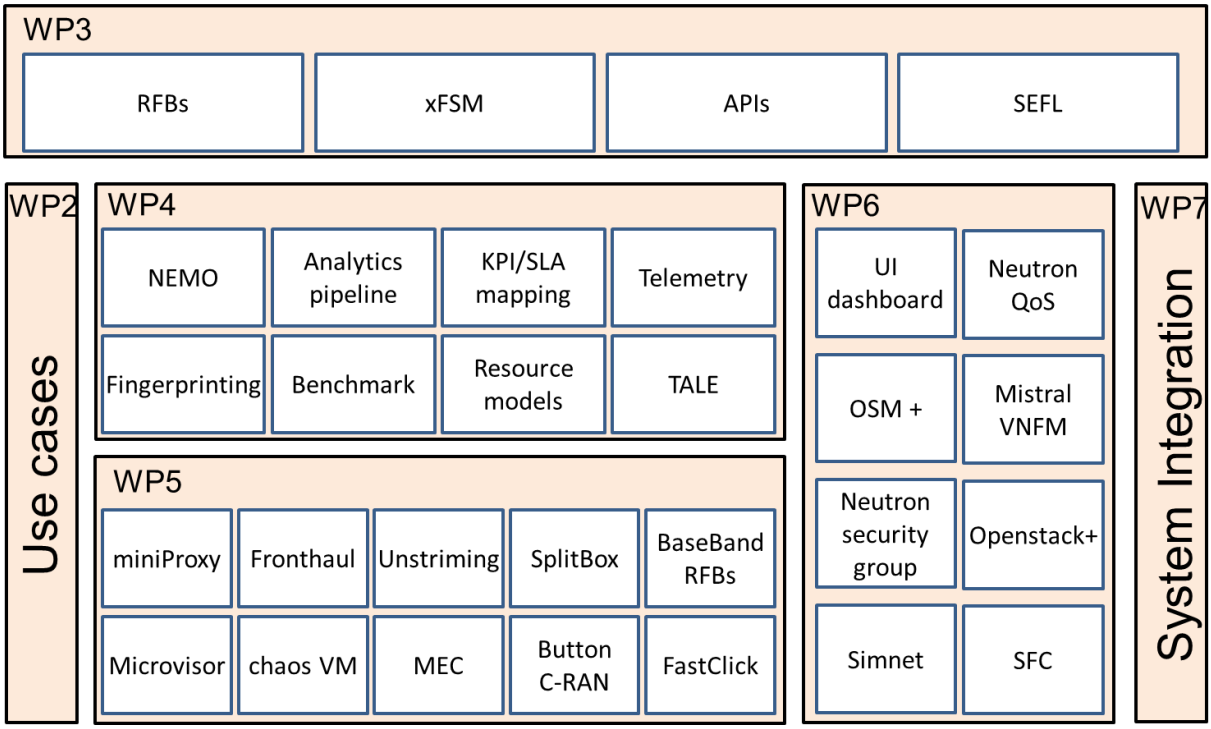


Figure 1: Summary of Superfluidity's technologies

As is apparent from the description below, most of these technologies are included in the demo.

The integrated demonstration consists in the running in sequence of several scenes. From an end user's perspective, the story is someone watching a video. Although this is simple, video is the dominant application on the Internet today and it is expected to remain critical, and our partner Unified Streaming can readily provide a streaming video application. However, with a little imagination it can incorporate a wider range of more novel user-centric stories, for example an augmented reality, multi-user game in which a headset displays live video from other participants.

Each scene shift is triggered by some change in circumstances (for example: more users arriving, or a DDoS attack, or some noisy neighbour effect on the platform) which is noticed by a significant change in the monitored KPIs. Pre-analysis has determined a critical value for KPIs and how to solve the issue – this is typically by the operator altering how the service is delivered (for example: scaling the MEC, or adding in some filtering, or traffic prioritisation on the platform). In principle, the analysis can be done in real time, but in practice most issues can be analysed in advance. The process is sketched in Figure 2 below.

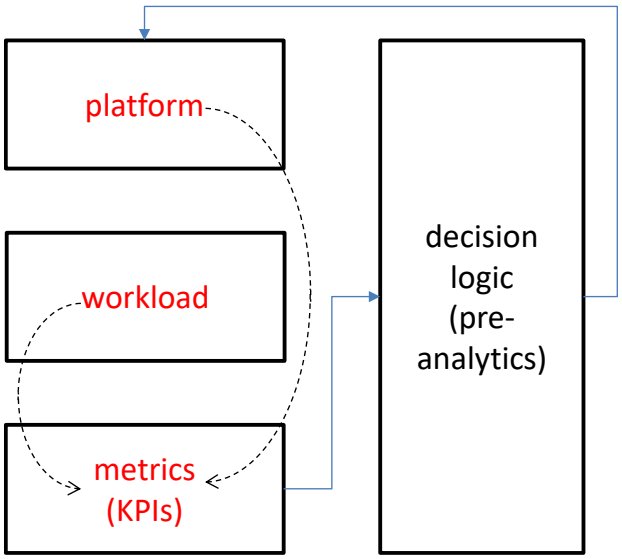


Figure 2: Changes in the monitored metrics trigger a change in the platform to address the issue

The different scenes will run in sequence, sharing a common infrastructure. This way, the final conditions left by one scene will serve as input or initial conditions for the following scene. This strategy will contribute to the achievement of an integrated system, showcasing Superfluidity technology.

2 Validation use case scenario

The demonstration use case consists of several scenes.

2.1 Scene 1: Initial access network setup (CRAN and MEC)

2.1.1 Scene description

Provide instantaneous deployment and instantiation of a full CRAN and Core network, leveraging RFBs dynamic composition to have a minimum downtime. Adapt resource usage to user needs in near real-time.

2.1.2 Technical trigger to start this scene

The operator uses the RDCL 3D tool (described below) to build a graph representing the composition of RFBs. From this graph, the RDCL 3D tool produces an artefact file, the RFBs.yml file, used for the deployment of the RFB composition.

In this scene, the RFB controller, from its REST API, loads the RFBs.yml that initiate the complete deployment (resources + RFBs).

In this scene, we are agnostic about the underlying infrastructure and resources. The RFB controller receives a set of RFB to be created. Resources and RFBs are decoupled using nodes as logical representation of resources and deployment requirements. The mapper registers available resources in a dedicated database containing all available VMs and hardware for deployment.

After the full deployment of RFBs on nodes, the CRAN and its related components are deployed on the Edge and the Core nodes; the LTE network is operational. At this step, the CRAN network can accept User Equipment (UE) connections.

2.1.3 Technical description and involvement of Superfluidity technology

RDCL 3D (*RFB Description and Composition Languages for Design, Deploy and Direct*) is a web application that supports the editing and deploying of NFV based services on top of existing orchestrators. It provides a web GUI that can be used to visually represent and design descriptor files for NFV services and components (e.g. VNFs). RDCL 3D is able to interact with the concrete descriptor syntax used by existing MANO orchestrators, as well as to represent services and components at a higher level of abstraction, following the RFB based architecture proposed by Superfluidity.

All deployments are performed by the *RFB Execution Engine* (REE), which contains an internal high-level orchestrator used for resource and RFB deployments. The role of the REE is to decouple resource deployments from service deployments and implement the RFB graph model execution.

Function composition needs to be deployed on nodes. The objective of the first pass is to build nodes from available resources, onto which RFBs are deployed in the second pass. To address the resource element, the REE provides drivers on its southbound to control and deploy resource items (e.g. OpenStack Heat). The available resources items are stored in the resource repository, which is also managed by the REE orchestrator.

The deployment of the whole platform is performed in two main steps. On each step, the RFB controller performs a pass to read the RFB artefact file.

a) On the first pass (Figure 3), the RFB controller receives a set RFB to be created. Related resources are given by the resource mapper (VM and bare metal servers).

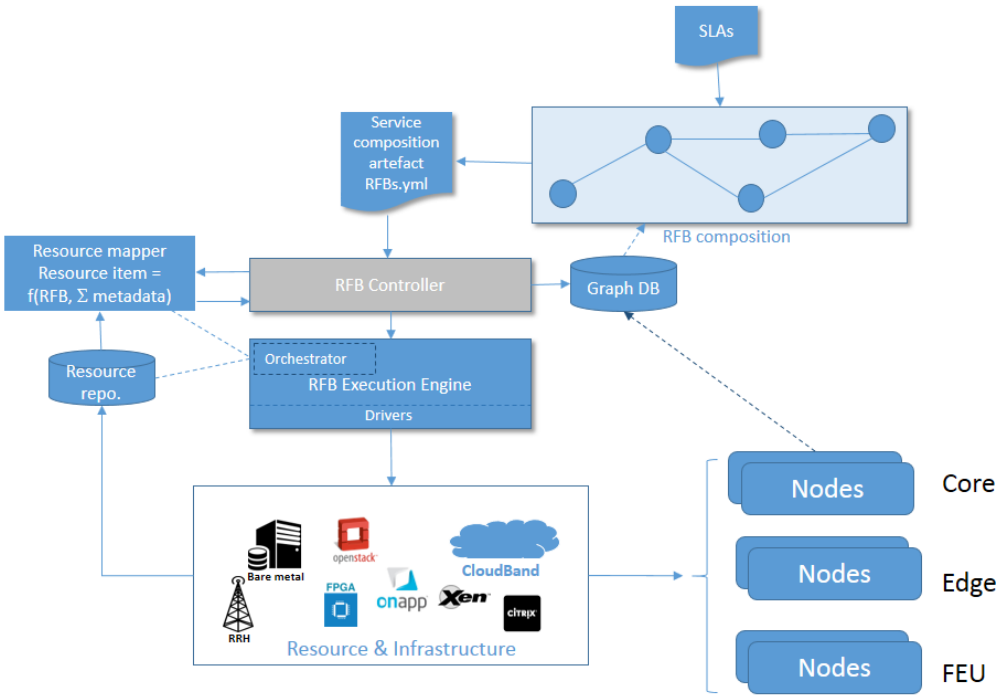


Figure 3 –Resources allocation for RFB

- The RFB controller is invoked with RFBs.yml which contains the full composition described as an RFB service graph
- The REE, from its orchestrator, initialises related resource deployment by interpreting each RFB and translating each of them into a resource element (e.g. VM, hardware server) from the resource mapper
- Related resource elements are initialised; from this point, each resource element is represented as node and stored in the graph DB

b) On the second pass (Figure 4) the CRAN RFBs are deployed: RRH, BBU, EPC, SDN fronthaul controller and its northbound application to set up the link between the RRH and the BBU.

