SOFTFIRE

Software Defined Networks and Network Function Virtualisation Testbed within FIRE+

Second Wave of Experiments on the SoftFIRE Platform

NFV and SDN Experiments for 5G Networks

April 2008

Table of Contents

List of Figures

List of Tables

1 Introduction

Before its 2nd Wave of Experiments [1], project SoftFIRE [2] has undergone a major change in it experimenter enablement technologies. Based on the feedback that the project has received from its previous set of experimenters in its 1st Wave of Experiments [3], the project decided to develop a modular and extensible middleware [4], which abstracts the complexity of underlying open-source software, i.e. the infrastructure controllers. This has made it possible to develop specialised software *manager* modules, each responsible for a certain group of functions, e.g. software-defined networking (SDN) [5], network functions virtualisation (NFV) [6], security enablement [7], physical device reservation [8], and experiment monitoring [9]. With the gained knowledge and experience from the 1st Wave of Experiments, and thanks to its more flexible and easy-to-use experimenter manager middleware, the Project was able to support 12 experiments during its 2nd Wave.

In this white paper, the experiments that were successfully deployed on the SoftFIRE platform during its 2nd Wave of Experiments are briefly presented. In doing so, the intention is to present what has been achieved by experimenters on the platform, and the types of NFV [10][11] and SDN [12] experiments that were executed on the platform. These selected experiments are:

- Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP),
- L7-aware Open Virtual Switch supporting programmable network functionalities (AVALON),
- Performance of Software-Defined Wireless Virtual Reality Gaming on SoftFIRE (SWVR),
- Intelligent resource allocation for 5G cloud environments (MODIO),
- Adaptive Video streaming with Software Defined Networking (AVISSOS),
- Dynamic WAN interconnection for multiple NFVI-PoP locations (DEMI),
- IntelligEnt MoNitoring oF NetwORking ServiCEs (Enforce),
- Smart City data management with scalable privacy and security based on distributed access networks via Cellular technology and scalable data storage in SDN-based data centres (Privacity),
- Service Mobility & Policy Management extension (Inmarsat),
- NFV-Shield: A Scalable Intrusion Detection Framework for Network Function Virtualization Ecosystems (NFV-Shield),
- 5G mobile backhaul Network as a Service (5gNaaS),
- Drone based dynamic QoS and fault tolerance in the context of high volume, real time computation in harsh environments (Aerial Insights).

The white paper presents summaries of the architecture, experimentation, and contributions of this set of experiments. Each experiment is presented in a separate section below.

2 CCVP - Cost-efficient Centrality-based VNF Placement Algorithm on the SoftFIRE Platform

Virtual Network Function (VNF) chain placement on a virtualisation platform effects customer quality of experience in a network service and cost of operation for the provider. There are several cost items in the total deployment cost for a VNF: instance license, site (platform) license, and virtual resource usage (computation and communications), to name a few. The publication [13] by *Institut mines Telecom* [14] considers these individual cost items and provides a model for total cost evaluation in VNF chains. This work, called the Cost-efficient Centrality-based VNF Placement and chaining algorithm (CCVP) [13] has a model that takes into account any possible variations in cost weights for different items under different deployment environments. The objective of CCVP is to find the optimal number, location, and chaining strategy of a set of VNFs in such a manner that the provider cost is minimized. The experimenters from *Institut mines Telecom* aimed to evaluate this cost model on a real testbed, i.e. the SoftFIRE platform. With the tests on the SoftFIRE platform, the experimenter aimed to check how the previously obtained simulation results [13] compare with results obtained from a testbed deployment. Main evaluation metrics were *deployment cost* and *latency*.

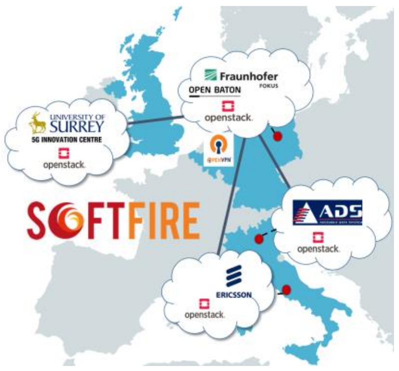

Figure 1. The specific SoftFIRE component testbeds used by the CCVP experiment.

The deployment on the SoftFIRE testbed involved four component testbeds, as highlighted in Figure 1.

There were two types of cost items considered: *Compute* and *communication*. Compute costs were assigned to different testbeds according to the Azure cost model [15].

The VNF instance flavour on OpenStack [16] consumed 1 virtual CPU (vCPU) and 2G virtual RAM (vRAM). The following compute costs were assigned per instance per month in Azure:

ADS = $30.50, Surrey (UK) = $37.20,

Fokus (Germany): $34.97, and Ericsson = $30.50.

Communication cost was taken as the sum of the bandwidth used by a VNF chain in the network. Outbound data cost for each Gb per month between testbeds was considered in the Azure model, whereas the traffic cost inside a testbed was considered to be free [15]. On the other hand, there was a VNF license cost of $1250 involved for each VNF deployed.

On each testbed, the total allocated capacity for this experiment was 2 vCPU and 4G RAM; and each VNF required 1 vCPU and 2G RAM. Data traffic was generated from a server VNF to a client VNF, which were deployed via the SoftFIRE Experiment Manager [17]. In addition, some

VNFs for determining the “quality” of communication by monitoring the data traffic flows were deployed (i.e. a Zabbix [18] server and agents as VNFs).

The experiment evaluated two cases in each of its experimentation scenarios. Case 1 involved *minimum number of instances*, which was a strategy to reduce total computation and deployment license costs. The instance was a firewall (FW) VNF running on a VM. On a setup with three testbeds, this corresponds to a single VNF deployment on one of the testbeds. Case 2 involved *multiple VNFs of the same type*, i.e. one per each target testbed, which is a strategy to reduce end-to-end latency. Although deploying multiple instances decreases the communication cost (i.e. time delay), it also increases the computation and license costs. Hence, the experimenter aimed to study this trade-off on a real testbed environment.

2.2 VNF chains

To establish a VNF chain, the experimenter defined multiple networks, where a VNF in the middle of two others had two ports, one per each network. However, the experimenter observed that intermediary VNFs did not pass packets on reverse paths. As a solution, *allowed address pairs* were added. Chaining was enabled by IP forwarding rules in VNFs, with two sets of rules for intra-testbed and inter-testbed traffic flows.

2.2.1. Intra-testbed using routing rules

To forward traffic between two VNFs in a chain in the same testbed, a rule in one VNF is set to send traffic to the destination point via the other VNF.

2.2.2. Inter-testbed with using Netfilter

The Linux OS has a packet filter framework called *netfilter* [19]. This framework enables a Linux machine with an appropriate number of network cards (interfaces) to become a router capable of network address translation (NAT). Using the *iptables* utility, complex rules were created for packet modification and filtering.

Two scenarios were tested on the SoftFIRE platform:

- *Scenario 1:* Fokus [20] and Surrey [21] testbeds were used in the experiments. Each testbed has a client-server VNF pair, and data traffic is from the server VNF to the client VNF of that testbed. This scenario involves a Firewall (FW) VNF, which is placed along the route of data traffic, i.e. it inspects traffic flows in the chain. Two traffic flows, namely T1 and T2, were used in experiments, with T2 carrying three times the traffic carried by T1:

Traffic 1 (T1): Fokus → Fokus

Traffic 2 (T2): Surrey → Surrey

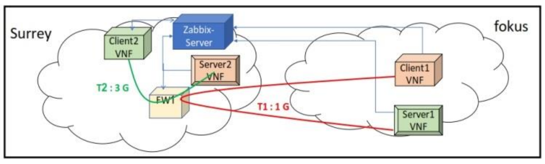

Figure 2. *Scenario 1 (Case 1):* Single firewall (FW) instance for two testbeds.

In this scenario, two cases were tested, as follows:

Case 1: One FW instance per testbed was deployed, and inspects the traffic between the server and the client. This is shown in Figure 2.

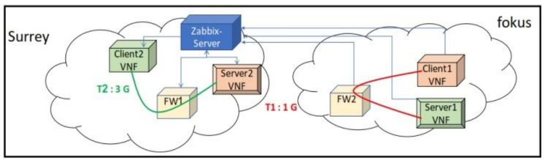

Figure 3. *Scenario 1 (Case 2):* Dedicated firewall (FW) instance per testbed.

Case 2: There is a single FW instance deployed on one of the testbeds, which checks the data traffic flows of both testbeds. This is shown in Figure 3.

In Case 2, the total delay for T1 is around 0.194 seconds less than that in Case 1, while the total cost of the T1 and T2 is $1174.72 more. CCVP selects Case 2 (cheaper option).

- *Scenario 2:* Placement of a chain of two types of VNFs, with traffic between three testbeds.

The chain consisted of two VNFs: a Firewall VNF and a QoS VNF. Traffic flows between ADS, Ericsson, and Fokus testbeds were generated. This scenario was also specific to a case where the data load was not high. Furthermore, traffic T2 carried 3 times the load as T1.

Traffic 1: ADS → Ericsson,

Traffic 2: ADS → Fokus.

There were two cases tested in this scenario:

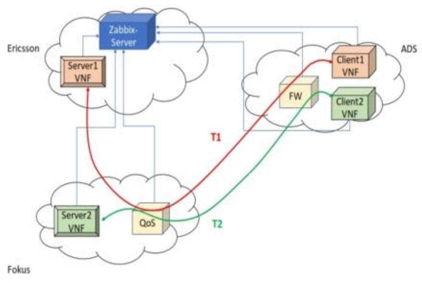

Figure 4. *Scenario 2, Case 1:* Single QoS VNF.

