This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No 856691.
Revision history (including peer reviewing & quality control)

<table>
<thead>
<tr>
<th>Version</th>
<th>Issue Date</th>
<th>% Complete</th>
<th>Changes</th>
<th>Contributor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>18/01/2022</td>
<td>0%</td>
<td>TOC proposal</td>
<td>Ioannis Markopoulos (NOVA), All</td>
</tr>
<tr>
<td>V0.2</td>
<td>04/02/2022</td>
<td>0%</td>
<td>TOC finalisation</td>
<td>Konstantinos Oikonomou (NOVA), All</td>
</tr>
<tr>
<td>V0.3</td>
<td>25/03/2022</td>
<td>20%</td>
<td>Initial input till 25/03/2022</td>
<td>All</td>
</tr>
<tr>
<td>V0.4</td>
<td>25/04/2022</td>
<td>80%</td>
<td>Final input till 25/04/2022</td>
<td>All</td>
</tr>
<tr>
<td>V0.5</td>
<td>08/05/2022</td>
<td>83%</td>
<td>Adjustments before internal review</td>
<td>All</td>
</tr>
<tr>
<td>V0.6</td>
<td>12/05/2022</td>
<td>85%</td>
<td>Ready for internal review</td>
<td>Ran Rahav (LIVEU)</td>
</tr>
<tr>
<td>V0.7</td>
<td>27/05/2022</td>
<td>95%</td>
<td>Internal review comments addressing</td>
<td>Konstantinos Oikonomou (NOVA)</td>
</tr>
<tr>
<td>V0.8</td>
<td>29/05/2022</td>
<td>96%</td>
<td>Format changes</td>
<td>All</td>
</tr>
<tr>
<td>V0.9</td>
<td>30/05/2022</td>
<td>98%</td>
<td>Final adjustments</td>
<td>Konstantinos Oikonomou (NOVA)</td>
</tr>
<tr>
<td>V1.0</td>
<td>31/05/2022</td>
<td>100%</td>
<td>Ready for submission</td>
<td>All</td>
</tr>
</tbody>
</table>

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the 5G Solutions consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the 5G Solutions Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the 5G Solutions Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© 5G Solutions Consortium, 2019-2022. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

1 According to 5G Solutions Quality Assurance Process:
1 month after the Task started: Deliverable outline and structure
3 months before Deliverable’s Due Date: 50% should be complete
2 months before Deliverable’s Due Date: 80% should be complete
1 months before Deliverable’s Due Date: close to 100%. At this stage it sent for review by 2 peer reviewers
Submission month: All required changes by Peer Reviewers have been applied, and goes for final review by the Quality Manager, before submitted
Table of Contents

1 Executive Summary ...........................................................................................................................................9
2 Introduction.....................................................................................................................................................10
  2.1 Mapping Projects’ Outputs .......................................................................................................................11
  2.2 Deliverable Overview and Report Structure .............................................................................................12
3 LL4 Cycle 2 trials results, lessons learned & recommendations for Cycle 3 trials ...........................................13
  3.1 UC4.1.........................................................................................................................................................13
    3.1.1 UC test objective, design and deployment .........................................................................................13
    3.1.2 UC Architecture................................................................................................................................13
    3.1.3 Trials description and measured KPIs .................................................................................................17
    3.1.4 Lessons learned from the Deployment phase ....................................................................................23
    3.1.5 Results comparison with Cycle 1 .........................................................................................................23
    3.1.6 Results comparison with 5G targets ....................................................................................................23
    3.1.7 Planning for Cycle 3 trials...................................................................................................................23
  3.2 UC4.2.........................................................................................................................................................23
    3.2.1 UC test objective, design and deployment .........................................................................................23
    3.2.2 Trials description and measured KPIs .................................................................................................26
    3.2.3 Lessons learned from the Deployment phase ....................................................................................28
    3.2.4 Results comparison with 5G targets ....................................................................................................28
    3.2.5 Planning for Cycle 3 trials...................................................................................................................28
  3.3 UC4.3.........................................................................................................................................................28
    3.3.1 UC test objective, design and deployment .........................................................................................28
    3.3.2 UC Architecture................................................................................................................................29
    3.3.3 Trials description and measured KPIs .................................................................................................30
    3.3.4 Lessons learned from the Deployment phase ....................................................................................31
    3.3.5 Results comparison with 5G targets ....................................................................................................32
    3.3.6 Planning for Cycle 3 trials...................................................................................................................33
  3.4 UC4.4.........................................................................................................................................................33
    3.4.1 UC test objective, design and deployment .........................................................................................33
    3.4.2 UC Architecture................................................................................................................................34
    3.4.3 Trials description and measured KPIs .................................................................................................36
    3.4.4 Lessons learned from the Deployment phase ....................................................................................37
    3.4.5 Results comparison with Cycle 1 .........................................................................................................38
    3.4.6 Results comparison with 5G targets ....................................................................................................38
    3.4.7 Planning for Cycle 3 trials...................................................................................................................39
  3.5 UC4.5.........................................................................................................................................................39
    3.5.1 UC test objective, design and deployment .........................................................................................39
    3.5.2 UC Architecture................................................................................................................................40
    3.5.3 Trials description and Measured KPIs .................................................................................................40
    3.5.4 Lessons learned from the Deployment Phase .....................................................................................40
    3.5.5 Results comparison with 5G targets ....................................................................................................41
    3.5.6 Planning for Cycle 3 trials...................................................................................................................41
  3.6 UC4.6.........................................................................................................................................................41
    3.6.1 UC test objective, design and deployment .........................................................................................41
    3.6.2 UC Architecture................................................................................................................................41
    3.6.3 Trials description and Measured KPIs .................................................................................................43
    3.6.4 Lessons learned from the Deployment Phase .....................................................................................44
    3.6.5 Results comparison with Cycle 1 .........................................................................................................44
    3.6.6 Results comparison with 5G targets ....................................................................................................45
3.6.7 Planning for Cycle 3 trials........................................................................................................ 45
4 Consolidated feedback from Cycle 2 deployment phase and lessons learned – Recommendations for Cycle 3 trials and the testbed ................................................................. 47
5 Conclusions and Next Actions ........................................................................................................... 48
6 Annex ................................................................................................................................................. 49