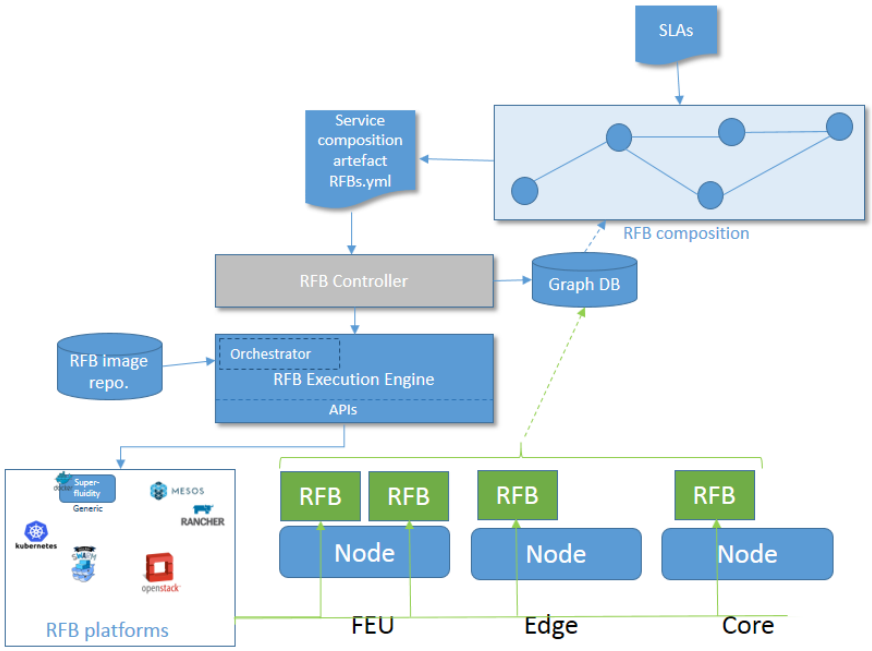


Figure 4 – RFBs deployment

- The second pass starts from deployed resources and nodes; the REE reparses the RFBs.yml and each RFB and its related metadata are extracted and interpreted
- According to requirements and affinities (describes in RFB metadata), the REE orchestrator deploys RFBs on corresponding nodes using the RFB image repository

2.1.4 Implementation challenges for the demo

a) Provide a composition model for CRAN deployment as RFB.

A service composition format has been defined according to service requirements and customer SLAs for the CRAN and fronthaul deployment.

We provide a model for service composition as an artefact file (RFBs.yml) that contains the RFB chaining graph and related metadata used for the deployment.

The composition structure built for the demo follows SLA original requirements. We enrich the artefact file with refinements to describe specific capabilities (redundancy, compute & network capacity, …) as a metadata sub-section for each RFB.

b) Decoupling RFB from nodes for dynamic deployments.

Successive service compositions, deployments and 'undeployments' can be performed using the same resource deployment. It means that resource (nodes) do not have to be redeployed between each RFB deployments. The same RFBs.yml artefact file is used for the two passes (resource and service deployment). It is also use to undeploy RFBs and nodes.

c) Provide a high level orchestrator, enable multi-level orchestration for CRAN

All deployments are performed by the RFB Execution Engine (REE), which contain an internal high-level orchestrator used for resource and service orchestration.

The role of the REE is to decouple resource from service deployments and orchestration, and to implement the RFB graph model execution.

A light and generic RFB platform is used for service deployment. The REE allows orchestration from state of the art RFB platforms for orchestration such as Google Kubernetes, Docker Swarm and Apache Mesos.

All deployed RFBs are registered and managed as graph, using a graph database and related query language (e.g. Neo4j, Cypher).

2.1.5 Evaluation plan and validation metrics

The evaluation plan will focused mainly in quantifying the improvement in the **instantiation time** for the RFBs and nodes.

We also intent to evaluate **orchestrator efficiency** for resource-RFB optimization, **cost** and **performance optimization** (locating RFBs in the specific geographic area).

Some other fronthaul and SDN capabilities (**latency**, **throughput**) measurement will be performed.

2.1.6 Relevant 5G-PPP metrics

Metrics will be measured in milliseconds for instantiation time and other CPU usage, memory allocation and network performance and load for different deployment patterns.

2.2 Scene 2: Initial metrics start and analytics

2.2.1 Scene description

Scene 2 describes the pre on-boarding characterisation of workloads before deployment onto the Superfluidity system using the automated application of telemetry and analytics. The scene also outlines how the Superfluidity system maintains SLA compliance for services through telemetry and modelling to support system actuation events such as service scaling.

2.2.2 Technical trigger to start this scene

When on-boarding a workload onto the Superfluidity system, characterisation is carried out as a pre-deployment step to develop a clear understanding of the workload's behaviour under the operational conditions required by the system demonstration as outlined in scene 1 and scene 3. The characterisation process allows the identification of the most influential metrics, which must be monitored and collected during the deployment. Secondly, the process supports the development of models/heuristics, which are used by the Superfluidity service orchestration layer to manage the performance of the workload. Specifically the Orchestrator must maintain the SLA defined for the workload for the demonstration. The SLA contains the specific KPIs that must be maintained by the workload in order to ensure the required end user QoE. Telemetry also provides on-going monitoring of a service during operational deployment. For the purposes of the Superfluidity demonstration, the telemetry will be used to monitoring the performance of the workload under different scenarios e.g. deployment with noisy neighbours. Telemetry will also enable the monitoring of the Superfluidity system elements such as fronthaul and SDN capabilities as outlined in scene 1, which require the measurement of throughput and latency for example.

The rest of the process is described in the next sub-sections.

2.2.3 Technical description and involvement of Superfluidity technology

Off-line Phase (Characterisation and KPI Mapping)

The characterisation process starts by deploying the workload in an offline phase on the Superfluidity system using an automated characterisation framework developed in WP4. This framework has responsibility for deploying the workload, executing predefined test cases e.g. measurement of throughput etc., application of workload traffic (e.g. HAMMER), collection of telemetry data, processing and persistence of the data and data modelling. The snap telemetry framework provides telemetry data collection and is pre deployed on the Superfluidity system with a set of plugins to collect metrics from layers within a deployment stack i.e. from the infrastructure layer up to the service layer. The specific plugins deployed depend on whether the workload is being deployed as a virtual machine, container or Unikernel.

The data is initially used to characterise the performance of the workload using the Throughput/Anomalies/Latency/Entropy approach developed in WP4 to identify opportunities for optimisations e.g. improving throughput performance. Secondly, the characterisation process is used to identify metrics, which can be used for scaling triggers. This data can then be used to build a scaling model. The model comprises: the service level objectives (SLOs) which encompass the SLA that must be maintained; the KPIs for the SLOs; the platform metrics mapping to KPIs'; the actuation thresholds for the metrics and the actuation actions.

The second step of the characterisation process is identify the metrics that most significantly influence the KPIs of the workload to be deployed. The metrics data collected during the execution of the test cases by the framework are ingested into an analytics pipeline. The analytics pipeline implements different techniques depending on the type of workload and relationships to be elucidated. For the Superfluidity demonstrator an ensemble approach (a mixture of statistical/clustering and machine learning approaches) is used to identify which platform metrics have the most significant influence on a workload's KPIs. This allows the number of metrics to be collected and monitored to be significantly reduced in the Superfluidity demonstrator i.e. reducing hundreds of metrics to tens of metrics.

On-line (Monitoring and Scaling)

The snap framework manifest is configured only to collect the metrics, which are identified as being important in the offline characterisation phase. When the workload is deployed the snap framework uses plugins to collect relevant metrics from the infrastructure layer, virtualisation layer (VM, Container) and service layer i.e. workload. Metrics are persisted to a database for storage. An API is used to expose the metrics, which have been identified for the purposes of monitoring and maintaining the KPIs associated with a workload. These metrics together with the scaling model are used by the service orchestrator to maintain the SLA for the workload and to carry out actuation actions such as scaling in order to maintain the SLA. The data persisted in the database is also visualised using a Grafana based user interface. By default, metrics relating to scaling and QoE are displayed. The key steps to in the offline and on-line phases are shown in Figure 5.