Case 1: Both traffic flows pass through the same single QoS inspection VNF which was deployed on the Fokus testbed. This is shown in Figure 4.

Case 2: Separate QoS VNFs were deployed in Ericsson and Fokus testbeds, each of which inspected the traffic on the corresponding testbed. This is shown in Figure 5.

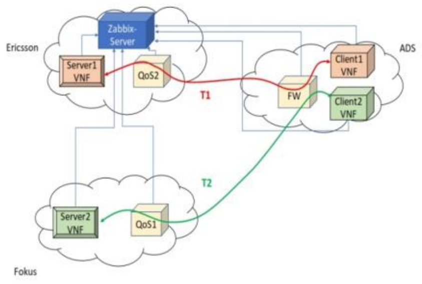

Figure 5. *Scenario 2, Case 2:* Dedicated QoS VNF per testbed.

A total amount of 10 MB was sent for T1 in both cases, and Case 1 had 40 seconds more delay, whereas the cost for the operator was $1287 less. Based on this, CCVP selected Case 1 as the more favourable option.

In general, the algorithm favours the cheaper option; however when the data amount is higher, communication costs become more favourable, which triggers CCVP to choose deployment of multiple VNFs.

2.3 Conclusion

The objective of the CCVP experiment was to find the optimal number of VNFs along with their locations and chaining among them in such a way to minimise the overall cost. It was shown that while deploying more instances can decrease time delay in delivering content, it may also lead to increase in total cost, due to license and computational costs incurred to network operators. In high load conditions though, communication cost is the overwhelming factor.

3 Avalon

The experiment Avalon aimed at identifying traffic patterns in the virtualisation platform asynchronously, without adversely affecting system performance. To achieve this goal, the experimenters implemented dynamic service function chains that perform packet inspection in Layer 7, rather than Layer 2 to 4. The key technology was a modified version of Open Virtual Switch (OVS) [22] implementation, which made it possible to generate a set of NFV services that form an interconnected series of VNFs. The sequence of OVS VNFs where programmed to perform traffic forwarding and steering. By identifying and classifying traffic based on application criteria, an SDN controller can program these OVS elements to treat traffic suitably.

For the 2nd Wave of Experiments on the SoftFIRE platform, the experimenters from the company Eight Bells Ltd. [23] designed and developed different types of new L7-aware OVS VNFs: (a) *a Classifier* which can classify traffic flows, (b) *a Firewall*, and (c) *a Rate Limiter* (RL) providing QoS support mechanism, and (d) *an Enabler* for programmable L7 Service Chaining.

3.2 OVS as a firewall or QoS enforcer

Two simple configurations of L7-aware OVS were developed: The first allows OVS to perform like a L7 Firewall and the second can perform QoS enforcement per application, i.e. a rate limiter which can limit the bandwidth of specific traffic flows tagged by the Classifier.

3.3 OVS for programmable L7 service chains

Secondly, the experimenters designed an OVS which was programmable to be a VNF that can be part of a L7 Service Function Chain (SFC) [24][25], by means of adding an interface that can interact with an external Deep Packet Inspection (DPI) VNF.

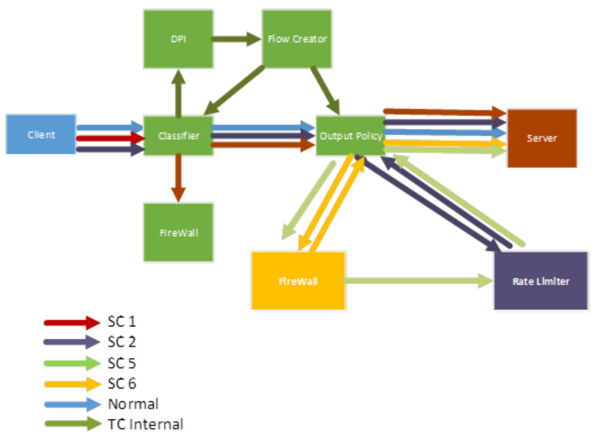

Figure 6. AVALON service chains.

The experiments with this type of OVS had the aim to demonstrate the benefits of dynamic programmable function chaining. With an intelligent traffic steering mechanism, it was possible to choose the best number and sequence (chain) of service functions. This dynamic SFC is based on L7 inspection (Layer 7), i.e. traffic identification and classification at the application layer.

The choice to host DPI functionality externally from OVS inside a VNF (and not as an embedded flow classifier in OVS) was carefully selected due to the following reasons:

- ✓ The DPI VNF can remain as a self-contained and scalable VNF that can be used on demand.
- ✓ The OVS is in charge of only traffic steering, in an efficient and predictable way.

By design, the new L7 OVS targets at real-time packet processing and with the support of DPDK packet processing libraries, and it can fulfil real-time carrier-grade switching requirements of telecom operators.

In the experiments, Avalon provided SFC composed of these different types of VNFs. The OVS VNFs were applied differently to different application traffic flows:

- SSL traffic goes through DPI, Firewall, and Rate Limiter VNFs,
- BitTorrent traffic goes through DPI and Rate Limiter VNF,
- FTP, as a use case of general application data, goes through DPI and Firewall VNFs.

Traffic was tagged using the *type of service* (ToS) header. The DPI engine was based on the open source project nDPI [26]. This component is able to detect HTTP/FTP/BitTorrent traffic even if it is directed to a different port than the default port.

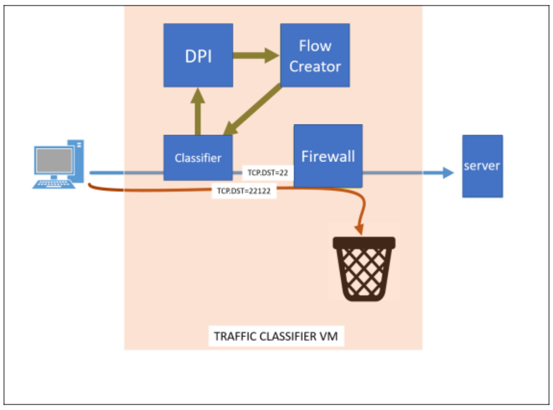

Figure 7. AVALON experiment configuration 1: Allow connections to TCP port 22 only.

Avalon experimenters performed two types of experiments as below. In the first configuration, firewall capabilities that can be developed with OVS are demonstrated. This is shown in Figure 7. Upon detecting connections on TCP port 22122, DPI informs Flow Creator (SDN controller), which then programs the Classifier so that packets on that connection are dropped.

In the second configuration, Avalon demonstrated QoS enforcement capabilities per application. The experiment involved permitting FTP traffic to be forwarded at full speed whereas HTTP traffic was limited to a small rate (0.5 Mbps) to allow the receiver to verify that traffic limiting functionality is in place. Hence, different traffic types were treated differently based on DPI analysis. This is illustrated in Figure 8.

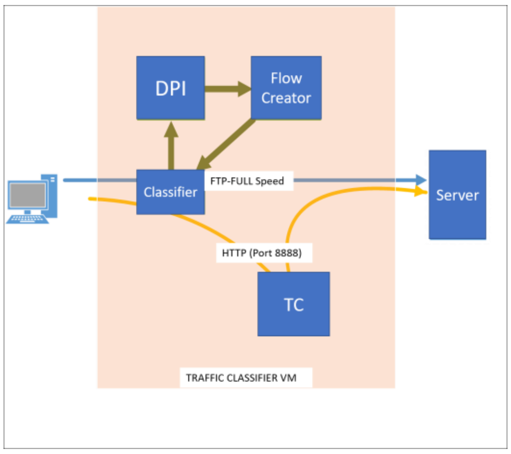

Figure 8. Rate limiting experiment.

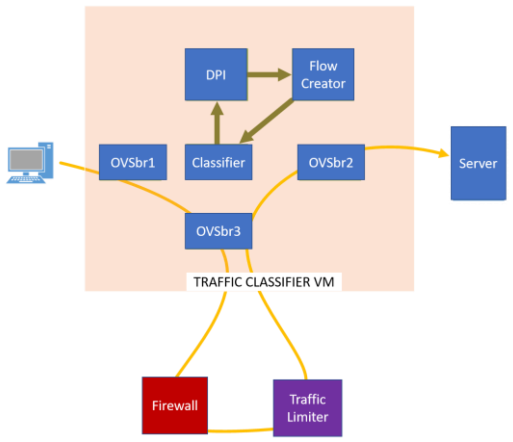

Figure 9. Firewall and Rate Limiter SFC in AVALON.

In the third configuration in Avalon, specific traffic flows were steered according to Service Chain configuration to relevant VNFs. This is shown in Figure 9. A client VM initiated HTTPS requests towards port 443 using the wget application. nDPI libraries detected HTTPS traffic and identified it as SSL flows. Flow provisioner software provisioned OVSbr1 and OVSbr2 bridges with flows containing the client IP address, the server IP address, and the server TCP port to be processed by a specific flow table, which contained the information to mark all packets with ToS value 0x40, and then forwarded them via OVSbr3 Bridge to the Firewall VNF. The configuration in the Firewall VNF applied a security policy for ToS 0x40 and then forwarded traffic towards to the Rate Limiter VNF without modifying the ToS value in the IP header. Finally, the Rate Limiter VNF applied a maximum rate value matching its configuration for 0x40 value and sent traffic to the Traffic Classifier VNF to be delivered to the Apache server.