6.1 External In-Lab testing for UC4.4 .................................................................................................. 49
6.1.1 Leveraging virtual infrastructure elasticity ................................................................................. 49
6.1.2 Experimental Scenarios ............................................................................................................ 50
6.1.3 Remarks ..................................................................................................................................... 52
6.2 External CCTC In-Lab testing for UC4.5 ...................................................................................... 52
6.2.1 Inter-slicing algorithms ............................................................................................................ 52
6.2.2 Bonding resources across different network domains ............................................................ 52
6.2.3 Discussion .................................................................................................................................. 53
6.2.4 Scenario 1: Bonding resources based on service-level metrics ................................................... 53
6.2.5 Scenario 2 proposal for Cycle 3: Condition service placement (or migrate) to minimize inter-slice interference .................................................................................................................. 60

List of Figures

Figure 1: 5G-SOLUTIONS trials roadmap ............................................................................................... 10
Figure 2: UC4.1 Architecture Cycle 2 .................................................................................................. 14
Figure 3: Infrastructure layout for the evaluation of ODS scenario .................................................................. 15
Figure 4: Implementation of ML-based caching in UoP infrastructure .................................................. 16
Figure 5: In lab: Caching Delay. no_cache refers to no Caching strategy. (Normalized to no cache at all) ........ 19
Figure 6: In lab: Offloading Ratio (F) .................................................................................................... 19
Figure 7: UoP: Caching Delay. no_cache refers to no Caching strategy. (Normalized to no cache at all) ........ 20
Figure 8: UoP: Offloading Ratio (F) .................................................................................................... 20
Figure 9: Performance comparison between ML-Based caching and LFUC .................................................. 21
Figure 10: Effect of a sudden change in popularity pattern ..................................................................... 22
Figure 11: UC4.2 deployment setup in 5G-VINNI, PatrasUC Architecture .................................................. 24
Figure 12: High level architecture of UC4,2 ........................................................................................... 25
Figure 13: UC4.3 Flow diagram ............................................................................................................ 29
Figure 14: UC4.3 Achievable UL rate without traffic ............................................................................ 31
Figure 15: UC4.3 Achievable UL rate with traffic ............................................................................... 32
Figure 16: Real World complex Production: Epic Mountain Bike, multi-source, remote areas, multiple feeds... 34
Figure 17: UC4.4 architecture ............................................................................................................... 35
Figure 18: UC4.4 Information flow ......................................................................................................... 36
Figure 19: LU800 unit achievable rate and latency ............................................................................... 38
Figure 27: uc4.5 Architecture .................................................................................................................. 40
Figure 28: UC4.6 Information flow ......................................................................................................... 42
Figure 29: UC4.6 Architecture ........................................................................................................... 42
Figure 30: UC4.6 achievable rate and latency ...................................................................................... 44
List of Tables

Table 1: Use Cases selected for Cycle 2 trials.............................................................. 10
Table 2: Adherence to 5G-SOLUTIONS GA Deliverable & Tasks Descriptions.................. 11
Table 3 – UC4.1 Cycle 2 Network KPIs ..................................................................... 17
Table 4: In lab: Summary of Results ........................................................................ 20
Table 5: UC4.1 Cycle 2 Vcache KPIs ........................................................................ 22
Table 6: UC 4.2 Technical network performance KPIs ................................................. 26
Table 7: UC 4.2 Network Service Management KPIs .................................................. 27
Table 8: UC 4.2 QoS Monitoring and Predictions (QMP) KPIs ..................................... 27
Table 9: UC4.3 Network Configuration Table ............................................................. 29
Table 10: uc4.3 Test cases .......................................................................................... 30
Table 11: UC4.3 KPIs .................................................................................................. 30
Table 12: UC4.3 results comparison with 5G targets ...................................................... 32
Table 13: UC4.3 Network Configuration Table ............................................................. 36
Table 14: UC4.3 test cases .......................................................................................... 36
Table 15: UC4.3 KPIs .................................................................................................. 37
Table 16: UC4.4 results comparison with 5G targets ...................................................... 38
Table 17: UC4.4 Planned tests ....................................................................................... 39
Table 18: UC4.6 Network configuration table ................................................................. 42
Table 19: UC4.6 tests ................................................................................................... 43
Table 20: UC4.6 KPIs .................................................................................................. 43
Table 21: UC4.6 results comparison with 5G targets ...................................................... 45
Table 22: UC4.6 Planned tests ....................................................................................... 45
# Glossary of terms and abbreviations used

<table>
<thead>
<tr>
<th>Abbreviation / Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>5G-PPP</td>
<td>5G Infrastructure Public Private Partnership</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BSS</td>
<td>Business Support System</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Delivery Network</td>
</tr>
<tr>
<td>CDSO</td>
<td>Cross Domain Service Orchestrator</td>
</tr>
<tr>
<td>CISI</td>
<td>Container Infrastructure System Instance</td>
</tr>
<tr>
<td>CPE</td>
<td>Customer Premises Equipment</td>
</tr>
<tr>
<td>DL</td>
<td>Downlink</td>
</tr>
<tr>
<td>E2E</td>
<td>End-to-end</td>
</tr>
<tr>
<td>IMT-2020</td>
<td>International Mobile Telecommunications-202</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>LL</td>
<td>Living Lab</td>
</tr>
<tr>
<td>Mbps</td>
<td>Mega bit per second</td>
</tr>
<tr>
<td>NS</td>
<td>Network Service</td>
</tr>
<tr>
<td>NSSI</td>
<td>Network Slice Subnet Instances</td>
</tr>
<tr>
<td>OSS</td>
<td>Operations Support System</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>REMI</td>
<td>Remote Integration</td>
</tr>
<tr>
<td>SDI</td>
<td>Serial digital interface</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>TD</td>
<td>Technological Domains</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UC</td>
<td>Use Case</td>
</tr>
<tr>
<td>UL</td>
<td>Uplink</td>
</tr>
<tr>
<td>VNF</td>
<td>Virtual Network Function</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual private network</td>
</tr>
<tr>
<td>VS</td>
<td>Visualization System</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
<tr>
<td>ZTA</td>
<td>Zero Touch Automation</td>
</tr>
</tbody>
</table>
1 Executive Summary

The purpose of this deliverable is to report the status, findings and lessons learned of the LL4 UCs during the cycle 2 trials.

Specifically, the document focuses on:

- The analysis of the cycle 2 trials performed by LL4 use cases with respect to their architecture, deployment process and measured KPIs.
- Performance evaluation of each use case, particularly from the KPI perspective to illustrate how current 5G capabilities can be leveraged for different applications in each use case.
- Results comparison with both Cycle 1 KPIs, as well as 5G targets.
- Evaluating lessons learned in order to improve planning and developments for cycle 3 thus reflecting the latest improvement of the 5G infrastructure from the application perspective.