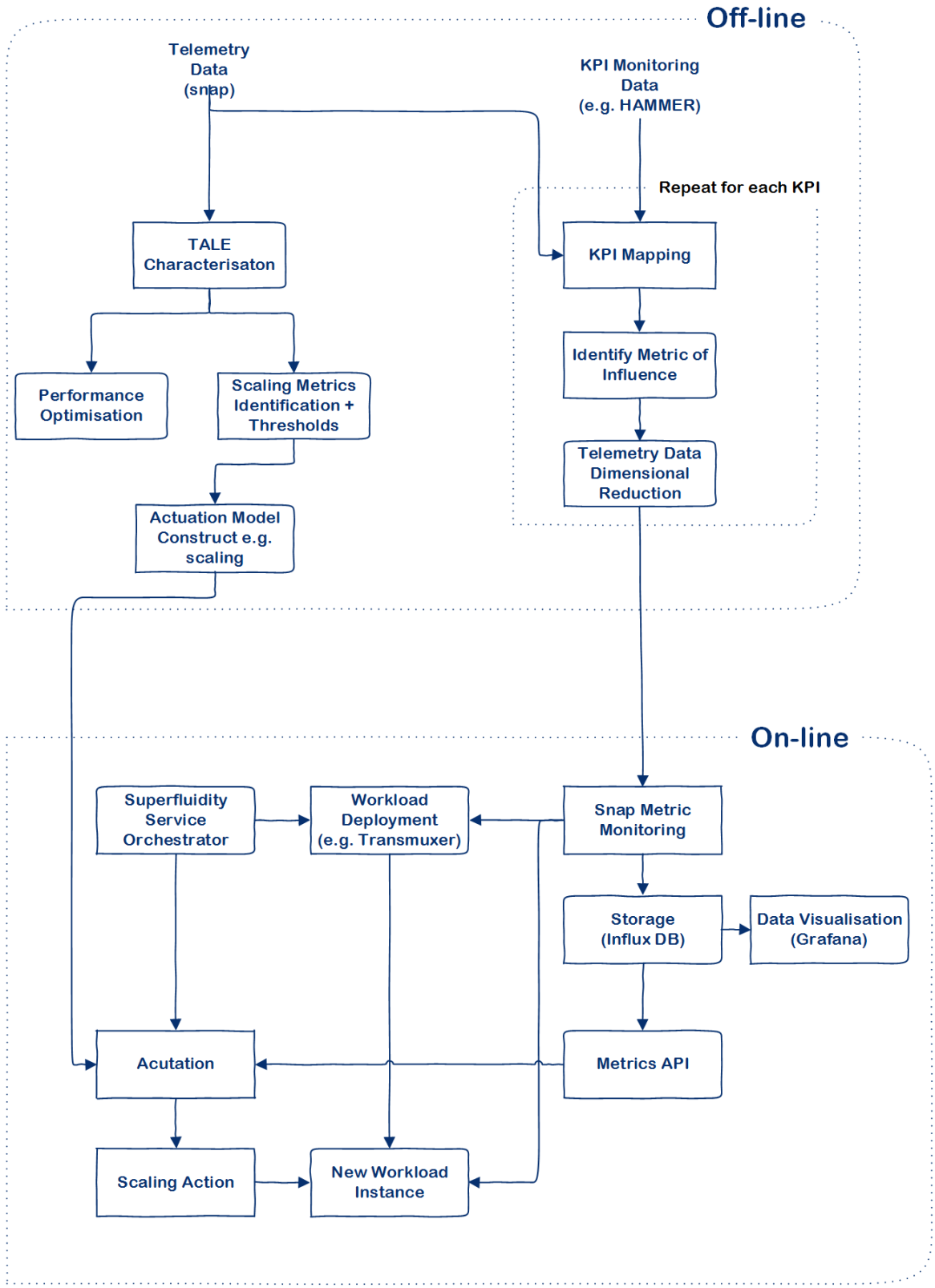


Figure 5 - Superfluidity Workload Metrics Identification, Monitoring and Actuation

2.2.4 Implementation challenges for the demo

- The potential need to characterise workloads using different virtualisation approaches (VMs and Containers) and bare metal (Unikernels)
- Telemetry framework may need to support VMs, Containers and Unikernels (requirement for RFB platform – see scene 1). Unikernels are particularly challenging as the minimal kernel footprint means that most of the standard Linux counters are not available and therefore Unikernels appear as essentially black boxes to a telemetry agents.
- Scaling model implementation – where to locate scaling logic i.e. extension to telemetry or implemented as part of the service orchestrator. The service orchestrator is the most appropriate. However, this would require customisation of the pre-existing solution, which will be adopted by Superfluidity.

2.2.5 Evaluation plan and validation metrics

Characterisation of the Unified Origin video transmuxer workload was previously conducted in WP4 (see deliverable I4.1). The following metrics were found to most significantly influence the performance of the Unified Origin workload:

- The **user request arrival rate** (number of TCP connections per second) has a significant effect on the stability of network throughput performance. The service exhibits throughput stability when the user arrival is less than or equal to the number of users the system can support per second. The specific number will be identified for the Superfluidity demonstrator.
- The maximum **network throughput** directly affects the number of users that can be supported. The specific threshold will be identified for the Superfluidity demonstrator.

The key conclusions from the characterisation activities was that service throughput should be consider in the application scaling strategy. The absolute number of users is not a useful metric for a scaling strategy, as the total network throughput generated by any given number of users will vary depending on the resolution of the video requested and streamed.

Evaluation of KPIs for the Unified Origin transmuxer was previously carried out in WP4. The HAMMER workload generator provides the ability to measure KPIs of relevance to performance aspects of the workload, which have a direct effect on the QoE for users, namely throughput and latency.

Analysis of platforms metrics and the KPIs as measured by HAMMER identified the following correlations:

Latency performance mapped to memory subsystem related parameters while the throughput mapped to network subsystem parameters. The top 10 metrics for each KPI as shown in Table 2 in descending order of relevance.

Latency	Throughput
proc_stat_meminfo_active_file	net_interfaces_network_utilization
proc_stat_meminfo_sunreclaim	net_interfaces_receive_bytes
proc_stat_meminfo_cached	net_interfaces_transmit_packets
proc_stat_meminfo_active	net_receive_bytes
proc_stat_meminfo_active_anon	proc_stat_meminfo_slab
proc_stat_meminfo_anonpages	proc_stat_meminfo_sunreclaim
proc_stat_meminfo_memavailable	net_interfaces_receive_packets
proc_stat_meminfo_anonhugepages	net_tx_mb
proc_stat_meminfo_sreclaimable	net_transmit_bytes
proc_stat_meminfo_free	net_tx

Table 2: Most influential workload metrics on latency and throughput KPIs

Analysis of the data from optimized configuration of Unified Origin provided system level correlations for KPIs is shown in Table 3. Both KPIs correlate primarily with scheduler or CPU related metrics.

Latency	Throughput
proc_schedstat_running	proc_schedstat_wake_up_local
system_bmsr	system_bmsr
net_interfaces_receive_bytes	net_interfaces_receive_bytes
net_interfaces_transmit_bytes	proc_schedstat_schedule
proc_schedstat_tasks	proc_schedstat_tasks
proc_schedstat_waiting	proc_schedstat_waiting
proc_stat_cpu_steal	proc_stat_cpu_steal
proc_stat_cpu_all	proc_stat_cpu_idle
proc_stat_cpu_sys	proc_stat_cpu_sys
proc_schedstat_schedule_idle	proc_schedstat_schedule_idle

Table 3: Most influential system level metrics on latency and throughput KPIs

Evaluations

The evaluation of the telemetry and analytics in the Superfluidity system demonstrator will comprise the following elements:

Telemetry

- Confirm the quality and accuracy of the telemetry data stored in the InfluxDB
- Confirm the temporal resolution of the data as configured i.e. if a sampling rate of 10Hz is used confirm there are 10 samples per second persisted to the database
- Confirm metrics from VMs and Containers are successfully collected and persisted to the database.

Analytics

- Quantify the accuracy of the models used to identify the correlations between the platform metrics and KPIs (applicable to scene 3)

Scaling Model

- Quantify that the scaling model maintains the service level KPIs while maintaining resource utilisation.

2.2.6 Relevant 5G-PPP metrics

The performance of the telemetry and associated analytics do not directly influence the 5G PPP metrics but rather have indirectly influence on them within the Superfluidity system. The telemetry system and the performance of the real-time analytics need to be scalable in order to support the 10 to 100 increase in the 5G network footprint i.e. 10 to 100 times more connected devices. The increase in the number of connected devices and the number of network elements required to support this expansion will requirement a highly scalable and performant telemetry system, which can provide local data processing and filtering before exposure to a higher-level orchestration process. Telemetry also provides a key tool in the process of service optimisation in order to achieve a 10 to 100-fold increase in user data rates. This will require service characterisation and optimisation prior to deployment followed by continuous monitoring and service rebalancing order to sustain the increase in user data rates. Similarly, telemetry and analytics will be a key tool in helping to achieve end-to-end latency of <1ms through the identification of bottlenecks in the service stacks i.e. both hardware and software.

2.3 Scene 3: Central deployment of initial streaming server

2.3.1 Scene description

The video streaming server is deployed at some central point in the network.

2.3.2 Technical trigger to start this scene

In traditional video streaming the server runs at the core on a bare metal server. This server responds to user requests, possibly arriving through a networked cache, serving different devices with different protocols. This approach creates duplicate media data throughout the network (core, edge) to support the different devices as different protocols use different formats (e.g. MPEG-DASH, Apple HLS). In addition, the server could be overloaded by too many user requests, i.e. there is limited scalability. Furthermore the distance from core to device introduces latency thanks to the overhead of the TCP protocol used for video streaming. To combat this, using a transmuxing video server is beneficial.

2.3.3 Technical description and involvement of Superfluidity technology

Superfluidity technology is mainly used in the following scenes, which involve edge computing and virtualisation.

2.3.4 Implementation challenges for the demo

In this demo the transmuxing and caching functionalities are split in different blocks. The block based decomposition enables deployment in the mobile edge based on the scene 4 described next.

2.3.5 Evaluation plan and validation metrics

The system is evaluated using load testing using Hammer software.

2.3.6 Relevant 5G-PPP metrics

A benefit of using a transmuxing server is that storage is reduced and video streaming made possible in the 5G architecture.

2.4 Scene 4: Video streaming uses MEC

2.4.1 Scene description

These will demonstrate the switching of a service being provided at the core to start being provided at the edge, using MEC (Mobile-Edge Computing or Multi-access-Edge Computing).

2.4.2 Technical trigger to start this scene

In the previous scene (scene 3), Alice starts the application to watch a very popular video; this service was provided from a central streaming server deployed in the Core. In this scene, a new nearby user, Bob, starts requesting the same video, using a different streaming protocol (e.g. AppleHLS).

In this scene, the operator's policy for this streaming service states that:

- "when 2 or more users, located at the same edge, are watching the same video using a different streaming protocol or rate, it is worthy to deploy an LTM ME App at this edge". (Late TransMuxing Mobile Edge)

This minimizes the amount of traffic traversing the core, reduces backhaul cache storage, increases the bandwidth efficiency and ensures a good quality of service for the users.

2.4.3 Technical description and involvement of Superfluidity technology

The detection of two or more users watching the same video can be performed in three different ways:

1. The TOF (*Traffic Offloading Function*) component at the edge acts as a DPI (*Deep Packet Inspection*), and identifies situations where 2 or more requests for the same video content are active. When this happens, it notifies the MEO (*Mobile Edge Orchestrator*), which manages the instantiation of a new LTM ME App and the provisioning of the TOF, in order to redirect the traffic to that App at the edge.
   Note: Previously, the TOF was provisioned by the MEO to detect when 2 or more users are watching the same video.
2. The Core service component detects users requesting the same video content and notifies the MEO anytime any user starts or stops watching a video. The MEO has the information about the edge where users are attached, and is checking all the time whether the MEC management conditions (2 or more users watching) is met. When this happens, the MEO will manage the instantiation of the ME App and the provisioning of the TOF.
   Note: The Core service has no means to know from what edge the users come.
3. The Core service detects users are requesting the same video content, and asks the MEO about the edge where the users are attached to. When the core service realizes that 2 or more users are attached in a particular edge, it contacts the MEO to manage the instantiation of the ME App and the provisioning of the TOF.