The final configuration of the Avalon experiment was about Dynamic Service Flow provisioning of traffic patterns that are not so easily detected or cannot be classified with typical Layer3 or Layer4 tuples. This is illustrated in Figure 10. Peer-to-peer traffic was established over random TCP and UDP ports which cannot be detected by monitoring well-known ports used by standard TCP/IP protocols. To address this, the experimenter steered BitTorrent traffic to Traffic Limiter VNF, and FTP traffic to the Firewall VNF.

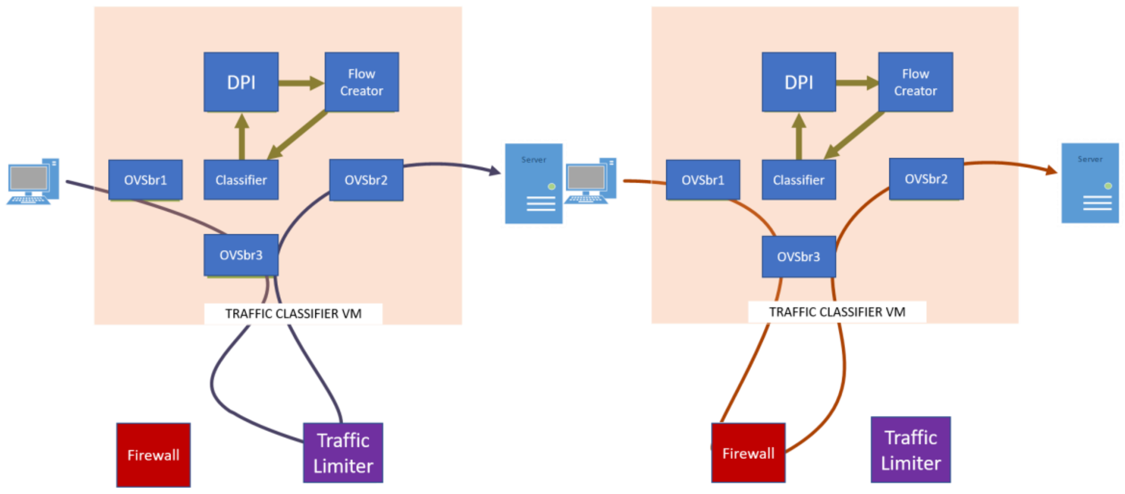

Figure 10. Dynamic service flow provisioning with Avalon.

3.4 Conclusion

The Avalon project has provided a number of conclusive remarks. First, traffic classification with DPI should be the first operation in a service chain to classify traffic flows according to specified/required criteria. Any flow that do not match a classification scheme should not be delayed and must be passed without modification. Secondly, OVS must be operated in kernel space to reduce delay, and must be operated by an SDN controller for dynamic traffic steering. Third, automated operations must be favoured so as to minimise service interruption in cloud services. Finally, service decomposition to multiple VMs is favourable for flexibility and dynamicity, despite a single VM might be sufficient capacity-wise.

4 Performance of Software-defined Wireless Virtual Reality Gaming on SoftFIRE (SWVR)

Real-time Virtual Reality (VR) gaming has become a promising but challenging technology for the next generation of multimedia systems. Different VR headsets have been so far developed and offered in the market, such as SONY PlayStation VR, Facebook Oculus RIFT, HTC Vive, and so on. To provide high quality VR gaming experience, off-the-shelf VR headsets usually demand a wired connection due to the requirement for high bandwidth. The controllers are powerful gaming units, such as Personal Computer (PC) or a custom gaming controller (e.g. PlayStation). In summary, VR applications have high hardware and connection bandwidth requirements, which is often a limitation for mobility scenarios.

To overcome the above-mentioned issues, the company DozeroTech [27] has developed a software backend system that supports *wireless* VR gaming applications over common wireless technologies like, e.g., WiFi, and 4G/5G. Today, there are three typical solutions to make wireless based VR gaming applications a reality:

- *Advanced encoding/decoding algorithms*, which however may increase user plane latency in VR gaming applications,
- *Specialised wireless technologies* like WiGig [28], which do not support connectivity to 4G/5G mobile networks,
- *Interpolation algorithms* for image downscaling/upscaling. Specifically, the native images of a game are downscaled at the game server side before transmission, while the downscaled image can be accordingly upscaled at the game client side. As such, the bandwidth usage for data transmission can be significantly reduced. However, the higher the image quality requirements are, the more computational complexity the algorithm has and the more resources (e.g., memory, CPU) are needed,
- *Using extra information* retrieved from multiple images in order to enhance the quality of a particular image. However, this solution may lead to higher video processing latency, and hence is not suitable for real-time VR gaming scenarios.

The unique approach pursued by DozeroTech was based on multiple important factors in VR gaming scenarios, such as image quality, processing time, cost, portability, and energy efficiency. The solution was submitted as a patent application to the Swedish Patent Office.

The experiment's main goal on the SoftFIRE testbed was to evaluate the performance of this solution in a real 5G virtualisation platform for mobile networking. The image downscaling/upscaling method and the VR functionality were integrated in an open-source cloud gaming system called GamingAnywhere (GA) [29]. Experimental results showed that the performance of the system well met the KPI verification requirements for Quality of Service (QoS) and Quality of Experience (QoE).

4.2 Architecture

Figure 11 illustrates the deployment architecture. FOKUS 5G core [30] provided access to the virtualisation server to a mobile equipment (Samsung Galaxy S8+), which was running the experimenter's Proxy Server VM that was deployed via the SoftFIRE Experiment Manager [17]. The VR game server had a wired connection to the FOKUS testbed and had connectivity to the local virtualisation platform of the SoftFIRE network. Hence, the game server had communication with the mobile terminal and the SWVR proxy server through the SoftFIRE network and the FOKUS 5G network base station. The SWVR proxy server collected statistical information about the communication between the game server and the game client. The collected information was used for performance analysis.

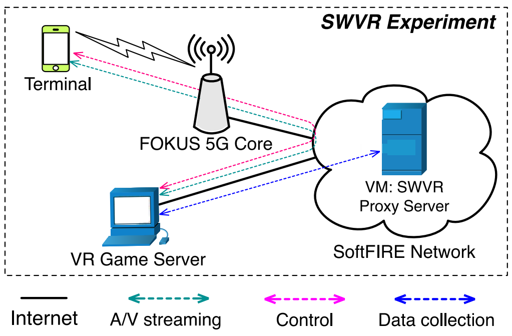

Figure 11. SWVR setup and network topology.

4.3 Key performance indicators

Three key performance indicators (KPIs) were used in the experiments to evaluate the effectiveness of the experimentation system, which are as follows:

4.3.1. Percent loss in image resolution

The game server downscaled each original image (game frame) generated by the game server and streamed it to the client. The game client upscaled the downscaled image for further rendering. The key target in this process was that the upscaled image should have the same resolution as the original image. Original images that were created by the game server had a resolution of 2200 pixels x 1080 pixels. The tolerance level in this KPI was a maximum of 10% degradation in received image resolution.

4.3.2. Round trip time (RTT)

In this experiment, round trip time in the context of a VR game was essentially considered to be the time difference between the instance a user takes a game action (button click, console input, etc.) to the game server and the instance some feedback is received on the user's screen. Experiments on the SoftFIRE architecture as defined above showed that average RTT was 76.196 ms. This met the experimenter's requirement of a maximum 120 ms within a 10% tolerance margin.

4.3.3. Structural similarity index

Structural SIMilarity (SSIM) Index is referred to as a way of evaluating the image quality of a VR game video by comparing the upscaled game frame at the client side with the native game frame generated at the server side. Using a downscale factor of 0.5, average SSIM was found to be 0.8801, which met the experimenter's requirement for an SSIM of 0.8, with a 10% tolerance margin.

5 MODIO - Intelligent resource allocation for 5G cloud environments

State-of-the-art NFVO autoscaling mechanisms are often manually configured. Specifically, autoscaling parameters, i.e. NFV *scale-in/out* and the *cooldown period* parameters, are manually set and do not change over time, irrespective of the incoming client load. This approach has important drawbacks. First, in case of a sharp rise in demand for VNFs from the clients, it is quite likely for the autoscaler's fixed step *scale-out* action to prove insufficient to keep up with the high load, leading to SLA violations. Secondly, a sharp drop in demand combined with a small *scale-in* step value and/or high cooldown period causes waste of resources and unnecessary power consumption of the telco cloud infrastructure. Third, similar scenarios can be outlined in cases of high *scale-in* or *scale-out* values being statically configured within an autoscaler. All such drawbacks stem from the same underlying problem within the autoscaler's engine: *the impedance mismatch between the actual demand and the implemented autoscaling policy within the NFVO*.

The "*Intelligent resource allocation for 5G cloud environments* experiment" was designed by Modio Computing [31]. The experiment implemented and validated a novel computational intelligence mechanism, which dynamically determines appropriate values of OpenBaton's [32] *scale-in/out* parameters[1]. The mechanism incorporates forecasting algorithms to predict the upcoming system load in order to help determine OpenBaton's autoscaling action parameters.

The experimenter compared the performance between:

- The current static auto-scaling mechanism in OpenBaton where the autoscaling policy's specific action parameters are manually configured
- The experimenter's machine-learning approach for dynamically generating the policy content and submitting the resulting policies to OpenBaton.