During cycle 2 trials extensive collaboration between WP1-WP3 and WP6 was achieved. To this end, D6.3B will be used to report the collaboration through the trials and provide improvement suggestions for Cycle 3 trials.
2 Introduction

The 5G-SOLUTIONS roadmap follows the 3GPP, the IMT-2020 and the 5G-PPP implementation roadmaps according to the evolution and upgrades of the 5G EVE and 5G-VINNI ICT-17 facilities as depicted in Figure 1: 5G-SOLUTIONS trials roadmap. In LL4 the 5G-VINNI UoP testbed was used.

Following Phase 1 and 2 completion, in Phase 3 and more particular on Iteration Cycle 2 the focus was placed on including all Use Cases in Cycle 2 trials. The LL4 Use Cases that participated are presented in Table 1 and they represent 100% of the total LL4 Use Cases.

Table 1: Use Cases selected for Cycle 2 trials

<table>
<thead>
<tr>
<th>UC</th>
<th>UC Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Ultra High-Fidelity Media</td>
</tr>
<tr>
<td>4.2</td>
<td>Multi CDN selection</td>
</tr>
<tr>
<td>4.3</td>
<td>On-site Live Event Experience</td>
</tr>
<tr>
<td>4.4</td>
<td>User &amp; Machine Generated Content</td>
</tr>
<tr>
<td>4.5</td>
<td>Immersive Gaming</td>
</tr>
<tr>
<td>4.6</td>
<td>Cooperative Media Production</td>
</tr>
</tbody>
</table>
2.1 Mapping Projects’ Outputs

The purpose of this section is to map 5G-SOLUTIONS Grant Agreement commitments, both within the formal Deliverable and Task description, against the project’s respective outputs and work performed.

Table 2: Adherence to 5G-SOLUTIONS GA Deliverable & Tasks Descriptions

<table>
<thead>
<tr>
<th>Project GA Component Title</th>
<th>Project GA Component Outline</th>
<th>Respective Document Chapter(s)</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASKS</td>
<td>The purpose of this task can be briefly described from the following three aspects. First of all, the performance of each use case will be evaluated, particularly from the KPI perspective to illustrate how current 5G capabilities can be leveraged for different applications in each use case. Secondly, according to the results received from each use case in every agile-based iteration, provide requirements and suggestions to further improve both functional and non-functional capabilities of 5G facilities. Following that, each use case will be further evaluated to reflect the latest improvement of the 5G infrastructure from the application perspective. This task will also provide and establish systematic feedback loops to WP1-WP3, for continuous refinement. Results will be analysed both quantitively and quantifiably. Conclusions and recommendations will be drawn including recommendations for further trial validations. When possible, the impact of 5G deployment will be also analysed so to potentially allow operators to evaluate their 5G network deployment scope, pattern and duration. Throughout the validation testing period, knowledge and findings will be documented in deliverable D6.3 together with evaluation reporting and impact assessment for the LL, and extracting lessons learned for internal dissemination among the consortium, capacity building and external dissemination as appropriate.</td>
<td>Sections 3, 4</td>
<td>Detailed description of the LL4 UC cycle 1 trials, architecture, deployment and evaluation of results.</td>
</tr>
</tbody>
</table>

DELIVERABLE
2.2 Deliverable Overview and Report Structure

A consistent methodology is required to evaluate the results of Cycle 2 trials, catalogue the lessons learned and communicate findings with both testbeds and relevant WPs, i.e. WP1-WP3, to drive improvements implementation. To this end, Section 3 of this deliverable will provide the description of the Cycle 2 trials performed during Cycle 2 for all participating LL4 trials. Each UC Trial’s description includes its architecture and real experiment deployment, the testbed infrastructure utilized, the KPIs measured during the trial, the lessons learned from the trial, the results’ comparison with Cycle 1 trials and the 5G targets and finally the planning for the upcoming Cycle 3 trials. Section 4 provides consolidated feedback from Cycle 2 deployment phase and the lessons learned, while also providing recommendations for Cycle 3 trials and the testbed. Finally, section 5 provides the conclusions and next actions.
3 LL4 Cycle 2 trials results, lessons learned & recommendations for Cycle 3 trials

The purpose of this chapter is to provide an overview over all the UCs in cycle 2 trials, presenting their characteristics, the plan that the UC owner considered for the execution, the setup and the expected results. The overview of each UC consists of the following sections:

- Test objective, design and deployment
- UC Architecture
- Trials’ description and Measured KPIs
- Lessons learned in the Deployment phase
- Results comparison with Cycle 1 and 5G targets
- Recommendations for Cycle 3 trials

3.1 UC4.1

3.1.1 UC test objective, design and deployment

The challenge broadcasters currently face is the understanding of pros e.g. additional capabilities such as slicing, or limitations that emerging NGA networks (with focus on 5G) offer in order to take advantage and adapt technological infrastructures and business models. To this end, no matter if content mobile distribution services exist over 4G mobile networks, they cannot guarantee higher quality and eventually support lower content formats in most cases. In order to test the potential of distributing UHFM over emerging 5G networks NOVA will provide content and services to the 5G-SOLUTIONS UOP 5G-VINNI testbed. In addition, NOVA will define a set of comprehensive scenarios able to provide meaningful outcomes to analyze technological, application and business aspects. Specifically, streaming content services provided by NOVA in various formats to available 5G devices using UOP 5G-VINNI testbed.

Both scenarios that will be presented below are integrated with the Cross-Domain Service Orchestrator (CDSO) in order to easily control the configuration of the system and ensure seamless connection with the Visualization System (VS) and the UoP infrastructure. The first scenario also includes integration with the Zero Touch Automation (ZTA) component in order to test policies that ensure the quality of a stream in different connection conditions.

The second scenario of UC4.1, On-Demand Streaming (ODS) scenario, seeks to exploit the Cloud/Virtualization qualities of 5G as to provide efficient caching techniques. Such caching aims at reducing the apparent delay experienced by users requesting multimedia content by implementing a cache at the Edge of the 5G end-to-end segment. 5G enables unprecedented interactions with the communication infrastructure. Thanks to the softwarization of network and compute functions (i.e., SDN, NFV), services can now perform live operations conditioned to the current state of communication resources. This feature is particularly relevant in CDNs, which seek to increase Quality of Experience (QoE) by implementing dynamic caching techniques at different levels of a 5G telco cloud, implying resource isolation from 3er-party services (i.e., network slices). 5G makes it possible to leverage many caching strategies from Cloud technologies, like exploiting global knowledge of content popularity for intelligently and dynamically provision a limited cache.