Once the MEO realizes that a new LTM ME App instance needs to be deployed at a given edge, it identifies the manager for this particular Application, asking for its immediate deployment. When the ME App is fully operational, the TOF is provisioned to start offloading the traffic towards the edge ME App, instead of letting it go to the core, via the traditional path defined by the 3GPP standards.

2.4.4 Implementation challenges for the demo

The main challenges for this scene are the following:

1. Detection of two or more users requesting the same video content. This can be done by the MEC environment, which puts a heavy processing and logics on the TOF, or by the Core service, which puts additional requirements on it.
2. Production of light weighted and fast instantiation of the LTM ME App on the edge, and provisioning of the TOF in order to offload the traffic to the edge ME App. This process shall ensure service continuity, making this process seamless for the user and for the mobile network.
3. Deploying the transmuxing and caching functions as different RFBs. The original content is cached and then the transmuxer generates segments on the fly.

2.4.5 Evaluation plan and validation metrics

The most relevant metrics to be evaluated on this scene are the following:

- Instantiation time
  - Time elapsed between the decision that a new ME App needs to be instantiated in a given edge and the time where the ME App is fully instantiated and operational (in sec or msec).
    Note: addresses Superfluidity references to "Instantiate services on-the-fly" and "Shift them transparently to different locations"
- Bandwidth savings
  - Amount of bandwidth saved by using the edge transmuxing technology (in Mbit/s or %).
- Latency reduction
  - Latency reduction by using the edge technology (in msec or %).
- Traffic loss
  - Amount of traffic lost in a service migration from the core to the edge or vice-versa (in Kbit/s or %).
    Note: may not apply to all scenarios, as this particular one since the usage of DASH in conjunction with traffic rules configuration at the edge, avoids any packet losses
- Cache storage reduction
  - Media segments caching only on the edge (volume or %)

2.4.6 Relevant 5G-PPP metrics

- End-to-End latency of < 1ms
  - MEC contributes significantly to the end-to-end latency reduction, by getting applications close to the users, serving them from the edge.
- 10 times lower energy consumption

- MEC contributes to reduce the energy consumption, by deploying applications at the edge, only when they are required, making a more efficient use of the infrastructure resources.

- 10 times to 100 times higher typical user data rate
  - MEC contributes to the increase of the user data rate by making an efficient use of the bandwidth, in particular the backhaul, between the RAN (Radio Access Network) and the Core.

2.5 Scene 5: Scaling of the transmuxing function

2.5.1 Scene description

A workload, namely Late Transmuxing (LTM), is being run in the 5G deployment. There is a specific end-user, EU1, whom we are examining for the purpose of this description. The use-case assumes though that there are multiple other users of the platform all performing operations similar to the EU1, who are collectively known as EUx. Other users who are using the operator for other services are labelled as neighbour users, NUx.

EU1 would like to watch a high quality video on demand, from a service provider, on their mobile device. EU1 would like to watch the video, free from any delays or interruptions and to watch the video without it being too low quality. In particular EU1 is watching a video stream about a 5G EC Project. The video service provider utilizes video transmuxing so that the video is not stored in multiple different formats but instead is encoded specifically for every end-user.

Other users of the infrastructure, the NUx, though are running machine learning workloads.

2.5.2 Technical trigger to start this scene

The trigger in the previous scene (scene 4), consisted in the operator deciding to migrate the LTM service from the core to the edge as there were sufficient users to justify this effort.

In this scene we assume nothing about the topology of the network, the users and the workloads. The user, EUx is assumed to have an established video service that uses the preceding scenes. The video streaming service provider though is using a Cloud based instance of their workload, which runs on an operator that also hosts multiple other tenants.

The NUx start increasing the network activity between two other Virtual Machine guests that are hosted on the same hardware as the LTM service. The infrastructure owner knows that the LTM provider pays more for a certain level of performance through pre-established service-level agreements. The owner also knows that the NUx workloads cannot be stopped or moved from that hardware due to certain hardware specific features being present.

2.5.3 Technical description and involvement of Superfluidity technology

The network activity between the VMs on the same hardware causes a spike in key metrics associated with the LTM service, which are being monitored and, as such, the following actions are taken:

1) A new LTM service is spawned on a second edge site
2) Temporarily a Quality of Service (QoS) policy is put into place on the hardware with the NUx and LTM workloads that reduces the maximum network activity
3) The load-balancer is reconfigured to redirect traffic to the new instance, LTM2, once it is ready
4) The QoS constraint put in place in step 2 is relaxed

2.5.4 Implementation challenges for the demo

Although an implementation of this scene already exists as Demo-1, for it to be fully fledged we need to introduce more controls in the feedback loop. Currently we can monitor coarse-grained network activity. This only allows the capturing of overall data throughput, which is currently measurable between Virtual Machines on the same hardware.

1) Need better telemetry input as acquired from SNAP. Using this framework from Intel we can then acquire lots of data about a lot of parameters.
2) Selectively choosing which data to monitor and process also requires work, which is currently done in the offline analytics phase. There is some effort needed to pipe the work from the offline analysis back into the orchestration framework
3) Orchestration improvements – to allow for fine-grained control of where we place workloads, will require improvements to the actions available, which are currently limited to start, pause, stop, move. By better tying in with the schedulers and orchestration system we will be able to enact finer control.

2.5.5 Evaluation plan and validation metrics

The evaluation plan is to quantify, in terms of User-Experience, how the video streaming service performs in the scenario where Superfluidity features are not enabled versus when they are enabled.

We will be measuring how long the observation window should be for different types of metrics, including if we should capture any historical information for certain types of data.

The main evaluation though will come through changing the number of users utilizing the workload, as simulated by Citrix Hammer. Hammer can be used to simulate a large number of users, using different types of net workloads, which in this case are simulating users also requesting the video service. Different users can be used to simulate different types of requirements, as specified in their

profiles and the orchestrator can then be checked to see how it assigns users based on the different types of requests it is receiving.

Metrics include:

- Service provisioning time in order of seconds not days

Currently getting administrators to change the allocation of services is a manual task and takes a long time. Automating this will allow the orchestration system to cut down the time taken to react the service incidents and also to react to standard increase and decrease in demand.

- Energy saving – 10 times lower energy consumption

By having a reactive system (with some possibly predictive elements), we can reduce the number of dormant workloads in the infrastructure. Each workload will be fine-tuned and much smaller in space, memory, and CPU requirements as they will be based on smaller VMs, containers and unikernels. In this use-case we will be particularly focused on the size of the LTM instance.

- User experienced data rate

With feedback from the LTM client on end-users equipment, we will also get feedback on the data rate actually achieved from users. We will be looking at seeing how to globally maximize this.

- Monitoring information

To be able to make informed decisions, we will need to gather a set of metrics, which are relevant to the workloads. Without a-priori or offline analysis it is impossible without domain expert information to know which metrics should be monitored. By careful analysis we will be able to determine which metrics are more relevant for particular types of workloads. This will then go towards improving the workload models and is foreseen as an evolving task.

- Reaction time

For an action to be useful it has to be performed in a timeframe from when a condition was detected otherwise at best it won't influence the system and at worse make it unstable.

- Contextual information

To make a correct decision on where workloads should be placed, the models will also need some environmental variables that are dynamic in nature. This contextual awareness will be important for gaining the maximum out of the system as possible.

- Service context analytics

As per the monitoring metric, we will also need to determine the status of the various services that may be a combination of multiple different metrics. The analytics also implicitly suggest processing of data, which means that we will need to dedicate some resources for the processing of this information.

- Standard Network Node Level KPI

For this particular workload we will be investigating network throughput and latency. However as the workloads become more complex and/or there are variants we will be expecting to also read other network parameters (including jitter and errors for example) for determining what the optimal service level involves.

- Quality of Experience Metrics-Analytics

To be able to take more informed decisions, the system will also have to process QoE parameters. For instance a particular user may be suffering from a degraded level, which would otherwise be missed if only coarse grain monitoring is used.

- Scalability

This scenario revolves around the ability to spawn new instances of the Unified Streaming Origin instances. The VIM, the monitoring system and all other constituent parts need to be able to scale to the number of workloads expected in 5G deployments. We therefore need a scalable system that can handle the amount of workloads and decisions that are expected. This includes being able to continue services in the order of ~100s ms (fast service migration).

- Load-balancer

An integral part of this demonstration is being able to use a load-balancer to redirect between heavily loaded and lightly loaded or new services. This redirection forms part of the user experience, whereby the more seamless it is, the less chance that the user will discontinue using the service.

- Location independent scaling

For a truly Superfluid system, the workloads should be executable anywhere in the infrastructure. This means that workloads should be able to run, as much as possible, on other types of hardware while being hardware agnostic. For performance reasons though, it is important that the properties of the hardware are known in case there is a way to accelerate certain workloads. In this scenario, the new instance of Unified Streaming LateTransmuxing server can be set up on any of the compute node instances within an OpenStack managed framework.

- Manageability of the whole platform

The manageability of the platform is important for this scenario as we can only place resources and migrate them within the scope of a single VIM currently. This becomes increasingly important in Scene 6 where we will manage both VMs and containers together.

- Infrastructure scalability

For our platform and demo to grow we need the ability to add new hardware into the deployment and for this then to be accessible by the VIM and the rest of the system, without needing the whole existing infrastructure to suffer degradation or temporary non-availability.

2.5.6 Relevant 5G-PPP metrics

- 10 to 100 times more connected devices.

By moving the processing and decision making towards the end-points of the 5G network we expect that we will be able to support many more users.

- 10 times to 100 times higher typical user data rate.

By offloading network traffic from the backhaul network to closer to the end-points as described in this scene we expect that throughput can be increased considerably.

- 10 times lower energy consumption.

By only provisioning workloads as they are necessary, rather than having to have them pre-provisioned we expect to massively decrease the amount of idle power consumption used.

- Supporting lower cost access to a wider spectrum of applications and services

The Superfluidity platform as described in this scene, enables many more workloads to run on a given set of hardware, which will increase the competitiveness and reduce costs for those services.

- Reduction of the network management OpEx by at least 20% compared to today

A large part of this scenario revolves around moving away from having an administrator that needs to manually intervene for many different types of routine and expected conditions. It will also help to more efficiently utilise the resources, meaning less hardware will be needed and providing a further reduction in management and power OpEx.