5.2 Deployment of the experiment on SoftFIRE

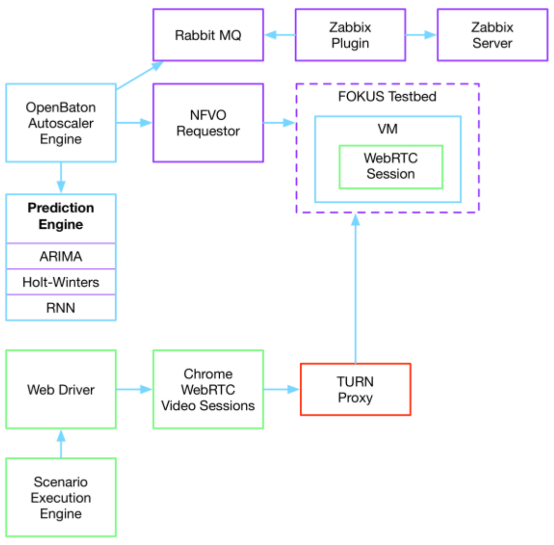

Figure 12. MODIO experiment architecture.

The high-level architecture is depicted in Figure 12. The prediction engine leveraged Modio's commercial Qiqbus streaming analytics platform to predict upcoming resource allocation

[1] The work to derive a performant cooldown parameter constitutes ongoing Modio work towards MVP development.

adjustment requirements for each active network service. Load forecasts were generated by three distinct timeseries prediction algorithms, namely, ARIMA, Holt-Winters, and Recurrent Neural Networks (RNN).

To evaluate each forecasting approach, a number of virtual machines (VMs) were instantiated within the FOKUS testbed in order to host the WebRTC service and emulate a telco NFV cloud. The client load was generated by a cluster of VMs located in Modio's private Google cloud where multiple WebRTC sessions were launched from Google Chrome processes. All sessions were tunnelled through a *TURN* server to bypass NAT traversal issues. The number and duration of the client sessions were handled by *Selenium* and they were configured by experimentation scenario scripts.

5.3 Experiment scenarios

Scenario 1: Under- provisioned number of VNFs (scale-out)

The experimenter first evaluated the performance of SLA guarantees provided by the predictive autoscaling approach. An insufficient number of VNFs were initially instantiated at FOKUS for serving incoming WebRTC sessions from the Virtual Machines within the Google cloud. *In scenarios with fast increase and decrease of demand as well as with high-sustained load*, it was validated that predictive algorithms (ARIMA, Hold Winters, RNN) improved the performance of the standard autoscaler of OpenBaton which does not currently use predictive models and performs scale-out operations with fixed-step scale-out autoscaling policies. The effect of our approach is summarised in Table I below.

Table 1. Ratio of sessions per VMs in patterns including fast increase and decrease of demand and high-sustained load

Algorithm	**Measured metrics**		
	Active VM time (seconds)	**SLA-Conforming Session Time (seconds)**	**Ratio (Sessions/VM)**
ARIMA	544	405	0.744485294
Holt Winters	259	320	1.235521236
RNN	365	240	0.657534247
Standard	153	38	0.248366013

However, in other experiments with different traffic patterns, OpenBaton's standard autoscaler exhibited *slightly worse performance* than its predictive counterparts. This is due to the slow start-up time of OpenStack VMs during the scale-out phase. Specifically, despite the fact that predictive algorithms are able to capture the pattern of the load increase, the significant time taken by OpenStack to allocate additional VMs (typically ~120 secs) hindered the potential benefits of Modio's approach. This is a lesson learned and is being addressed by employing containerized environments instead of virtual machines.

Scenario 2: Over-provisioned number of VNFs (scale-in)

The performance of Modio's approach against OpenBaton's default autoscaler was evaluated for different over-provisioned states, involving more VNFs than necessary. One such state was achieved by first increasing the number of WebRTC clients and then reducing their number and thus the total load which was received as input to OpenBaton NFVO.

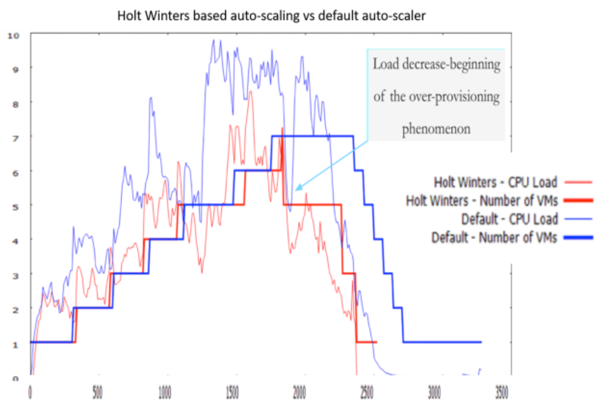

Figure 13. Scenario: Over-provisioned number of VMs. Comparison of default auto-scaler and Holt Winters.

Figure 13 depicts the performance of an indicative experiment that involved over-provisioning of resources, which used the Holt-Winters algorithm. During the scale-in phase, the figure demonstrates Holt-Winter's capability to decommission WebRTC VMs compared to the default autoscaler of OpenBaton. This is due to the fact that the machine learning approach is able to quickly detect the sudden load decrease and trigger appropriate scale-in actions. This capability cannot be offered by static autoscaling mechanisms. It is important to note that, in this experiment, the default autoscaler spent *4880 CPU-seconds* during the over-provisioning phase, while the Holt-Winters autoscaler within the autoscaling engine only occupied *2550 CPU-seconds*, resulting in an energy saving of approximately 48% for the provider hosting the WebRTC 5G service. Analogous energy saving outcomes have been observed when using different forecasting approaches.

5.4 Conclusion

For the over-provisioned scenario, the experiment clearly led to the conclusion that the most effective method in terms of meeting SLA requirements of WebRTC clients was the ARIMA algorithm, as presented in Figure 14[2]. Furthermore, Holt-Winters and RNN both outperformed the default OpenBaton autoscaler, being able to scale-in resources more effectively. Moreover all validated forecasting techniques have resulted in energy savings for the NFV cloud provider (emulated by FOKUS testbed and OpenBaton) since all technqiues have resulted that the experiment used less resources to serve the input load. A video describing Modio's experiment is available on YouTube [33]. Ongoing work is carried out by Modio in order to produce a minimum viable product and secure the implementation through filing of a patent.

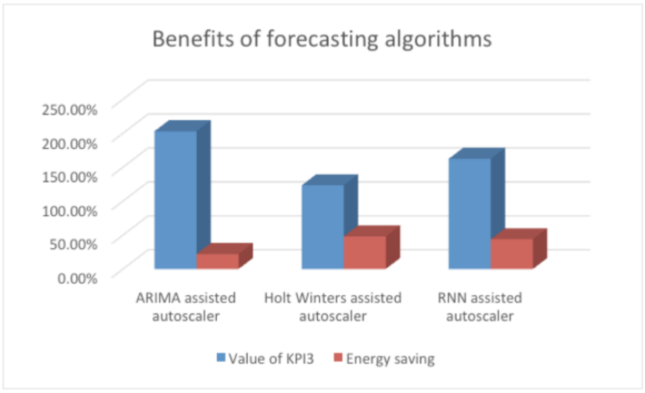

Figure 14. KPI3 and energy savings in over provisioned use cases

[2]KPI 3 in Figure 14 the refers to KPI that measures if the approach outperforms OpenBaton's default autoscaler in terms of guaranteeing the SLA for WebRTC session for over-provisioned use cases.

6 Avissos

Wireless networks, which have stringent resource limitations and fast-changing conditions require better architectures and a cross-layer approach to support high-volume and bandwidth-intensive applications, like video streaming. The experiment, called *Adaptive Video streaming with Software Defined Networking,* from the company StreamOWL [34] was on Quality of Experience (QoE) for video streaming applications. The experiment targeted at improving QoE of video streaming users by means of a cross-layer approach that considered end-to-end data paths. This adaptive solution considered varying network conditions of video clients, and determined optimised routing paths, also taking into account the network's topology and resources. At the application layer, the Dynamic Adaptive Streaming over HTTP (DASH) approach enabled seamless adaptation of the video client to current network conditions.

6.2 Architecture

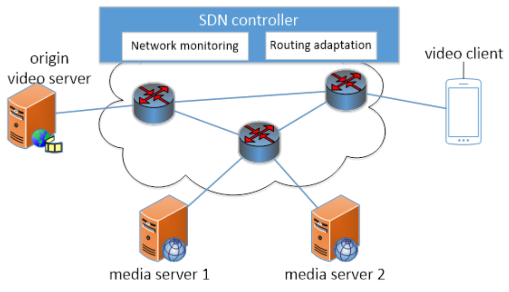

Figure 15. Video streaming SDN architecture.

The experiment is simply illustrated in Figure 15. Initially, the video client requested a video from the video server, which communicated with an SDN controller that monitored network conditions (e.g. congestion, delay) to decide which media server is more appropriate for content delivery to the video client by changing the routing path. The routing decision was based on optimization of the end-user QoE. The SoftFIRE federated testbed provided multiple testbeds at different locations, supporting SDN functions and allowing routing adaptation.

The solution considered (*i*) network topology, (*ii*) link capacities, and (*iii*) the specific QoE requirements of each application. The path selection process was adaptive to changing network conditions, e.g. jitter, delay, bandwidth, node/link failure. The quality assessment model was based on QoE assessment methods of audio-visual quality assessment for adaptive streaming over HTTP, as described in ITU-T Recommendation P.1203 [35], which has been validated in many different network conditions. Performance was evaluated for various metrics, such as start-up delay, number/frequency of rebuffering events, sustained video bitrate, and objective Mean Opinion Score (MOS).