3.1.2 UC Architecture

The first scenario architecture for Cycle 2 was based on the one from Cycle 1, maintaining the connection with the CDSO and the Visualization System. The VNF and Service orchestration, as well as the control of the Encoders is similarly done by the UoP testbed, which is also connected to the CDSO. The major changes in Cycle 2 architecture are the introduction of a database that is connected to the Visualization System and of course the Zero Touch Automation (ZTA) component. The former stores metrics and KPIs and makes them available to the
latter, in order to have the data needed for the automation. ZTA evaluates all the data from the DB and predicts changes to the quality of the stream based on a set of policies and thresholds. If it evaluates that a change is needed, it calls the respective CDSO API to notify about the new parameters and the change is propagated via the UoP testbed to the Encoders. The new metrics are again collected by the Visualization System and the DB and the loop can start again, fine-tuning the stream quality based on the implemented ZTA policies.

Figure 2: UC4.1 Architecture Cycle 2

For the ODS scenario, the architecture from Figure 2 is still valid, with the necessary replacement of the DB and ZTA with the caching mechanism. The infrastructure layout for this scenario is based on ETSI Network Functions Virtualization Management and Orchestration (NFV MANO) architecture, leveraging OSM as NFV Orchestrator and OpenStack as Virtualized Infrastructure Manager (VIM). Additionally, the vCDN service components are supported by a Container Infrastructure System (CIS), which is deployed as Platform as a Service (PaaS), as enabled by ETSI IFA 029. That is, a CIS is “deployed” on top of resources (e.g., VM) orchestrated by OSM.

Figure 3 provides a high-level overview of the infrastructure. From bottom to top, it shows that the virtual Network Functions Virtualization Infrastructure (vNFVI) is deployed on top of a hardware component (Metal). It also shows that the vCDN Client is deployed as an application or script running inside a 5G UE Emulator. The vCDN service itself runs as a set of Kubernetes Workloads, whose exposed endpoints are reachable from UE thanks to the Router (and all other network setup shown).
Inside the UoP infrastructure, the Vcache implementation consists of the following parts, as shown on Figure 4:

1. VNFD and NSD
2. A Docker container that contains the Vcache algorithm and an HTTP Flask server
3. A client (python script)

The VNF is deployed using OSM and launches the Vcache server. The initial training is done in the launch time and different popularity patterns can be imposed in the newly created VNF. The python script is set to send requests based on the defined popularity distribution and evaluate the performance of the Vcache server.
The client and Vcache server are inside a private network, which can be accessed by VPN and is managed by UoP. The client sends the request through the 5G network. Then, the Vcache server replies to the client by sending back the cache status for the requested file and updates its internal state using the discussed ML-based approach.

In cycle 2 (the current cycle), a data-driven approach has been incorporated to capture the time-location dependent pattern on content popularity. The proposed approach capitalizes on:

- Advanced machine learning (ML) models that can be trained using the content popularity data. In this work, we have used the discrete version of the soft actor-critic networks.
- Simulated popularity pattern, where content popularity is modeled with a Zipf distribution. Nonetheless, the proposed framework can be extended to real-world data once it is in the production stage.

Regarding the ML-based caching, let us assume we have a total of M contents of which C contents can be kept in a cache storage. In this project, we use a deep reinforcement learning (DRL) approach called soft actor-critic network. Formulating the Vcache as a DRL algorithm, we define the state and action space of the DRL, the logic of the Vcache server, observation, and the reward function as follows:

**State:**
- Cached content ID
- The Total number of requests for each cached content in the given period or window.

**Action space:** The DRL agent can either replace the selected cached content with the currently requested content or keep the cached contents the same.

**The logic of the Vcache server:**
- If the content is cached, return the content (or its ID), and update the state.
- If the content is not cached, then ask for the content from the server, and take an action.

**Observation:** It is a request for a file index. In this project, we assume that the observations follow the Zipf distribution. It means that at any given time the content popularity has a specific structure that is
imposed by Zipf distribution and its parameter. As an example, for the Zipf parameter of 1.25, a small percentage of the contents (5%) accounts for most of the traffic (80%).

**Reward:** It is a combination of the short-term, and long-term cache hit probability

For training the Vcache algorithm, the agent (Vcache server) receives an observation and takes an action in a way that maximizes its long-term reward, which is the cache hit ratio.

The proposed approach has several advantages:

- It is flexible and can be applied to other types of popularity patterns or real-world data traffic.
- It can learn the time-location dependent popularity pattern.
- It can adapt to a sudden change in the popularity pattern (for example inside a music concert).

### 3.1.3 Trials description and measured KPIs

For the ZTA scenario in Cycle 2, integration with ZTA was tested and initial configuration policies were explored. The aim of this Cycle was to fully integrate all relevant components and achieve their orchestration via the CDSO and the UoP infrastructure. Further ZTA policies and training techniques will be explored in Cycle 3 in order to explore their potential impacts on the measured KPIs.

Multiple end-to-end trial runs have been performed in Cycle 2, in order to improve on the measured KPIs and try to achieve the commercial targets for the UC. The following Table 3 summarizes the final trials which achieved the best results and will be the base for the next trials for Cycle 3.

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>5G Target</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Passed</td>
<td>Partially Passed</td>
<td>Not Passed</td>
</tr>
<tr>
<td>Latency</td>
<td>&lt; 20 ms</td>
<td>20 - 40 ms</td>
<td>&gt; 40 ms</td>
</tr>
<tr>
<td>Experience Rate</td>
<td>30-40 Mbit/s</td>
<td>10-30 Mbit/s</td>
<td>0-10 Mbit/s</td>
</tr>
<tr>
<td>Lost Frames</td>
<td>&lt; 20 per minute</td>
<td>20 - 40 per minute</td>
<td>&gt; 40 per minute</td>
</tr>
</tbody>
</table>

For the ODS scenario in Cycle 2, the aim was also to fully integrate all components and also test various caching policies and algorithms. In order to achieve this, a Vcache server is utilized via a deep reinforcement learning (DRL) approach called soft actor-critic network. After a content request is received, the shared Cache is checked and if the content is found, then the minimum cache delay is returned, otherwise a caching policy is enforced. Caching policies include Least Recently Used (LRU), Least Frequently Used (LFU), or Windowed-Least Recently Used (W-LRU). As their names suggest, LRU will immediately evict the least-recently used content from the shared Cache and replace it with the just-received request. LFU will do so with the least-frequently used content. Both approaches are aggressive with their eviction criteria.