2.6 Scene 6: Container implementation of RFBs

2.6.1 User-centric description

Scene 6 assumes that the initial starting conditions are like those presented in scene 5. The difference though is that the content, instead of being a video of a popular 5G project, is instead a collection of advertisements. The difference with advertisements as opposed to video content is that they have a much higher value for the publisher. The user with an interest in this is the advertiser who pays for the video content to be pushed with very little delay and at high quality.

Advertisements are also different in that they are short-lived and ideally should be personalized to the user receiving the content.

2.6.2 Technical trigger to start this scene

A user from scene 5 is starting the next video and in between an advertisement is played from a third-party site, which uses a specialized advertising variant of LTM.

The user, EUx, doesn't have any knowledge or choice in the advertisement that is received but triggers the advertisement action once a video is chosen.

2.6.3 Technical description and involvement of Superfluidity technology

Once the advertisement action has been triggered it is important to quickly start up the service close to the end-user and offer the personalized advertisement content.

Rather than triggering a VM with LTM capabilities, instead a lightweight specialized version of LTM is created near the end-user, which starts up in ~300-500ms. This service then handles the user interactions related to the advertisement. Once the advertisement is completed, the instance is no longer needed and is discarded.

2.6.4 Implementation challenges for the demo

Need to efficiently and quickly transfer the Docker image from the Core to close to the end-user.

The application should then pick up the configuration and state data that is held on the end-user device or close to the end-user and then integrate it with the video content.

Requires changes on the application side to enable the interactive advertisement.

Requires containers management and orchestration integrated with a VM based infrastructure management system (e.g., OpenStack). That is, for example, having the same network to connect VMs and Containers.

2.6.5 Evaluation plan and validation metrics

The time taken to create the service from a cold and warm start (e.g. Docker image not local and in the case where it is local) will be evaluated to understand how long the service takes to start. The latency between the advertisement and the VM / container running the image will also be evaluated to show that the latency is much improved in the case that the content is close to the user.

Also the overall utilization of resources will be monitored and compared to see how well the system performs if the platform uses this technology versus using a-priori setup.

2.6.6 Relevant 5G-PPP metrics

The following metrics were identified as relevant to take into consideration for this scene:

- Tactile internet: latencies of services ~10ms


- Latency between end user and service
- Service provisioning time (<1s)
- Impact on energy usage
- Using a-priori set up services vs. on the fly deployments of personalized service applications.
- Service density

By allowing services to start up on the fly, the number of services available to an end-user can increase by a large proportion, even if in fact they are not on all the time.

2.7 Scene 7: DDoS attack: xFSM detection and unikernel-based VNF mitigation

2.7.1 Scene description

In this scene it is demonstrated the rapid deployment of network functions using Unikernel based VNF to overcome a DDoS attack.

2.7.2 Technical trigger to start this scene

At the beginning of this scene, video streaming is using late transmuxing in the MEC: several LTM transmuxing VMs have been instantiated and the traffic offloader is diverting video streams towards these.

Then a DDoS attack floods the network from the edge. This attack can potentially cause service disruptions.

2.7.3 Technical description and involvement of Superfluidity technology

In the following subsections we describe new Superfluidity components which are introduced in this scene.

Open Packet Processor (OPP)

The Open Packet Processor (OPP) is a stateful SDN switch that allows the execution of extended finite state machine (xFSM) directly at dataplane without interaction with an external controller.

OPP supports the execution of per-flow programmable XFSMs by providing the following features:

1. Programmable definition of a flow
2. Flow state retrieval and update
3. Per Flow registers (data variables) update
4. Enabling function definition

5. Evaluation of condition to trigger per flow state transitions

In this scene, one (or more) OPP switch(es) can be programmed to monitor network traffic at the edge, in order to detect a DDoS attack and proactively mitigate it by dropping or rerouting a portion of the traffic towards other network functions for further deeper online analysis. A major advantage of using OPP in such scenario is the possibility of tracking and reacting to given events without the intervention of a controller, avoiding communication delays and the creation of potential bottlenecks.

It is worth noting that an external orchestrator will still be required. Indeed, in our envisioned scenario, the OPP distributed switches will be able to asynchronously notify the management and orchestration plane about suspicious activities. Thus the orchestrator will be able to create on demand virtual machines towards which the switch(es) will reroute the traffic.

Unikernel based VNFs

Unikernels are virtual machines based on a minimalistic OS and tailored to a single application. Unikernels make an efficient use of computing and memory resources and can reach faster instantiation and boot times than paravirtualized or fully virtualized VMs.

In this scene, we make use of Unikernel-based VNFs which contain the logic to detect and block (or eventually mitigate) DDoS attacks.

More detailed scenario description

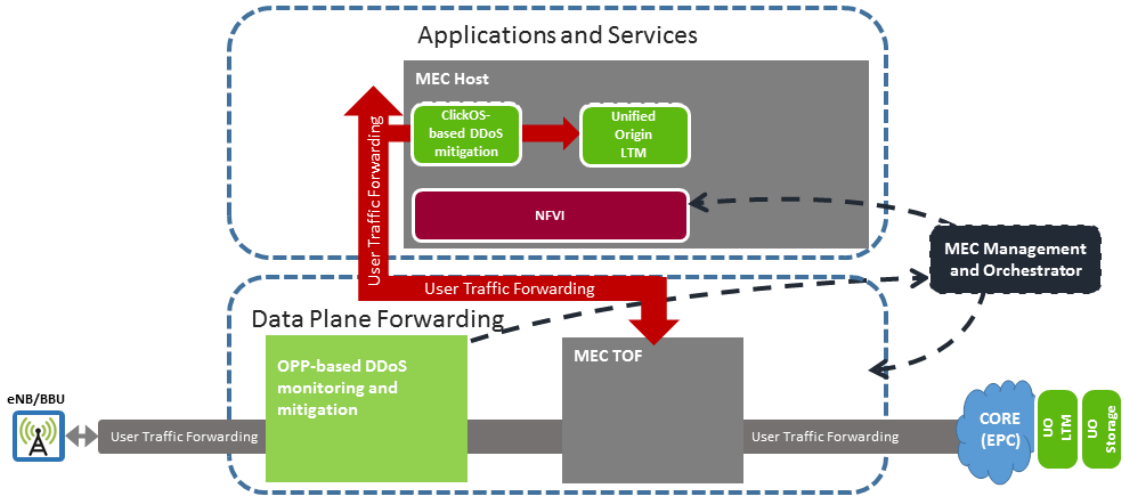


Figure 6 – OPP integration with MEC

Between the edge of the network and the TOF we deploy an instance of the OPP (see Figure). When it detects a DDoS attack, the OPP immediately implements countermeasures to mitigate the attack and then triggers the orchestrator, which performs a sequence of actions:

- Instantiates Unikernel VNFs in the MEC infrastructure. If there are not enough resources available in the MEC infrastructure because these are allocated to transmuxing VNFs, the

video flows which are traversing the MEC are redirected to the core of the network and the resources used by the transmuxing VNFs in the MEC infrastructure are freed. Note that the video service should not be disrupted by this action;

- Instructs the TOF to redirect the traffic containing the DDoS attack to the Unikernel VNFs in the MEC cloud. The Unikernel VNFs are more effective than the OPP in countering the DDoS attack, but their instantiation, although very fast (in the order of milliseconds), requires intervention from the orchestrator. When the above actions are completed, the OPP allows traffic to flow again transparently through it.

Note that when the OPP detects a DDoS attack it immediately (the reaction times are in the order of the microseconds) starts reacting to the DDoS attack, even before contacting the orchestrator. This allows for an immediate mitigation of the attack.

2.7.4 Implementation challenges for the demo

- The MEC server(s) should support the coexistence of Unikernel VMs and the Unified Origin VM.
- Implementation of the DDoS mitigation actions inside the Unikernel VMs.

2.7.5 Evaluation plan and validation metrics

- Reaction time
  - The time elapsed between the start of the DDoS attack and the actual countermeasures taken by the OPP and unikernel VNF.
- Instantiation time
  - The time needed to instantiate the Unikernel-based VNFs.
- Traffic loss
  - The video traffic lost in switching the traffic from the core to the edge (and back).

2.7.6 Relevant 5G-PPP metrics

- Reliability
- Mitigating or countering DDoS attacks improves the reliability of the service

2.8 Scene 8: Alternative offloading function implementation

2.8.1 Scene description

The offloading function is implemented in Fastclick.

2.8.2 Technical trigger to start this scene

In this scene, we refer to a change in the availability of technologies: new technologies at the data plane level become available for realizing the same scenario. The introduction of the new technologies is facilitated by the fact the service is described in terms of RDCL (*RFB Description and Composition Languages*). The different solutions can coexist in different locations in the infrastructure or even in the same location, if they can rely on the same physical resources provided by the NFVI.

In particular, the idea is that the TOF (*Traffic Offloading Function*) is described in terms of RDCLs and then (re) implemented using Fastclick and perhaps also OPP (this is optional, as OPP is already demonstrated in scene 7).

2.8.3 Technical description and involvement of Superfluidity technology

Starting from the existing MEC offloading solution, a new TOF, called *TOF-fastclick* can be added in parallel to the existing TOF. TOF-fastclick is instantiated on a hardware (or virtual) resource identical to the one of the existing TOF (to allow simple comparison of load). A load balancer can be added in front of the two TOFs. The load balancer is initially configured to send 100% of the traffic toward the standard TOF. Then a small percentage of the traffic (e.g. 10%) is diverted toward TOF-fastclick. The monitoring infrastructure allows analysis of the performance of the TOF-fastclick implementation, compared to the existing one.

2.8.4 Implementation challenges for the demo

- A detailed description of the TOF functionality (both data plane and control plane) is needed
- The description of the TOF should be translated in RDCL (e.g. using ETSI modelling ?)
- The implementation effort with fastclick needs to be estimated.