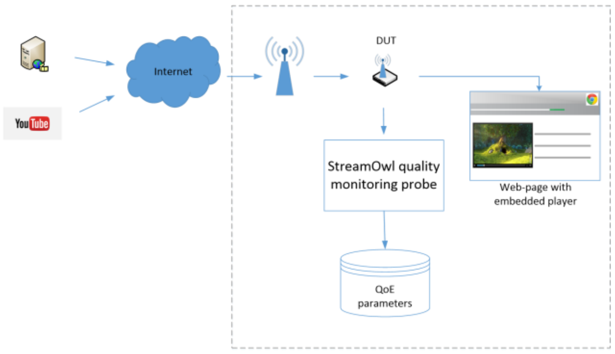
Internet
DUT
StreamOwl quality monitoring probe
Web-page with embedded player
QoE parameters
YouTube

Figure 16. AVISSOS experiment setup.

The experiment setup is illustrated in Figure 16, which includes an HTML5 player (based on the YouTube IFrame API) which provides measurements of video resolution, startup delay, number of re-buffering events and duration. This web page was hosted at the Experimenter’s site, and the video was fetched from YouTube.

The setup also included a DASH client which was responsible for downloading the input video and controlling the adaptation strategy based on the network conditions of the client. The client chose to request segments with lower resolution if it detected a drop in the throughput of the previous segments, in order to avoid buffer under-runs which would result in interruptions in video playback. The video client was *mplayer*, since it can operate in a headless mode and is a lightweight and robust application which can handle DASH streams. In the case of a stand-alone video client, the proprietary “*StreamOwl over-the-top (OTT) probe*” was used, which monitors Internet traffic in a passive and unobtrusive way using Deep Packet Inspection (DPI) techniques and evaluating the impact of service parameters and performance metrics (e.g. video bitrate, network degradations, type of service, etc.) on user-perceived quality.

The algorithm for QoE estimation was based on the ITU-T Recommendation P.1201-PD [36], the international standard for quality assessment in IP-based applications. More specifically, the probe enabled the monitoring of the following key performance/quality indicators:

- Audio/video QoE (in terms of MOS),
- Video startup delay,
- Video rebuffering/freezing (timestamp+duration of each event),
- Video quality switches in ABR testing,
- Segment throughput and bitrate,
- Segment download time,
- Number of video representations (analysed by the manifest file),
- Audio/video quality of each segment,
- HTTP errors (4xx/5xx messages).

6.3 Results

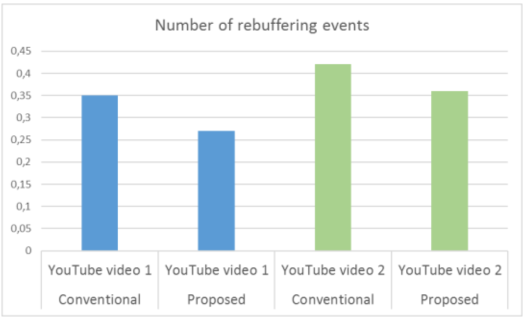

Figure 17. Re-buffering events.

The results of the StreamOWL OTT probe were stored locally in the user equipment and transferred to a local repository after experiment execution. In contrast, the results from the embedded HTML5 player for the YouTube test were stored directly in the StreamOWL server which hosted the web-page, immediately after the end of the video playback.

The other nodes, which are depicted in Figure 15, were configured as intermediate nodes (switches), so that the traffic could be routed through a specific combination of them to the final node which contained the video client. In this respect, it was possible to emulate the links between the nodes with the desired level of background traffic (or signal attenuation), to modify the throughput and therefore to emulate different levels of traffic bottlenecks.

Experimental results showed improvement of video-related metrics, such as number and duration of re-buffering events, and Mean Opinion Score. Sample results are shown in Figure 17, Figure 18 and Figure 19.

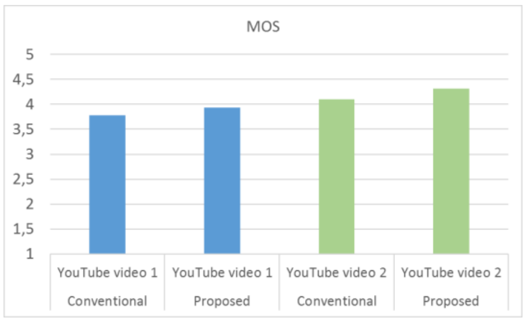

Figure 18. Re-buffering time.

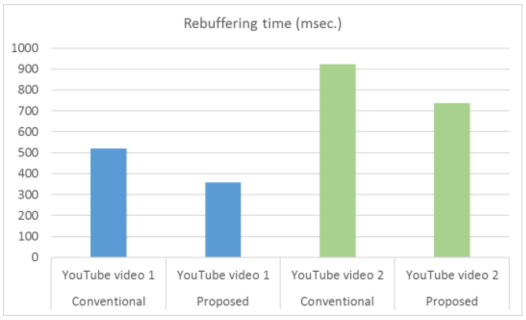

Figure 19. Mean Opinion Score.

7 Demi

In order to achieve integration of VNFs with wide area network (WAN) connectivity services, this experiment called “Dynamic WAN interconnection for multiple NFVI-PoP locations” (DEMI) deployed a WAN Infrastructure Connectivity Manager (WICM) on SoftFIRE. WICM was particularly targeted at integrating transit VNFs, which account for a large share of the VNFs that are candidate to be offered “as-a-Service”.

7.2 WAN Infrastructure Connectivity Manager (WICM)

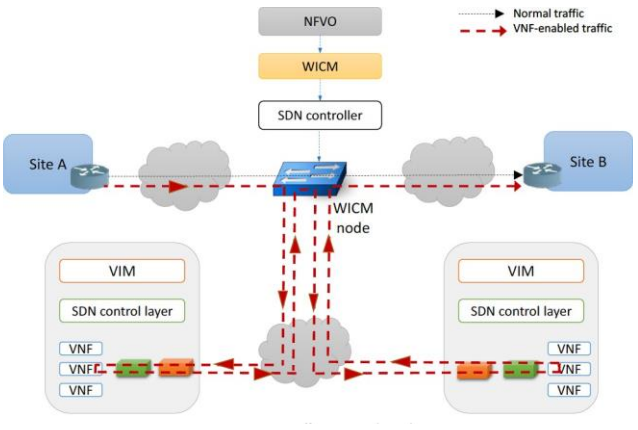

Figure 20. Data traffic steering through WICM.

WICM manages the WAN resources in a similar way that the Virtualized Infrastructure Manager (VIM) manages resources within an NFVI Point of Presence (PoP) network. It steers data traffic whenever a new service is instantiated across multiple NFVIs. When an NFV services is selected, the request is forwarded to the NFVO. Through the NFVO, service mapping is performed in order to select the appropriate NFVI PoP, and connectivity from the WICM can be requested. To achieve this type of connectivity, WICM steers traffic based on VNF Forwarding Graph Descriptor (VNFFGD) rules and network WAN connectivity resources.

As shown in Figure 20, the NFVO requests the WICM to steer the data traffic so that packets pass through a pre-defined sequence of VNFs, located in NFVI-PoPs **A** and **B** (Sites A and B, in the figure). It should be mentioned that the complexity of the process performed in the

WICM depends merely on (i) the physical and logical locations of VNFs, (ii) the common network abstractions used, and (iii) the available network connectivity resources.

The architecture of DEMI, as shown in Figure 21, enables a clean separation between the role of NFVO (i.e., resource management, service mapping, and NFVI-PoP selection, etc.) and the role of WICM (end-to-end network connectivity establishment, NFVI/WAN integration).

WICM API: Provides a REST interface through which the NFVO can request connectivity services,

WICM Traffic Redirection Services: Retrieves the networking graph, translates them correctly into traffic steering rules that are to be sent to the controller.

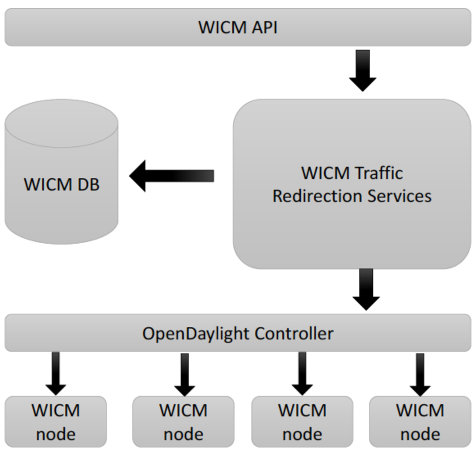

Figure 21. WICM architecture.

WICM Database: The database consists of two tables: (1) *Connectivity*, which keeps the location where the information about the underlying infrastructure has to be declared and pre-configured. (2) *UserClient*, which contains information about the source of the data traffic and its final destination, i.e. the target.

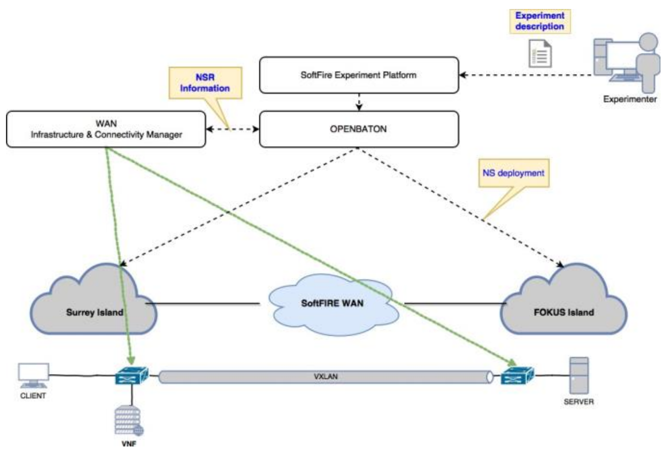

Figure 22. Deployment of DEMI on SoftFIRE.