More specifically, the proposed **Windowed** version of LRU (W-LRU) will wait for W cache misses before applying LRU eviction criteria with slight variations. In this case, the eviction candidate will be chosen between two subgroups: 1) contents with no hit counts at cache, or 2) pure LRU approach. That is, if there are contents with no hit counts in cache, the eviction candidate will be the one with less probability \(q_i\) of being requested (i.e., the greater content id); otherwise, the LRU content is evicted. The content replacing the evicted one is the **most-recently used** within the last W content requested. This eviction and replacement criteria modification is referred to W-LRU-SMART.

The vCDN service is composed of Frontend Servers and a Cache. The Frontend Servers expose an HTML API, which is triggered by clients for making content requests. After a request is received, the shared Cache is checked.
If the content is found, then the minimum cache delay is returned\(^2\), otherwise a cache policy is enforced. The result of such policy will then return different associated caching delays.

The current configuration is the following:

- **UE emulation.**
  - Request follow a Zipf distribution. That is, the probability \( q_i \) of a UE requesting the \( i \)-th ranked file in a collection is given by \( q_i = \frac{i^{-s}}{\sum_{j=1}^{L} j^{-s}} \); where \( s \) is the Zipf parameter.
  - \( s = 1.56 \)
  - Each request occurs at a random interval between 0.5 and 0.7 seconds.
  - Experiment duration, \( T = 60 \) seconds.
  - At time \( \frac{T}{2} \), the pool of requested content is shifted by \( \Delta_i \); that is, Clients will consider \( i^* = i + \Delta_i \) the most popular content.
  - Number of users simulated \( N = 100 \).
  - Users’ spawn rate \( s = 100 \) user per second (all users are launched simultaneously).

- **vCDN Cache configuration.**
  - Content library size \( L = 1000 \).
  - W-LRU-SMART window \( W = 5 \).

- **Metrics collection.**
  - Metrics include:
    - Caching delay (normalized to a non-caching scenario).
    - the Offloading Ratio (F), which is defined to be \( F = \frac{E[r_o]}{r} \), where \( r_o \) refers to offloaded requests (content cached at the primary edge node), while \( r \) are all requests.

Preliminary results of a Containerized service of such deployment are shown. As can be seen, W-LRU-SMART considerably outperforms the other policies at both Caching delay and Offloading Ratio (see Figure 5 and Figure 6, respectively). Table 4 provides a summary of the obtained results.

Relevantly, the WLRUS_SIMBOX bars in the figures and the last row in the table corresponds to results obtained with an end-to-end 5G segment leveraging a 5G Private Node and 5G UEs emulated using Amarisoft Callbox and SimBox, respectively. That is, the Client code runs in an IP Namespace at the SimBox, which forces each Client’s traffic via a 5G interface towards the vCDN Server.

These tests aimed at evaluating the caching technique, nevertheless more tests could include actual file transfers with varying 5G NR and Core configurations.

\(^2\) Also the hit-count for the requested content is increased in the global record, etc.
Figure 5: In lab: Caching Delay. no_cache refers to no Caching strategy. (Normalized to no cache at all)

Figure 6: In lab: Offloading Ratio (F)
Regarding performance evaluation, first we compare the performance of the ML-based caching with the traditional caching techniques such as LFU. Shown in Figure 9 is the performance comparison between ML-based caching and LFU for different values of the Zipf parameters. This graph presents the steady-state performance, i.e., both algorithms are trained for a large number of observations. Although the gap is small, ML-based caching uniformly outperforms the LFU.
Second, we stress test both techniques for a sudden change of the popularity pattern. We train both algorithms with Zipf parameters of 0.4 (30% of the contents account for 50% of the traffic). Then, the indexes of the popular contents are changed, and both techniques are evaluated with a different popularity pattern, Zipf parameters of 1.25 (5% of the content accounts for 80% of the traffic). Plotted in Figure 10 is the retraining convergence speed for ML-based and LFU caching. We can see that ML-based caching outperforms the LFU with a bigger margin and converges to its steady-state at a faster pace.
The proposed approach is implemented using the Deep Reinforcement Learning Algorithms with PyTorch library. The docker container, which contains the Vcache and its dependencies, is available publicly in.

Regarding the KPIs for the Vcache server the relevant ones are cache hit ratio, delay, storage usage, and content popularity pattern (share of contents that account for 80% of traffic). 4 scenarios based on content popularity and storage usage were defined and reported on Table 5.

Table 5: UC4.1 Cycle 2 Vcache KPIs

<table>
<thead>
<tr>
<th>KPIs</th>
<th>Delay (mean/std ms)</th>
<th>Storage (% of total)</th>
<th>Cache hit ratio</th>
<th>5G Target</th>
<th>% Of contents that account for 80% of traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>80/20</td>
<td>10%</td>
<td>0.74</td>
<td>0.8</td>
<td>5%</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>80/20</td>
<td>20%</td>
<td>0.82</td>
<td>0.8</td>
<td>5%</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>80/20</td>
<td>20%</td>
<td>0.75</td>
<td>0.8</td>
<td>10%</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>80/20</td>
<td>30%</td>
<td>0.81</td>
<td>0.8</td>
<td>10%</td>
</tr>
</tbody>
</table>

Cache hit ratio of 0.8 can be achieved in all the scenarios by adding storage and the cache hit ratio can be further improved, but the trade-off between backhaul traffic cost and storage cost in the edge should be considered.
3.1.4 Lessons learned from the Deployment phase

In Cycle 2, the main goal of the deployment was the integration of all relevant components that support the architecture for both scenarios. Through multiple end-to-end trial runs this integration was achieved and the relevant KPIs were measured. In order to improve on those KPIs and try to reach the 5G targets, additional test trials were and will be performed.

3.1.5 Results comparison with Cycle 1

Performance results and KPIs vastly outperform the respective KPIs from Cycle 1. This represents a major step towards the right direction for the UC but is an expected improvement, as both the integration of the ZTA and caching mechanism were factors that improve the network and caching performance respectively.

3.1.6 Results comparison with 5G targets

Table 3 and Table 5 summarize the resulting KPIs from Cycle 2 in comparison with the 5G targets. Most KPIs have already achieved those targets, with latency, cache hit ratio and lost frames being the characteristic ones. Further improvement will be explored for Cycle 3 for those and also for the KPIs that were considered partially acceptable for this Cycle, like jitter.