2.8.5 Evaluation plan and validation metrics

- Load on the hardware (or virtual) resources implementing the TOF
- Throughput

2.8.6 Relevant 5G-PPP metrics

- Improvements in performance in terms of throughput (e.g. packets/s) for the same hardware resources (“Peak data rate”)

2.9 Scene 10: Alternative Netscaler implementation of network functions

2.9.1 Scene description

The objective of this scene variation is to validate that a commercially available, operator-grade, Application Delivery Controller (ADC) [1], Citrix NetScaler [2], can be deployed on the Superfluidity platform, to enable:

- 10.A: Comparisons versus using the basic load balancer included with the Virtual Infrastructure Manager (OpenStack: HAProxy or Octavia).
- 10.B: Migration of mobile network services, such as TCP Optimization, Encrypted Video Traffic Management, Content Filtering, which are traditionally deployed behind the Packet Gateway (P-GW), i.e. in the S/Gi-LAN, to the Mobile Edge Computing (MEC) subsystem of Superfluidity.

Scene 10.A is an enhancement of scene 5 (Section 2.5), in the sense that it improves the load balancing and network analytics aspects, while scene 10.B is an extension of scene 3 (Section 2.3), from the standpoint that it validates additional network services, potentially also chaining them, increasing the workload diversity.

2.9.2 Technical trigger to start this scene

The technical trigger for scene 10.A is the same as for scene 5: In reaction to an SLA/KPI degradation, the orchestrator initiates a scale-out action, new service instance(s) are thus deployed, and the infrastructure load balancer (LB) will have to direct/dissect traffic to them. The difference from scene 5 is that, instead of the LB backend used by default by the Virtual Infrastructure Manager (OpenStack: HAProxy), Citrix NetScaler ADC [2] will be used instead.

On the other hand, the technical trigger for scene 10.B, as an extension to scene 3, is the consumption of new services by the end-user (mobile subscriber). Specifically, and depending on the network services that we decide to deploy:

- Encrypted Video Traffic Management: This is triggered by the end-user requesting a video stream from an OTT provider that delivers video using an encrypted transport, e.g. YouTube, Facebook or Netflix.
- Content Filtering: This is triggered by the end-user requesting content that is not allowed, either based on parental profile (not appropriate) or legislation (illegal).
- TCP Optimization: This is transparently triggered when the end-user consumes services that utilize TCP as the transport layer protocol.

2.9.3 Technical description and involvement of Superfluidity technology

The delivery of diverse transport- (Layer 4) and application- (Layer 7) layer services, such as the ones proposed above, traditionally involves deploying large/specialized network appliances, which are

installed in centralized data centres, behind the P-GW, i.e. in the S/Gi-LAN. Such an implementation is the Citrix ByteMobile ATM [4].

In scene 10, the Layer 4 (Load Balancing and TCP Optimization) and Layer 7 (Encrypted Video Traffic Management and Content Filtering) services are deployed in smaller/softwarised forms and closer to the mobile subscriber, in the Mobile Edge Computing (MEC) subsystem of Superfluidity, which was described in scene 4.

Scene 10.A requires the following additions to the Superfluidity platform:

- Deployment of NetScaler ADC instance.
- Deployment of NetScaler Management and Analytics System (MAS) [3], as the Element Management System (EMS) for NetScaler ADC.
- Deployment of the Neutron LBaaS plugin (Load Balancer as a Service) that is required for integrating OpenStack with NetScaler ADC (as infrastructure load balancer). NetScaler MAS provides a function that automates this integration. To facilitate comparisons against scene 5, the NetScaler LBaaS plugin will have to be deployed side-by-side with the existing HAProxy LBaaS plugin.

These new instances (NetScaler ADC and MAS) will be deployed as VMs, most probably the flavours for KVM.

Scene 10.B will require the deployment of additional NetScaler instances. Contrary to the infrastructure LB scenario above (NetScaler VPX), in this case the Docker-container version will be preferable (NetScaler CPX). This is to fulfil the fast and granular service instantiation requirements, considered vitally important for MEC. In that context:

- The NetScaler instances that will implement the TCP Optimization, Encrypted Video Traffic Management and Content Filtering services will have to be deployed on the MEC subsystem of Superfluidity, and integrated with the end-to-end UE – CRAN – eNodeB – vEPC setup.
- Given that TCP Optimization operates complementarily to services that use TCP as network transport, including Late Transmuxing, demonstrating it will require traffic steering or service function chaining.
- The orchestration aspects (NFVO – VFNM – VIM) of deploying the new service instances and chaining the network services will have to be implemented as well, utilising the respective Superfluidity components.

Lastly, demonstrating the new services will require adding new content assets and implementing new traffic profiles in the traffic generator used by Superfluidity (Citrix Hammer).

2.9.4 Implementation challenges for the demo

In certain scenarios, this scene may involve chaining services implemented as monolithic VMs with services implemented as containers. To facilitate this, recent additions to OpenStack, such as Kuryr, will have to be integrated to implement the networking integration between service instances.

One of the key objectives of Superfluidity is to compare traditional in-network services with next generation implementations that utilise lightweight virtualization schemes, data plane acceleration technologies, etc. Using the Docker-container flavour (NetScaler CPX) is a step to this direction, but we would like to compare versus Superfluidity innovations, such as MiniProxy or MiddleClick, that promise even faster service instance instantiation times, more efficient packet processing, smaller injection overhead, higher scalability, etc.

Finally, for evaluating services like TCP optimization, we will have to simulate realistic network conditions. The end-to-end UE – CRAN – eNodeB – vEPC setup implements an idealized mobile network environment. To validate the effectiveness of TCP optimization, we will have to introduce network impediments, such as delay, jitter, loss, etc., either caused by radio bearer behaviours, scheduler policies or network congestion. This may consequently require the addition of a network simulator to the Superfluidity demonstrator setup.

2.9.5 Evaluation plan and validation metrics

Scene 10.A will be evaluated in comparison to scene 5, focusing on the functionality and performance differences between using an operator-grade infrastructure load balancer and a basic implementation. In that context, the same validation metrics will be assessed, particularly in the areas of load balancing and network monitoring, and will be evaluated comparatively to the respective validation metrics of scene 5.

The evaluation plan of Scene 10.B will assess the services that are introduced, specifically in comparison to the scenario they are deployed in the traditional fashion, i.e. in S/Gi-LAN. Due to practical limitations (i.e. inability to fully replicate the traditional architecture), this will most probably be a qualitative evaluation.

2.9.6 Relevant 5G-PPP metrics

- Service Creation Time Reduction
- Network Latency Reduction
- Network Throughput Increase
- Reliability Improvement
- Energy Consumption Reduction

2.10 Scene 11: Infra-structure teardown

In this scene all the RFBs are torn down and the resources are available for others to use. Detail will be added in the next version of the deliverable.

3 Conclusion

This document describes the Superfluidity project's integrated demo. The demo consists of a series of scenes, each of which includes one or several pieces of Superfluidity technology. Overall the demo is intended to validate the project's developments by integrating, evaluating and showcasing our work.

For each scene the document provides:

- A brief description of the scene
- The technical trigger that starts the scene
- A technical description explaining the involvement of Superfluidity technology
- Implementation challenges foreseen for the demo
- Ideas about evaluation and suitable validation metrics
- The potential impact on 5G-PPP metrics

The Appendix briefly outlines our four initial 'stand-alone' demos, which were shown during the Period 1 Review, since they have not been described in previous deliverables.

The document is an interim version of the deliverable. Feedback is welcome in order to improve the demo and make it more useful, since the demo is being built over the remaining year or so of the project and the technology pieces continue to be developed.

4 References

Scene 10

[1] What is an Application Delivery Controller (ADC)? https://www.citrix.com/products/netscaler-adc/resources/what-is-an-adc.html

[2] Citrix NetScaler ADC, https://www.citrix.com/products/netscaler-adc/

[3] Citrix NetScaler MAS, https://www.citrix.com/products/netscaler-management-and-analytics-system/

[4] ByteMobile Adaptive Traffic Management, https://www.citrix.com/products/bytemobile-adaptive-traffic-management/tech-info.html

Annex A: Initial demos

A.1 Demo: Demand-driven orchestration for 5G deployments

Short description

This demonstration highlights the activities around telemetry, monitoring and orchestration. It integrates into a single unified demo a set of components that a good proportion of partners have been working on in Superfluidity.

We use some of the core OpenStack projects as the underlying Virtual Infrastructure Manager (VIM) along with some modifications that enable the use-case. The physical infrastructure has been set up as per the diagram shown in Figure 1, in the BT premises at Ipswich, UK. Access to the partners has been provided via VPN access. The servers S1 through S5 were then set up with Operating Systems; namely CentOS 7 for S1 through S4 and then a bare metal deployment of Citrix Hammer was deployed on S5. Two of the nodes (S1 and S2) are configured as OpenStack infrastructure nodes (e.g. Neutron, Horizon, Cinder etc). Two other nodes are configured as Compute Nodes (S3 and S4), with the final server dedicated as a Hammer node.

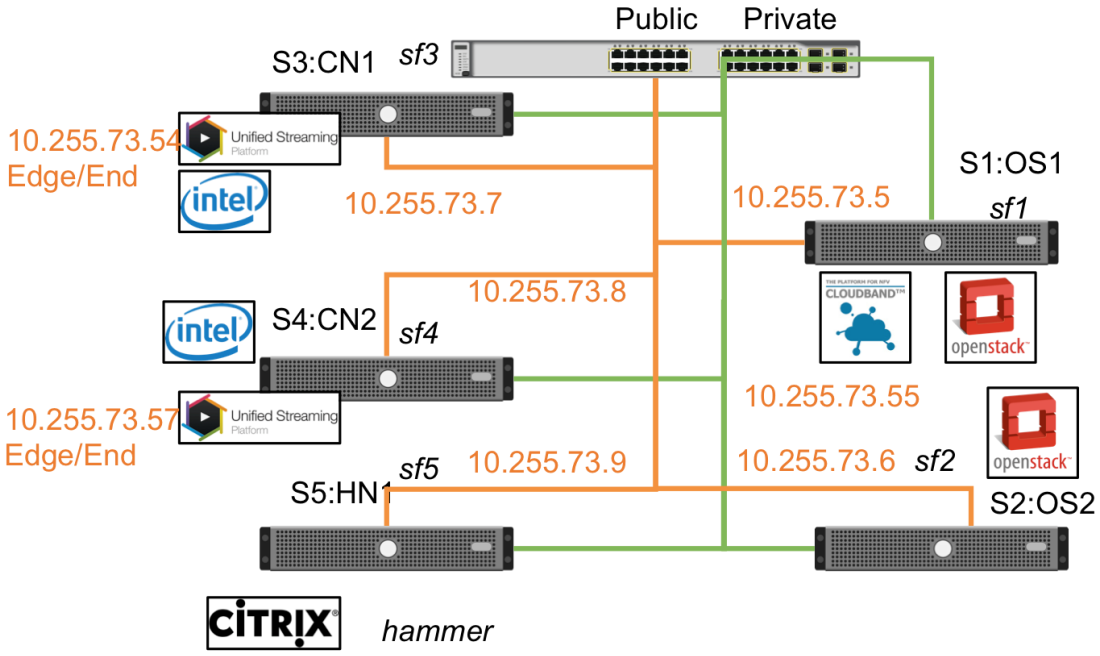


Figure 7 - Superfluidity test-site configuration for this Demo

The general idea with this use-case and associated demonstration is to show that for 5G networks we will need to leverage Cloud technology and also overcome some of the limitations that are present in current Cloud deployments. Currently Cloud deployments typically resemble that of Figure 8, where an administrator sets up an infrastructure and then creates virtual machines or enables a mechanism for end-users to set up virtual machines on the underlying platform. Generally this system is then left in an uncontrolled and weakly monitored state. If there are alerts or problems, these are raised, submitted etc. and eventually actioned upon.