The WICM module is composed of a set of functional components:

- *Server:* A UDP sender. Sends lowercase letters to the Client,
- *Client:* A UDP receiver that receives the text from the Server,
- *CapVNF:* The VNF that capitalizes the text intercepting the client-server connection,
- *Two OVS instances* that are controlled by WICM to enforce the network service embedding between the Client and the Server.

7.3 Deployment on SoftFIRE

The WICM in this particular deployment was located on an ORION server that had connectivity with the rest of the components located on the SoftFIRE testbed.

DEMI based on connectivity information available to the WICM in relation to the network attachment points (switch ports, IPs, mac addresses etc), the quantitative characteristics of the provisioned network links (e.g. latency, capacity etc) and the required end-to-end QoS characteristics, was able to create virtual networks and configure the proper rules at the SDN controller (i.e. OpenDaylight [37]) in order to steer traffic between two end-points through multiple NFVI-PoPs.

8 Enforce

This experiment from University of Thessaly, which was called the IntelligEnt MoNitoring oF NetwORking ServiCEs (ENFORCE), was themed on virtual setup boxes (vSTB). The experiment involved traffic monitoring and forecasting, to drive VNF scale-in and scale-out operations based on the monitoring outcomes. The experiment provided a benchmarking tool and an extension to the SoftFIRE framework that monitors virtualized resources and services defined for realizing Setup Box (STB) functionalities, called vSTBs. In particular, two parts of a vSTB, i.e. the vPVR (virtual Personal Video Recorder) and the aggregation of video streams defined in multiple formats were considered.

The tool simulated iTV service demand, and tested the performance of the platform. The monitoring mechanism used by the tool was the basis for configuration or re-configuration flags to support efficient scale-in scale-out orchestration decisions. When vSTB traffic was predicted to be below a threshold, a scale-in alert was fired. Similarly, when the vSTB traffic was predicted to be above a higher threshold, a scale-out alert was triggered.

Tests on the SoftFIRE platform were performed, adopting multiple data video sources acquired by various URIs offering different TV services as well as synthetic traces. Results on the three performance metrics, i.e. (i) service start latency, (ii) service availability (percent of time when the requested service was available to users), and (iii) fulfillment of storage demand from the application (on-demand request for storage space availability) are shown in Table 2.

Table 2. Results of the ENFORCE experiment.

OpenStack VM Flavour	Service Availability (%)	Latency (ms)	Storage demand fulfillment (%)
Small	96.89	150.62	86.41
Medium	92.55	46.80	95.68

The latency of invoking iTV services was recorded just after the request for invocation of the offered services. It was observed that adopting a *medium* VM flavour reduces the latency due to increased virtual resources.

The storage demand fulfillment was below what was targeted at initially, but close to the pre-defined goal. It was observed that OpenStack *small* flavour can support 1000 users but not 2000 at the same time. Finally, storage demand fulfillment depends on the amount of the available storage in the VM at the time of request. It was observed that the higher the storage resources are, the higher the rate of fulfillment becomes. The average service availability was found to be above the pre-defined goal.

The results show that the proposed monitoring mechanism can help scale-in or scale-out actions to upgrade or downgrade the resources involved in the virtualized environment, by means of modifying the VM flavour where iTV services were deployed. This way, the management of the available physical resources were efficiently handled to maximize performance and ensure sufficient QoS.

9 Privacity

This experiment by the company HOP Ubiquitous [38] was focused on smart city data management services, to indirectly improve privacy and security of users by means of promoting distributed SDN-based data centres that host data storage for smart city services. The experiment in general targets at providing a scalable and trustable deployment of basic services such as data storage, where the benefits of IoT and 5G can be sufficiently offered, whilst ensuring trust in IoT deployments, by taking advantage of the NFV, SDN, and edge computing technologies.

The PrivaCity experiment was based on a service offered by HOP Ubiquitous for data storage over the OMA LwM2M [39] protocol. This protocol is widely extended and supported by various platforms, such as ETSI oneM2M [40], and also Future Internet software solutions, such as FIWARE [41], thereby making it friendly for multiple SDN/NFV and 5G infrastructure providers. The experiment focused on information reporting functionalities of OMA LwM2M, i.e. *read data*, *write data*, and *subscribe to events*. The experiment in the SoftFIRE platform was based on an OMA LWM2M client-server infrastructure, and involved the following components:

- *Standalone HOMARD platform (Cloud side):* HOP Ubiquitous provides an IoT device management platform, called Homard [42]. This platform supports OMA LWM2M connections and is able to connect and share data with LwM2M clients, providing a simple and intuitive dashboard where the user can find a playground for data visualization and device management. The platform keeps track of the data amount received from the LwM2M devices connected to the platform. This web service is called Historical Data Cache (HDC), which allows data collection and statistics derivation, or application of complex data mining algorithms. This service was used as a server in this experiment.
- *Virtual Object - VO (Sensor / IoT side):* The experiment included virtual sensor objects, which are simple micro-service instances, each connected directly to the IoT OMA LwM2M physical device that they are in constant communication with. The VO offered a simple API to fetch instant information, and make asynchronous observations.

The experiment focused on data management for this IoT service consisting of multiple VOs using SDN. The objective of the experiment was to achieve and showcase scalability and user privacy, by keeping the data in the neighbourhood of data clients with edge computing. Orchestration and deployment of services at the edge is also beneficial for better Quality of Service (QoS).

- ✓ *Privacy:* Edge deployment of services effectively reduces the number of network hops, which reduces the possibility of potential vulnerability points on data paths.
- ✓ *Scalability* is envisioned to be a by-product of NFV services in 5G, which the SoftFIRE project provided with its multi-testbed platform running OpenStack. Scalability is needed for massive IoT deployments, which are expected in the Smart Cities domain. In the experiments, IoT service scalability was tested with SDN, instead of the available conventional backbone in the Internet.

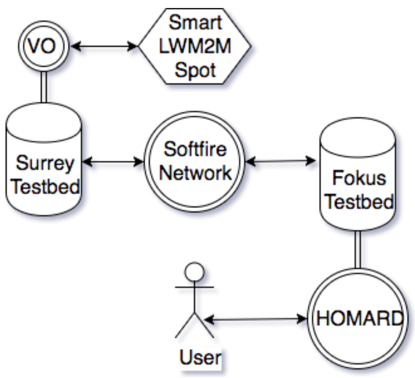

Figure 23. Service chain deployment of the Privacity experiment.

The experiment deployed IoT services in three countries: one location where the experimenter facilities is located (Spain), and two locations where a SoftFIRE testbed resides (Germany and the UK). This is illustrated in Figure 23.

The experimenters expect to exploit the findings as part of the 5G deployments during coming two years, when SDN and network slicing capabilities will be widely available.

10 Inmarsat

In this experiment, the company Inmarsat [43] provided a solution for service mobility and policy management extension in the SoftFIRE federated testbed which spans multiple sites across Europe. The solution was based on and inspired by the company's solution for global mobility of user equipment.

Having a dedicated Policy & Mobility Management mechanism (outside the NFVO space) provides:

- interoperability with different NFVO implementations,
- an entity dedicated in the NFVi architecture that is responsible for complex mobility event management,
- the ability to collect and consume data and events (synchronously or asynchronously) from various layers and sources of the infrastructure.

The experimenter first defined mobility and policy-change events. The provided framework consumed messages for mobility events, particularly for mobile equipment, which are associated with a defined policy. The company's infrastructure includes thousands of highly mobile customer user equipment that are served by different ground stations.

In the experiment, the events from a radio access network (RAN) equipment was considered to trigger a re-configuration procedure for a provisioned service running as a virtual network function (VNF) on the SoftFIRE platform. The experimenter used a policy mechanism to trigger orchestration events by interfacing with the Open Baton NFVO based on the policy assessment results. In doing so, the experiment also proved its interoperability with SoftFIRE's NFVO implementation, i.e. Open Baton, and also proved the concept of a dedicated entity in the NFV architecture that is responsible for complex mobility event managements.

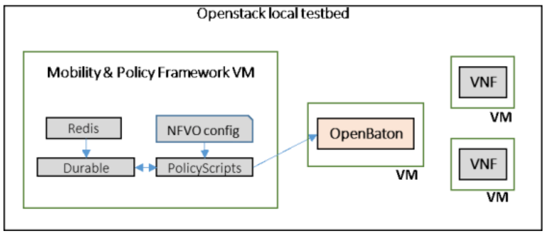

Figure 24. Inmarsat experiment's high-level setup.

The experimenter first tested their solution on a local virtualisation environment, which included OpenStack and Open Baton. The high-level setup is illustrated in Figure 24. The setup includes the mobility and policy framework running on a virtual machine, the local Open Baton, and running test VNFs. A simple firewall was selected as a test VNF, whose operation was verified by requesting access to a Linux server in the SoftFIRE testbed from an external Linux machine.