3.1.7 Planning for Cycle 3 trials

For Cycle 3 and regarding the first scenario with ZTA, a pretrained policy model will be explored in order to be compared with the current implementation of the “on-the-fly” learning model. Based on the results, the best approach will be selected, or a hybrid one can also be explored.

For the second ODS scenario, the following next steps are proposed:

1. Adding video streaming or file transfer
2. Serving multiple users
3. Lunching another VNF that acts as main server
4. Producing a demo

Furthermore, the discussed approach is only valid when the cache server and main server are part of the same business entity and so further investigation needs to be done to accommodate encrypted traffic. This is an active research topic, and several solutions such as blind-cache, cache proxy, and crypto-cache can be explored.

3.2 UC4.2

3.2.1 UC test objective, design and deployment

This use case aims in the exploitation of mechanisms for intelligent optimization and adaptation of CDN-based streaming services, by leveraging on MEC mechanisms for the evaluation of and the selection among alternative available CDN resources. CDN services, and in particular edge streaming proxies for IPTV and VoD services offered as Virtualized Network Functions (VNFs), are dynamically selected and adapted, depending on intelligent system-level monitoring and predictions of workload, resource utilization and resource requirements posed by end users’ service demand. This also includes short-term predictions of deteriorating service quality, allowing both system-level and application-level adaptations. Examples of system-level adaptations are the intra-CDN cache switching or the switching between different CDNs, while some application-level adaptations may be modifications on the video-chunk quality or on the cached content parameters.

The validation and evaluation of this use case is carried at the 5G-VINNI Patras facility, which is operated by the University of Patras (UoP). For Cycle 2, the setup uses components from the ICOM Commercial CDN, i.e., the fs|CDN™ Anywhere solution, including packaging, streaming, and caching functions. The video content is streamed from the Video Head-End & Management platform, via HLS or MPEG DASH, to the edge streaming...
servers and end users. ICOM’s CDN solution seamlessly integrates with content provided by NOVA. The CDN solution is developed as NFV artifacts on 5G network slices, using the NFV MANO tools from the 5G-VINNI Patra testbed. Three end-to-end CDN slices are deployed in the testbed in order to represent a multi-provider CDN scenario. Each slice has isolated and independent CDN management and storage network services, but they are all sharing MEC services for access, monitoring and adaptations. System-level monitoring data are collected from the CDN deployment using the KPI Visualization System. An existing anonymized data set from a commercial deployment of ICOM’s CDN solution is used for the training of prediction algorithms and the online evaluation of the QoS monitoring and prediction module. ETSI MEC mechanisms and APIs are used for interoperability between CDN services and MEC services, as well as for supplying the QoS Monitoring and Prediction (QMP) module with application level data information. ICOM Android applications is also available for the evaluation of CDN clients running on 5G devices.

The above description is illustrated in Figure 11, where three Network Slices are established on 5G-VINNI Patras facility. These slices are created by using three Access Point Names (APNs), where each of them has independent CDN management and network services. Each slice is comprised by VNFs for the ICOM central CDN Headend and Management as well as, at least, one VNF for the CDN edge cache. In Cycle 2, the VNF orchestration is done through Open Source MANO (OSM), while the NFV infrastructure in managed through OpenStack. Additionally, for the trials, free-licensed, local content is used, which is provided by NOVA. For the RAN part, just one gNB is required (as shown in Figure 11). ICOM fs|cdn™ Anywhere application is installed on 5G devices connected to the single gNB. UE assignment to a slice is done by selecting the respective APN.

The high-level architecture of UC4.2 is depicted in Figure 12. The multi CDN selection approach focuses on QoS-driven intelligent adaptation of CDN components. As CDN resources, in the context of this use case, we consider end-to-end CDN solutions of CDN providers deployed over the MNO network. Centralized CDN components, such as video head-end and management platforms, are typically deployed on the mobile core. On the contrary, caching and streaming server components can also be deployed on MEC level, acting as edge streaming proxies. Such CDN modular deployment allows for demand-based provisioning at the MEC level (i.e., instantiation of edge streaming proxies), taking under consideration number of connected users, content popularity, user mobility etc.
This approach examines intra-CDN adaptation primitives, for the dynamic assignment of end users to edge streaming proxies’ components (i.e., caches) and for the dynamic configuration of edge components (content caching triggering, caching parameterization etc.). Moreover, an additional level of adaptation is introduced with the availability of alternative CDN solutions from multiple providers, so as to select, at run-time, the CDN components for end users that offer the best performance for the streaming tasks at hand, while at the same time allowing additional layer of load balancing.

Figure 12: High level architecture of UC4.2

As shown in Figure 12, we assume the availability of solutions by different CDN Providers on the same MNO Network, which can all receive content by the same Content Origin and distribute it over their CDN solutions to end users. We also make the assumption that end-users establish business relationships only with the Content Provider, while the mobile network and CDN service details are transparent to them on a business level. Nevertheless, at each point of time, each CDN Provider serves a concrete set of end users, but based on real-time monitored or predicted QoS, the assignment of an end-user to a particular CDN provider is subject to change on run-time, both for load balancing purposes and for the assurance of streaming QoS and QoE for each end-user. This analysis is performed by QoS Monitoring and Prediction services and CDN selection and CDN edge adaptation functions at the MEC Host level.

Both intra-CDN and inter-CDN switching actuations take place using proper MEC services. In our scenario, intra-CDN adaptations include the dynamic switching between edge caches (of the same CDN Provider) that serve particular end-users, as well as the triggering of concrete content caching on a particular edge cache or the parameterization of that cache (e.g. buffer size, content Time-To-Live (TTL)). On the other hand, inter-CDN
switching include the re-assignment of some UEs to a different CDN slice. In general, CDN solutions of different CDN Providers are instantiated each in its own 5G Network Slice and adaptation primitives are realized as Network Function Virtualization (NFV) Management and Orchestration (MANO) Day-2 configurations.

3.2.2 Trials description and measured KPIs

The goal of the use case scenarios is to evaluate the end-to-end CDN functionality, the timely retrieval and analysis of monitoring metrics and the functionality of intra and inter-slice adaptations. Below is a list of the main test cases performed in Cycle 2:

- **UC4.2-SC1 - CDN NFV MANO artifacts on 5GVINNI, Patra:** This test is separated into three steps. The first one is about the preparation of the fs|cdn™ NFV MANO artifacts. These include the ICOM fs|cdn™ Anywhere NFV artifacts for the CDN components comprising the CDN slices. The second step is the onboarding of fs|cdn™ NFV MANO artifacts to the Patras Facility site. This step aims at validating and evaluating the deployment of CDN components over the 5GVINNI, Patra, facility through its orchestration layer, setting up the slice and verifying initial connectivity between all components. Additionally, it aims at verifying and evaluating the End-to-End (E2E) connectivity, by examining, for example, aspects related to data rate and latency. This also requires the use of the fs|cdn™ Anywhere mobile app running on 5G devices. Finally, this second step is repeated in order to create three CDN slices in total.