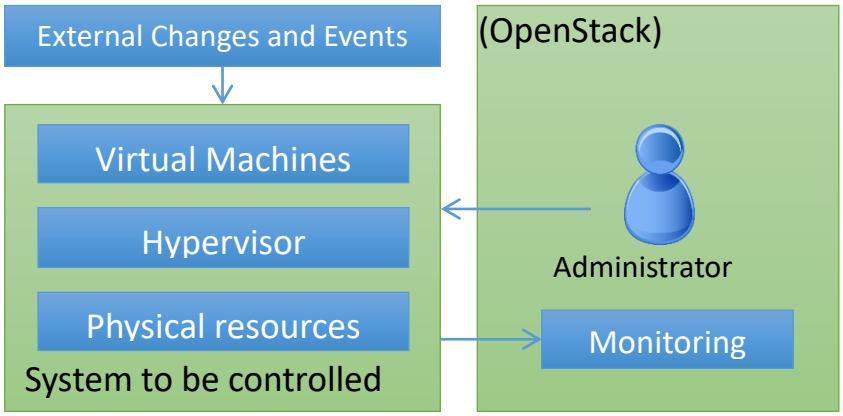


Figure 8 – Before: Administrators used to manage workloads

In Superfluidity we argue that the control loop and decision logic should be performed by the platform itself, given sufficient understanding of what is going on in the platform and adequate descriptive systems, as shown in Figure 3. This then allows the system to react to most conditions in a pre-planned way that takes away the administrator from the critical path on performing steps to resolve certain conditions. This does not completely remove the administrator but allows for better day-to-day functioning of the system and also for Service Level Agreements (SLAs) and Key Performance Indicators (KPIs) to be used to provide sufficient levels of service per contracted agreements.

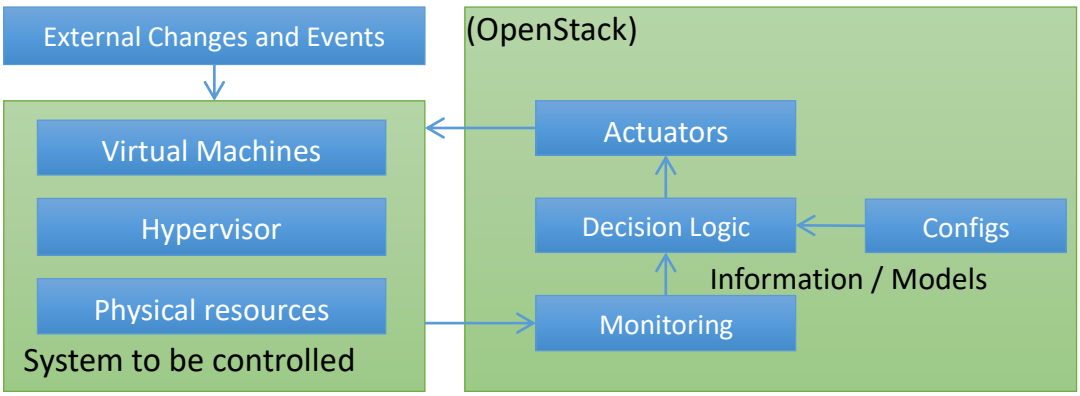


Figure 9 – After: 5G deployments, need (semi-)automated orchestration frameworks

For the demonstration that was shown at the EC project review offline analytics were used to determine that latency and throughput were key parameters for the use-case chosen which was that of playing video content via Unified Origin Streaming product. A high level of throughput would be detected by Ceilometer in two different scenarios. A) The case that noisy neighbours are present and create traffic between each other. B) Overall throughput is too high and we need to scale-out.

Once this was detected and passed to Mistral, it could then be reacted to via HEAT templates that changed parameters in the platform, in two possible ways:

a) Adding QoS rate limiting and changing the load balancer to another existing node;
b) Spawning a new VM, adding that new instance to the load-balancer pool and then changing the weighting to focus on the new instance.

Objectives

With this demonstration we intend to meet the demand for low-latency, high-bandwidth, always-on connections

- Currently have long provision times and waste resources (at the data centre)
- Video-streaming services are pre-provisioned (to try and decrease end-to-end latency)

The number of devices in 5G landscape is growing exponentially

- We expect 1000x volume of traffic of 3G/4G

Superfluidity wants to ensure that KPIs / SLAs are monitored and enforced.

The orchestration framework needs to meet the demands of the use-cases.

The Superfluidity platform can create resources on-demand to react to telemetry allowing for automated, transparent service delivery mechanisms.

We undertake a model based approach for determining resource placement that can decide the best way to utilize the resources.

In parallel to the online monitoring we perform offline low-level analysis of KPI metrics for optimisations.

Superfluidity is designed for scale-independence, allowing for the addition of resources anywhere and is architected to work on heterogeneous hardware platforms.

- Offline service profiling using analytics technologies
- Real-Time Telemetry to monitor a service
- Application driven on-the-go resource scaling
- Cost reduction and quality improvement for video streaming
- Noisy Neighbour interference mitigation by QoS Policing
- Useful for Video Streaming and other bandwidth heavy applications
- Starting point for the Superfluidity architecture that features on-the-go scaling of compute and network resources

Demonstration steps

Scenario A)

1) Video Streaming Starts (demonstrated using a custom javascript player)
2) Traffic generator starts to ramp up
3) A critical load at the server is reached and the user experience could start degrading soon
4) Ceilometer telemetry picks this up and raises an alarm to trigger a scaling based on pre-trained scaling model
5) Scaling is performed by Heat and Mistral
6) Load balancer updated to balance video traffic

Scenario B)

1) Video Streaming scenario with noisy neighbour and no rate policing
2) Video Streaming scenario with noisy neighbour and rate policing (QoS)

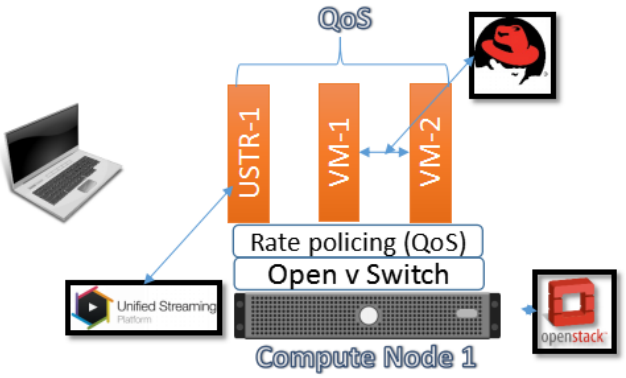


Figure 10 - Demonstrating the noisy neighbour scenario

Evolution

We intend to improve the decision logic feedback part of the demonstration, such that it is based less on human generated input, but rather through templates and enforced behaviours. This will require tooling to allow administrators to more quickly set up sets of actions to undertake.

We also intend to further evaluate the platform and understand how to make different parts of the system react at different rates, suitable for the heterogeneous workloads that are expected.

We also intend to fit the orchestration element as an overarching part of the fully integrated demonstration where the platform can react and meet the demands of users in real-time.

A.2 Demo: Software defined wireless network

Objectives

This demonstration aims to build a modular hybrid fixed and wireless front-haul system that enables high level of programmability. It integrates front-haul in a fully automated cluster based provisioning system for RFB's deployment. It mainly provides instantaneous deployments and instantiation of RFBs to have a minimum downtime.

It also demonstrates how to adapt resource usage to the users' needs in near real-time. it builds a full software wireless network with a common REST based interface for simpler and better programmability. We demonstrate the flexibility of RFBs (RFBs can be redeployed independently over time).

Demonstration steps

We demonstrate the deployment and operation of an end-to-end network in a one click – with a joint NFV and SDN control using RFB principles:

1) SDN controller instantiation
2) EPC based RFBs instantiation
3) Front-haul registration
4) Front-haul first flow (red flow) instantiation
5) BBU instantiation & registration
6) UE connection
7) Front-haul first flow (red flow) deletion
8) Front-haul second flow (green flow) instantiation
9) UE connection

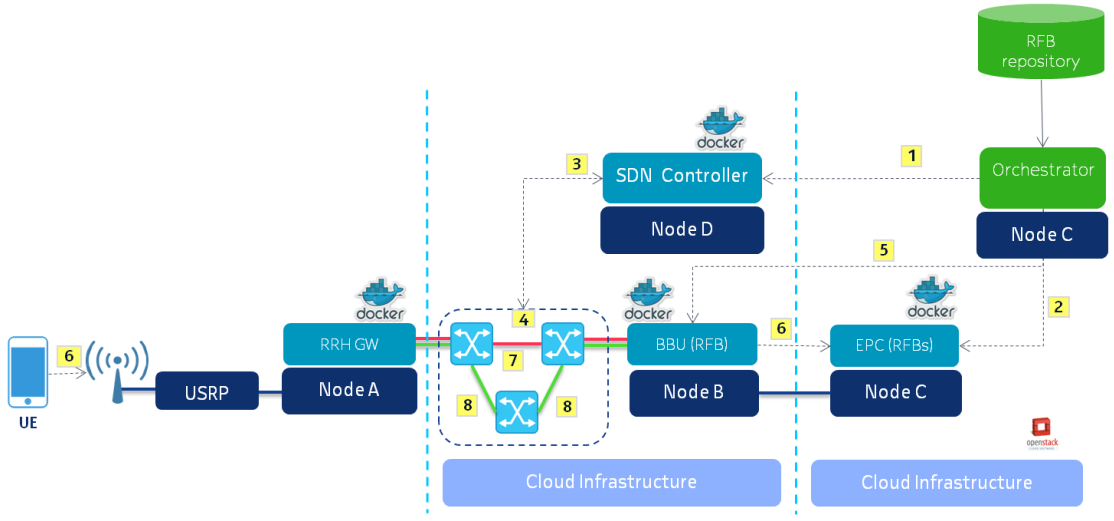


Figure 11: Demo: Software defined wireless network

A.3 Demo: Network configuration verification

Static verification of network behaviour is already proving its value in a wide range of scenarios [Stoenescu-Sigcomm16, Kazemian-NSDI12, etc]. Most of the current approaches exhibit a fundamental shortcoming because verification takes place after deployment. In particular, whenever the network control plane changes a new dataplane is deployed and the verification process is started: first, a model of the network dataplane is built, and the verification process takes place on this model; finally, the results are yielded back to the network operator that decides if any corrections have to be applied or not. Such an operation mode is suboptimal for two reasons:

- Potential bugs can be deployed in production and create effects before the verification process captures them and they are repaired by the operator
- The model of the dataplane may be inaccurate invalidating the results of the analysis.