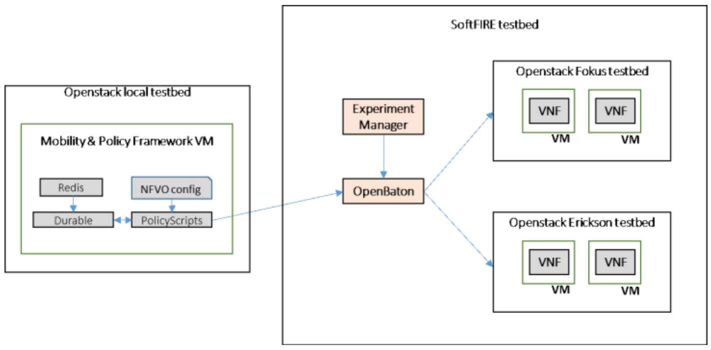

Figure 25. Test of the mobility and policy management mechanism by migration of running services.

Once the solution was verified, the test VNFs were deployed on the SoftFIRE testbeds. This is illustrated in Figure 25. The experimenter deployed the mobility and policy management solution on their local environment in a VM, which then interacted externally with the Open Baton NFVO hosted on the SoftFIRE platform. The deployment of the test VNFs was performed via the SoftFIRE Middleware [4] on the Fokus [20] component testbed of SoftFIRE. Once the deployment was validated, the mobility policy was set and then triggered by an event generation mechanism. This then demonstrated that the policy was triggered, resulting in orchestration events that migrate the service, i.e. the VNF, from one testbed to the other. The migration of the service (test VNF) was tested between the Fokus [20] and Ericsson [44] testbeds.

11 NFV-Shield

This experiment aimed at deploying, integrating, testing, and validating Intrusion Detection System (IDS) tools in NFV-based scenarios. Security in VNFs is one of the most sought after features, as NFV is expected to be adopted by various industry verticals with the flexibility and scalability it can provide. Furthermore, security in virtualisation systems is still a main concern for platform providers and customers. As such, this experiment aimed at enabling an orchestrated management of an IDS as a VNF that can be employed by other VNFs, such as Web Services, IMS (IP Multimedia Subsystem) and 3GPP's EPC (Evolved Packet Core) components. The experiment involved extension of the security functionalities supported by Open Baton.

The deployed IDS VNF, called *Shield IDS*, was evaluated for its accuracy, which was measured as a ratio of identified attacks to all performed attacks, i.e. detection ratio. Table 3 lists the types of attacks in the NFV-Shield experiment.

Table 3. Type of security attacks used in the NFV-Shield experiment.

Type of attack	Description	Employed tools
Scanning	Scanning available/open TCP and UDP ports, using a port scanner.	Nmap [45] is employed as a port scanner tool.
Vulnerability exploitation	Exploitation of vulnerabilities in web applications, such as cross-site scripting, and remote code execution, among others.	Web application as target, with included vulnerabilities. The OWASP ZAP [46] application is employed in attack mode.
System penetration (Brute-force ssh connections)	Brute-force attack from existent users (e.g. ubuntu, root), though the *ssh* service.	Use patator [47] or Ncrack [48] tools available in Kali Linux [49].

11.2 Attack Scenarios

11.2.1. *Scenario 1:* Base attack

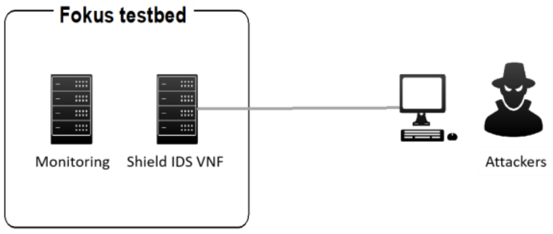

Figure 26. The base attack scenario in the NFV Shield experiment

In this scenario, a Shield-IDS VNF was deployed on SoftFIRE, and a Kali Linux attacker performs security attacks. A monitoring node gathered values on CPU usage, I/O statistics, etc. The scenario was aimed at configuring the IDS to support detection and protection from the envisioned attacks, as well as to identify the performance and accuracy of the Shield IDS.

11.2.2. Scenario 2: Tap as a service

This is a port mirroring scenario in which the traffic received on port 80 of the web server instance was forwarded to the Shield IDS instance, which can then provide active protection. From a deployment perspective, this scenario also enables the capability to dynamically configure port mirroring. For instance, port mirroring can be performed according to instance lifecycle (i.e. when instances are deployed, provisioned or deleted).

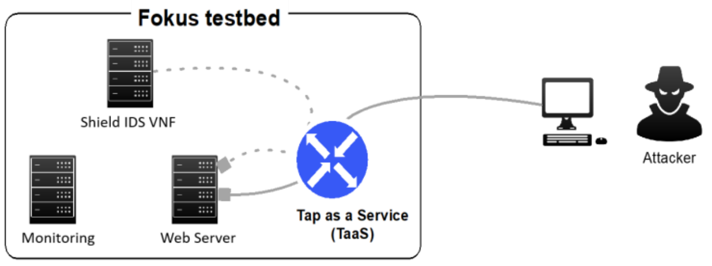

Figure 27. Port mirroring scenario in the NFV-Shield experiment.

11.2.3. Scenario 3: Port forwarding at target VNF

The Distributed IDS analysis required Shield IDS agents to be deployed in each instance that needs to be secured. The Shield IDS agents performed analysis of received information, according to the rules that can be configured for the service(s) running in the VNF instance to be secured. The results were then sent to the Shield IDS instance for aggregation and for integrated security management. Despite the granular configuration of this approach, where rules can be customised according to the service, this approach has the disadvantage of having a high performance impact, i.e. CPU, memory, and storage resources are required for real-time analysis of traffic.

On the other hand, the Port Forwarding approach mimics the Tap as a Service scenario in Section 11.2.2, with the difference that forwarding was performed at instance level and not at infrastructure level. For instance, in Linux based instances, iptables [50] can be configured to "duplicate" the received traffic and to send it to the Shield IDS. An immediate advantage of this approach is the reduced impact on the protected instances in terms of resource usage (i.e. storage resource is not required). Nonetheless, unlike the Tap as a Service scenario, port forwarding at instance level is not very scalable and cannot flexibly manage security policies. For instance, all instances that need to be protected needed to be modified.

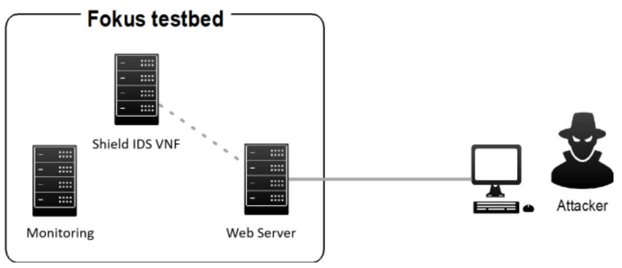

Figure 28. Port forwarding at target VNF instance.

12 5GNaaS

3GPP 5G network system architecture [51] has defined the User Plane Function (UPF) as a flexible software component that can be readily deployed and programmed from the control plane of the mobile packet core network. The 5G Network as a Service (5GNaaS) experiment was focused on evaluating the deployment of the UPF at the network edge, to provide a local breakout point north of off-the-shelf LTE eNodeB equipment. This edge UPF component was deployed on an SDN switch between an eNodeB and the virtualised packet core running on the SoftFIRE platform.

The experiment ran two separate VMs, each including a different type of network slice. The first VM included both control plane and user plane parts of an LTE network. This slice had a dedicated Femto cell equipment connected to it on a dedicated PLMN ID. The second slice had a separate LTE network core on another PLMN, including only its control plane; as the user plane of this second LTE network was deployed on a physical server on the path in between a second Femto cell and this second VM. This second LTE network was deployed to demonstrate the User Plane Function (UPF) at the network edge, of the newly introduced 5G system architecture. The UPF in this experiment consisted of the entire SGW and PGW, retaining the UP and CP parts as in LTE, yet was deployed at the network edge as a proof-of-concept demonstration of UPF deployment to support MEC applications.

Figure 29. 5GNaaS SDN hardware switch board.

In addition to showcasing deployment of custom mobile network core slices, either as CP-only or CP and UP combined, the experiment also included Software Defined Networking (SDN) functionality, identifying traffic flows based on PLMN IDs, which was needed to program an Open Virtual Switch (OVS) [22] that ran at the same server where the edge UPF was located, so that traffic redirection could be performed to this edge UPF.

Throughput and latency performance on the user plane was observed and compared between two scenarios: *(i)* Edge UPF on edge SDN switch and virtual core on the SoftFIRE platform, and *(ii)* virtual UPF and virtual core both on the SoftFIRE platform. The separate virtual cores running for each scenario were separate network slices running in parallel. Performance tests were carried out on Android mobiles towards an Internet server.

13 Aerial Insights

Due to the growing demand for connectivity across different types of networks and equipment, which is particularly expected when 5G mobile networks are rolled out, enabler technologies, such as drones, have now been equipped with communication equipment that would enable their connectivity to existing terrestrial networks. In view of this, the company Aerial Insights [52], which is a data analytics company with focus in the drone industry, designed and deployed an experiment on the SoftFIRE platform to test their visual data processing algorithms in a virtualisation environment, and to leverage industrial standards, such as TOSCA and ETSI NFV MANO [53], in use of virtualised drone data processing applications. The experiment involved processing raw images collected by drones dynamically. Specifically, the experiment tested image processing algorithms that build aerial maps from raw images and provided intelligent map streaming based on dynamic network conditions, whilst ensuring uninterrupted user experience.