- **UC4.2-SC2 - Test end-to-end CDN functionality:** The next step is to add the Streaming Content. Once the slices are established, we tested the E2E functionality of the deployed CDNs, including the streaming of content from Origin to end-user devices through the CDN instances. The content used video content from NOVA.

- **UC4.2-SC3 - Integration with KPI Visualization System (KVS):** In this test, the KPI-VS was configured according to use case requirements. Next, the retrieval of application-level and system-level monitoring metrics was tested. The goal is to verify that the KVS successfully retrieves the monitoring data both from the CDN application and from OSM.

- **UC4.2-SC4 - Integration with QoS Monitoring & Prediction (QMP) Platform:** This is to test the integration between KPI-VS and QMP, i.e. to verify that the QMP can get the data it requires from the KPI-VS, in order to perform its function. Moreover, it is needed to check that the QoS Monitoring & Prediction component can analyze this information and produce insights, notifications and alerts. Finally, the QoS Monitoring & Prediction module was evaluated.

Through the tests of this use case, a number of KPIs of different categories were evaluated. Particularly, the KPI categories that are relevant to UC4.2 tests for Cycle 2 are the following:

- **Technical network performance KPIs:** This includes KPIs related to data rate (e.g., peak downlink throughput and minimum downlink data rate per stream), latency (e.g., maximum packet latency from UE to N6 interface), and reliability (e.g., percentage of packet-loss from UE to N6 interface). Such KPIs can be satisfied by 5G EMBB Network Slices allocated for the different CDN solutions, in combination with Multi-access Edge Computing (MEC).

- **Network Service Management (NSM) KPIs:** This includes the measurements of time frames related to different slicing phases.

The next tables present the KPIs measured in Cycle 2 activities. The network KPIs refer to one slice, but the measurements of the rest slices were similar, since the network conditions and requirements are exactly the same.
### Table 7: UC 4.2 Network Service Management KPIs

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Measure Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downlink throughput</td>
<td>&gt; 200 Mbps</td>
<td>Average: 255 Mbps Peak: 351 Mbps</td>
<td>This is the E2E link throughput. Peak and average values are measured</td>
</tr>
<tr>
<td>Downlink data rate per stream</td>
<td>&gt; 5 Mbps</td>
<td>Peak: 20 Mbps</td>
<td>This was measured with one streaming video at one UE</td>
</tr>
<tr>
<td>Latency</td>
<td>&lt;50 ms</td>
<td>Minimum: 29ms Maximum: 49ms Average: 37ms</td>
<td>Packet latency from UE to N6 interface</td>
</tr>
<tr>
<td>Reliability</td>
<td>&gt; 99.9%</td>
<td>100%</td>
<td>The packet loss from UE to N6 interface was 0, because this was measured under the conditions of the experiment, where we had 1 UE getting 2 streams. We need to stretch the system in order to notice any packet loss.</td>
</tr>
<tr>
<td>Cache hit ratio</td>
<td>&gt; 90%</td>
<td>100%</td>
<td>Since we have only 1 stream (i.e. video), the cache hit ratio is always 100% in this use case.</td>
</tr>
</tbody>
</table>

### Table 8: UC 4.2 QoS Monitoring and Predictions (QMP) KPIs

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Measured Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice instantiation time</td>
<td>&lt; 5 min</td>
<td>~ 3 min</td>
<td>Refers to the creation and activation of a single CDN slice. Note that the slices are not activated concurrently. Another notice is that Juju charms are used in configurations, posing delays in slice instantiation. If the charms were removed, the measured time would be lower.</td>
</tr>
<tr>
<td>Successful predictions</td>
<td>&gt; 95%</td>
<td>~ 98.5%</td>
<td>This is measured with training data. We need to create more traffic and measure prediction accuracy for runtime data</td>
</tr>
</tbody>
</table>
3.2.3 Lessons learned from the Deployment phase

In Cycle 2 trials, the main focus was on the preparation of the required fs|cdn™ NFV MANO artifacts and the creation of the three slices. The goal was to validate and evaluate the deployment of CDN components (VNFs for packaging, streaming & caching/prefetching) over the 5GVINNI, Patra, facility through its orchestration layer, set up the slice and verify initial connectivity between all components. This was successfully completed for all three slices and, for each one of them, the E2E connectivity was assured. For the present trial, the end user was a laptop connected to gNB through a CPE and executing the fs|cdn™ Anywhere application through a browser.

Additionally, in this cycle, the KPI Visualization System (KPI-VS) was integrated with the testbed and the QMP and it was configured according to use case requirements. It was verified that the KPI-VS successfully retrieves the monitoring data from the CDN application as well as system-level monitoring metrics from NFV-MANO (through Prometheus).

Finally, some of the data is fed to the QMP module, which analyses them. The main goal was to evaluate one of the basic functions of QMP, that is to produce insights and notifications about imminent performance degradations. For this purpose, the accuracy of the prediction function was calculated. Up to the time of the present deliverable, the accuracy was calculated using data from the existing training dataset and it was satisfyingly high (~98%).

3.2.4 Results comparison with 5G targets

Various KPIs were measured (see Section 3.2.3), such as data rate and latency experienced by an end user, and was verified that the existing network satisfies the 5G and CDN requirements. For example, the typical 5G KPIs for an eMBB slice are 20 Gbps peak datarate for downlink. This can be supported by the E2E average throughput, which was calculated to 255 Mbps. Moreover, as indicated in Table 73, all the network KPI targets were satisfied.

3.2.5 Planning for Cycle 3 trials

The next trials for this use case (Cycle 3) will explore the possibilities of maintaining the QoS for users by means of dynamic selection or adaptations of the CDN cache where the user is receiving content from, as well as the prediction of congestion and the use of edge computing resources in order to instantiate new caches on demand or to automatically move to a different CDN slice. The possibilities of network slicing to guarantee a sustainable QoS at scale will be investigated, too. Also, the integration with CDSO will be completed in Cycle 3.