Our contribution is to perform verification before deployment and to ensure that the deployed dataplane is equivalent to the model. A data plane update is deployed only if it was proven to respect the network policy in place. In order to enable such an approach the network operator has to be provided with a flexible policy specification language that the verification tool can then use as an input in order to verify the validity of the network. The high-level approach is detailed below: given RFB specifications written in SEFL and the desired interconnections, our tool will build a dataplane model that will be verified by Symnet for consistency with the operator's policy. If that is the case, an equivalent dataplane implementation in the P4 language will be automatically generated and deployed.

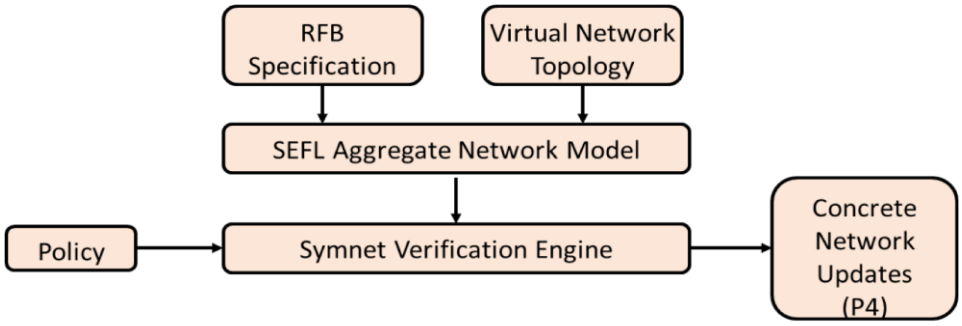


Figure 12: Network verification approach

Demonstration description

To demonstrate our work we will use a setup consisting of a P4-enabled data plane deployed within a mininet network. The input will be a SEFL description of RFBs and their connections in Symnet format (as shown in the first year demo too), together with a desired high-level policy specified in the CTL language.

a) The verification cycle begins with the policy specification phase. This is the only phase which requires human intervention. The requirements are specified using a dialect of CTL (Computational Tree Logic). The demo will explain the CTL policy and also show how changes can be made to specify different requirements.
b) Symnet will then be invoked to verify whether the policy holds; the answer will be either yes, or examples of packets/nodes where the policy does not hold.
c) If the policy holds, a parser from SEFL to P4 will be invoked and generates runnable P4 code automatically. The dataplane will be deployed in mininet, and the demo operator will show that the dataplane behaves as expected by injecting traffic at different ports.
d) The last part of the demo will examine the relationship between the SEFL model and the P4 code, highlighting interesting parts and showing how equivalence is achieved.

A.4 Demo: Video transmuxing and MEC

Short description

This demonstration focuses on the usage of *Mobile Edge Computing* (MEC) for efficient video delivery, using *late transmuxing* (LTM) technologies at the edge. LTM corresponds to the action of encapsulating differently the raw video/audio content, according to the receivers' environment. Doing it at the edge reduces caching requirements, since only the raw content is stored (single format) and only a single stream is transferred in the backhaul, as only the raw content is required at the transmuxing point.

For demonstration purposes, Altice Lab's MEC implementation has been integrated in a virtual deployment of an EPC (OpenEPC) and used to run Unified Origin's LTM. The integration is performed at the S1-U interface, between the eNB and SGW elements. This way, by opening the GTP-U tunnels, the MEC realization is independent from E-UTRAN and EPC functions. In a real environment, the MEC solution would be deployed physically at the same data centre PoP as the eNB.

Objectives

This demonstration shows the execution of services at the edge, using elements being implemented in line with the "early" MEC standardization work. It adopts SDN and NFV related technologies (OVS, OpenFlow, Openstack, etc.) and is aligned with the NFV architecture. The way the MEC Host is implemented guarantees isolation among ME Apps (networking and compute), since they can belong to different 3rd parties (App providers).

In this context, caching gains are significant due to the reduction of different video formats transferred*. For that purpose a *Smart Transmux Edge Cache* with byte range caching implementation was executed.

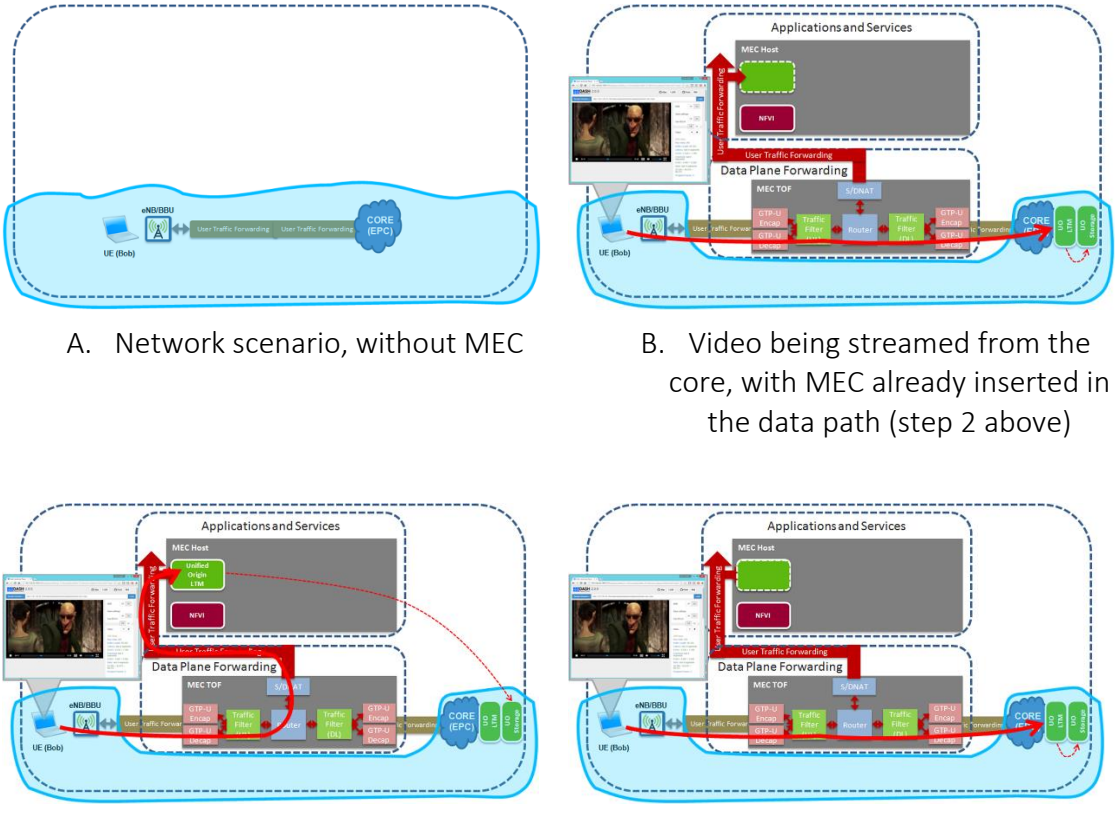
** For simplicity, this demo only uses a single user, not fully demonstrating the actual possible gains of this solution.*

Demonstration steps

1. A user willing to watch a video, types the corresponding URL in the browser
2. The respective video, without any other action, starts being retrieved from a central server, since the typed FQDN/URL corresponds to an IP address located there (behind the SGi interface)
3. At some point in time during the video streaming, and triggered by some operator policy, it is decided to start providing the video from the closest edge to the user (at this stage, these policies are not implemented)
4. Acting on the MEC Host, the LTM application, which was previously on-boarded in the system, is instantiated (in the OpenStack environment)

5. After the LTM instantiation is fully operational, MEC traffic rules at the TOF are configured in order to redirect (offload) the video request to the new LTM at the edge
6. Due to the service protocols in use (DASH) and the way video streaming works, the user will, transparently, start obtaining the video service from the edge LTM, which in turn accesses the central storage for the raw video contents, in a connection outside the GTP-U tunnelling
7. Later, due to another trigger in the operator policies, it is decided to provide the video service back from the core and the traffic flows again between the UE and the streaming server at the core, without being noticed by the user
8. The LTM application instance in the edge is then disposed

Some of the steps are shown in the following figures.



A. Network scenario, without MEC

B. Video being streamed from the core, with MEC already inserted in the data path (step 2 above)

C. After LTM is instantiated and traffic rules are configured, video flows from the backend to the LTM and is streamed from there to the UE (step 6 above)

D. (step 8 above)

Figure 13: Demo of video streaming from MEC

Evolution

This demonstration will be integrated with Demonstrations 1 (Orchestration) and 2 (C-RAN) in the scope of the overall Superfluidity integrated demonstrator. In that scenario, C-RAN and MEC functional blocks will be deployed using suitable orchestration decisions and operations, with ME

Applications (e.g. the LTM) Lifecycle Management (LCM) also being performed by orchestration. Mobility support, associated to UEs' mobility, will also be experimented.