The experiment on the SoftFIRE platform focused on near real-time map previews, with an aim to provide uninterrupted pre-computed map visualisation, while maintaining perceived QoS. Towards this, the experimenter first performed an accurate configuration of their software based on testbed conditions so that an automatic provisioning of (i) computing instances, (ii) the connectivity between the instances, and (iii) an accurate model of the customers operating in the field, are achieved. Then, multiple users that consume the map data are simulated in the testbed environment. These two steps enabled real-time test of the experimenter's algorithm.

Two types of users were modelled: (i) third party developers accessing the public APIs , and (ii) human users that connect to the system using a browser. Mimicking the user behaviour involved downloading two kinds of images: *map tiles,* which are similar to the ones that are consumed in online services like Google Maps, and *high resolution maps,* which are useful for batch processing on the customer premises.

The experiment had a modular architecture in which processing, persistent storage, and APIs and user interfaces were mapped to individual components. The processing components were based on a master-slave scheme in which workers can be added on demand. Most of the components were scalable by design, i.e. new instances of a given service can be replicated, booted, and can provide machine level parallel execution of individual requests. Most requests were atomic and stateless by design (except for the permanent storage used to "remember" the results), so they can be processed by containers. Several container instances of the frontend and public APIs could run in parallel so the load could be shared. They were hidden behind an nginx [54] server that acted as a reverse proxy and load balancer.

Each instance was found to be capable of responding to requests from several hundreds of users, which in part was due to the fact that the patterns of usage were more intensive on the client side (browser downloading and rendering web pages) than on the Internet-facing servers (software accessing the APIs, composing web pages and delivering them to the customers). Those servers were heavily optimized to use caching. In the case of public APIs, the experimenter demonstrated that each server could sustain good performance for several thousand users, and peaks of traffic associated to more than 8000 simultaneous users. The experimenter performed an extensive set of tests, each time simulating a different number of users on the platform, which included functionality, performance, and network impairment tests.

This experiment exposed the benefits of virtualization for network and computing resources. The experiment execution was consistent and reproducible, and the experimenter could simulate different loads, synthetic customer locations, and network capabilities.

14 Concluding Remarks

Various experiments were executed on the federated virtualisation testbed provided by SoftFIRE. This white paper presents the experiments that deployed NFV and SDN solutions on the SoftFIRE platform during its 2nd Wave of Experiments.

The newly developed SoftFIRE Middleware helped experimenters define different types of experiments, each provided by the platform as virtualised resources. Thanks to its Middleware, the project supported many more experiments during its 2nd Wave. The Project’s main achievement during this experimentation wave is hence a new and modular middleware, enabling services for various 5G applications. SMEs and academic organisations benefitted from the SoftFIRE platform, by testing their solutions in a virtualised test environment. The white paper presents all these applications and solutions, which are examples of near-future virtualised services that the industry and the technology market will benefit from.

Bibliography

[1] Second Wave of Experiments on the SoftFIRE platform, https://www.softfire.eu/open-calls/second-open-call/

[2] EU SoftFIRE project, https://www.softfire.eu/

[3] First Wave of Experiments on the SoftFIRE platform, https://www.softfire.eu/open-calls/first-open-call/

[4] SoftFIRE Middleware, http://docs.softfire.eu/softfire-middleware/

[5] SoftFIRE SDN Manager, http://docs.softfire.eu/sdn-manager/

[6] SoftFIRE NFV Manager, http://docs.softfire.eu/nfv-manager/

[7] SoftFIRE Security Manager, http://docs.softfire.eu/security-manager/

[8] SoftFIRE Physical Device Manager, http://docs.softfire.eu/pd-manager/

[9] SoftFIRE Monitoring Manager, http://docs.softfire.eu/monitoring-manager/

[10] "Network Function Virtualisation: State-of-the-Art and Research Challenges", Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, Raouf Boutaba, *IEEE Communications Surveys & Tutorials*, vol 18, no 1, 236-262, September, 2015,

[11] "Network Functions Virtualisation— Introductory White Paper", ETSI, 22 October 2012, retrieved 20 June 2013.

[12] "Software-Defined Networking: A Comprehensive Survey", Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, *Proceedings of the IEEE*, vol 103, no 1, pp 14-76, January, 2015,

[13] S. Ahvar, H. Pann Phyu, S. M. Buddhacharya, E. Ahvar, N. Crespi, R. Glitho, "CCVP: Cost-efficient Centrality-based VNF Placement and Chaining algorithm for network service provisioning", IEEE NetSoft, 2017.

[14] Institut Mines Telecom, Telecom SudParis, https://www.imt.fr/en/

[15] Azure cost modelling, https://azure.microsoft.com/en-gb/pricing/

[16] OpenStack open source cloud computing software, https://www.openstack.org/

[17] SoftFIRE Experiment Manager, http://docs.softfire.eu/experiment-manager/

[18] Zabbix, http://www.zabbix.com/

[19] netfilter, https://www.netfilter.org/

[20] Fraunhofer Fokus, FUSECO Playground, https://www.fokus.fraunhofer.de/go/en/fokus_testbeds/fuseco_playground

[21] 5G Innovation Centre, University of Surrey, http://www.surrey.ac.uk/5gic

[22] Open Virtual Switch (OVS), http://openvswitch.org/

[23] Eight Bells Research Ltd, http://www.8bellsresearch.com/

[24] Service function chaining architecture, IETF, [Online] available at: https://tools.ietf.org/html/rfc7665

[25] Service function chaining, OPNFV, [Online], available at: https://wiki.opnfv.org/display/sfc/Service+Function+Chaining+Home

[26] nDPI, Open and Extensible LGPLv3 Deep Packet Inspection Library, https://www.ntop.org/products/deep-packet-inspection/ndpi/
[27] DozeroTech, http://www.dozerotech.com/
[28] Wireless Gigabit Alliance (Wi-Gig), Wi-Fi Alliance, https://www.wi-fi.org/
[29] GamingAnywhere, an open-source cloud gaming platforms, http://gaminganywhere.org/
[30] Open5GCore, https://www.open5gcore.org/
[31] Modio Computing, https://modio.io/
[32] OpenBaton, https://openbaton.github.io/
[33] MODIO experiment video, https://youtu.be/p47y6nl9sv4
[34] StreamOWL, http://www.streamowl.com/
[35] ITU Recommendation 1208, https://www.itu.int/itu-t/recommendations/rec.aspx?rec=13158
[36] ITU Recommendation 1201, https://www.itu.int/rec/T-REC-P.1201/en
[37] OpenDaylight, https://www.opendaylight.org/
[38] Hop Ubiquitous, http://www.hopu.eu/
[39] Lightweight M2M (LwM2M), https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/
[40] ETSI OneM2M, http://www.onem2m.org/
[41] FIWARE, https://www.fiware.org/
[42] Homard platforms, https://homard.hopu.eu/
[43] Inmarsat, https://www.inmarsat.com/
[44] Ericsson RMED CloudLab, https://www.ericsson.com/portfolio/services-and-solutions/learning-services/education-centers/rmed
[45] Network Mapper (NMap), a free and open source (license) utility for network discovery and security auditing, https://tools.kali.org/information-gathering/nmap
[46] OWASP Zed Attack Proxy (ZAP), https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
[47] patator, a multi-purpose brute-force attack tool, https://tools.kali.org/password-attacks/patator
[48] Ncrack, a high-speed authentication cracking tool, https://tools.kali.org/password-attacks/ncrack
[49] Kali Linux penetration testing tool, https://www.kali.org/
[50] iptables, https://www.netfilter.org/projects/iptables/index.html
[51] TS 23.501, System Architecture for the 5G System, 3GPP.
[52] Aerial Insights, https://www.aerialai.com/
[53] ETSI MANO specification, http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
[54] nginx, https://www.nginx.com/

List of Acronyms and Abbreviations

Acronym	Meaning
3GPP	Third Generation Partnership Project
5G	Fifth Generation Mobile Network
ADS	Assembly Data System
API	Application Programming Interface
CP	Control Plane
CPU	Central Processing Unit
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
EU	European Union
FTP	File Transfer Protocol
FW	Firewall
HDC	Historical Data Cache
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IDS	Intrusion Detection System
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IP	Internet Protocol
KPI	Key Performance Indicator
LTE	Long Term Evolution
LwM2M	Leightweight Machine-to-Machine
MANO	Management and Orchestration
MOS	Mean Opinion Score
NAT	Network Address Translation
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure

NFVO	Network Function Virtualisation Orchestrator
NS	Network Service
ODL	OpenDaylight
OMA	Open Mobile Alliance
OS	Operating System
OTT	Over-the-top
OVS	Open Virtual Switch
PDN	Packet Data Network
PGW	PDN Gateway
PLMN	Public Land Mobile Network
PoP	Point of Presence
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAN	Radio Access Network
RNN	Recurrent Neural Networks
RTT	Round Trip Time
SDN	Software Defined Network
SFC	Service Function Chaining
SGW	Serving Gateway
SLA	Service Layer Agreement
SSIM	Structural SIMilarity
SSL	Secure Sockets Layer
STB	Setup Box
TCP	Transport Control Protocol
ToS	Type of Services
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDP	User Datagram Protocol
UP	User Plane
UPF	User Plane Function
VIM	Virtual Infrastructure Manager

SoftFIRE

VM	Virtual Machine
VNF	Virtual Network Function
VO	Virtual Object
VR	Virtual Reality
WAN	Wide Area Network

SoftFIRE has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 687860.