Additionally, Cycle 2 trials will be further extended in order to better evaluate the QMP functions. More specifically, it is required to generate more traffic in order to push enough monitoring data in the QMP to generate predictions and calculate its accuracy with the runtime data. What is more, in future trials, we plan to use 5G mobile phones and compare the KPIs with and without applying the dynamic adaptations.

3.3 UC4.3

3.3.1 UC test objective, design and deployment

In highly populated live events, such as sports venues, music concerts, or carnivals, lots of users try to upload images, live videos and recorded clips as well as watch other participants content or background content related to the event. At the same time, special groups of users require continuous Service Level Agreements (SLAs) higher than those of the mass. The challenge for media service providers is to offer a good QoE to their clients in a dense client environment.

Currently, the providers sharing the radio access, backhaul and core network try to optimize their quality independently causing a dramatic degradation of the QoE. There are also special users such as emergency services and first responders, that require a steady QoS. 5G network must be able to provide extra resources for specific users, groups or services.

UC 4.3 involves the generation and simulation of various subscriber profiles in terms of bitrate and quality requirements and the evaluation of the corresponding QoE. In addition, this use case will be validated with load
generation, helping us in validating the business potential of this particular use case. We will also demonstrate cross network connectivity on demand through the 5G-VINNI facilities in Patra. Uplink network performance and the quality of the uplinked/streamed content will also be measured and evaluated.

3.3.2 UC Architecture

The test architecture is in the diagram below.

The smartphone camera is used via LU-Smart LiveU application with the stream.

The LU-Smart feed is controlled either by the manual operator or from remote – via the LiveU web-based control SW. Using the coordination and control information, the Visualization system knows when to start collecting the platform/network information. This enables it to correlate the LiveU application data with the network/platform data.

During the test, the LiveU log files record the application level performance data such as bandwidth, latency and error rate, for the cellular data interface. After the test ends, the LiveU app log files are manually collected, preprocessed with a special proprietary SW so to extract the relevant information, CSV files are created and manually uploaded into the visualization system portal.

The visualization system analyses and correlates the application performance and the network/platform data. It will show, per modem/IP connection, the bandwidth, latency and error over time from the application level and from the core level. Statistical analysis such as standard deviations will also be done.

In cycle 2 we extended cycle 1 basic configuration by using a “standard” 5G slice as well as “Uplink” slice - NEM UL slice.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Generic Configuration</th>
<th>UL oriented Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq</td>
<td>3.6GHz (n78)</td>
<td>3.6GHz (n78)</td>
</tr>
<tr>
<td>BW</td>
<td>50 MHz</td>
<td>50 MHz</td>
</tr>
<tr>
<td>DL</td>
<td>2 × 2 (MIMO)</td>
<td>2 × 2 (MIMO)</td>
</tr>
</tbody>
</table>

Figure 13: UC4.3 Flow diagram

Table 9: UC4.3 Network Configuration Table
 Trials description and measured KPIs

UoP provides the 5G smartphone, LiveU provided the license for LU-Smart application that is doing the live streaming. The app also includes LiveU bonding capabilities, enabling transmission split between the phone’s modem and wifi. This capability will enable testing of enhanced transmission capabilities by using more than a single network, to gain both bandwidth and stability/reliability. LiveU will also provide the video server receiving the live transmission.

UoP will provide the 5G-VINNI infrastructure and network load simulator.

AppART will provide the visualization system that collects analyses and displays the results.

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Justification</th>
<th>Measurement details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate UL per user (Kbps)</td>
<td>Generic configuration target: 4Mbps per smartphone.</td>
<td>One smartphone upload speed, Generic configuration target: overall 4Mbps, Uplink configuration target: overall 8Mbps</td>
<td>Uplink throughput measurement at application server and network</td>
</tr>
<tr>
<td></td>
<td>Uplink configuration target: 8Mbps per smartphone.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UL Latency</td>
<td>Application 1.8sec, network: &lt;200ms.</td>
<td>overall 5G (RAN+Core) UL latency and application latency</td>
<td>Uplink throughput</td>
</tr>
</tbody>
</table>

UL

<table>
<thead>
<tr>
<th>1 × 1</th>
<th>2 × 2 (MIMO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL / UL slots</td>
<td>6:3</td>
</tr>
<tr>
<td>Modulation</td>
<td>qam64 / qam64</td>
</tr>
</tbody>
</table>

Table 10: uc4.3 Test cases

<table>
<thead>
<tr>
<th>Delay</th>
<th>Configuration</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>1.4</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>1.8</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>1.4</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>1.8</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>1.4</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>1.8</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>Uplink</td>
<td>Yes</td>
</tr>
<tr>
<td>1.4</td>
<td>Uplink</td>
<td>Yes</td>
</tr>
<tr>
<td>1.8</td>
<td>Uplink</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 11: UC4.3 KPIs
<table>
<thead>
<tr>
<th></th>
<th>Application 1.4sec, network: &lt;100ms.</th>
<th>measurement at the application server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>&gt;95%</td>
<td>Broadcasters and their viewers are looking for consistency in QoS and QoE</td>
</tr>
<tr>
<td>Reliability</td>
<td>&gt;99%</td>
<td>High to satisfy user experience, but it’s an entertainment, not a critical application</td>
</tr>
</tbody>
</table>

### 3.3.4 Lessons learned from the Deployment phase

In Figure 14 and Figure 15, experimental results for UC4.3 are shown. Both configurations are examined, while the load is enabled only in Figure 15. In both Figure 14 and Figure 15, in terms of achievable UL rate, the utilization of a UL-oriented profile leads to a consistent, reliable and high performance. The existence of introduced load, regarding the UL-oriented configuration, leads to an extremely small instability as observed in Figure 15. The monitoring results are reported by the LU-Smart application. Generic configuration leads to a performance degradation, around to 3 Mbps, and to more frequent fluctuations during the streaming process. The impact of the load generation is not so observable in the case of generic configuration.

![Figure 14: UC4.3 Achievable UL rate without traffic](image-url)
3.3.5 Results comparison with 5G targets

Table 12: UC4.3 results comparison with 5G targets

<table>
<thead>
<tr>
<th>Tech KPI</th>
<th>Trial KPI Targets</th>
<th>Target by the DOA</th>
<th>Commercial Target (MNO PoV)</th>
<th>Measured (representative example as of snapshot)</th>
<th>Conclusion / Issues / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate UL per user (Kbps)</td>
<td>Passed 2000</td>
<td>Partially Passed 1000</td>
<td>Not Passed 1000</td>
<td>&gt;2000 6000-9000</td>
<td>Satisfying results</td>
</tr>
<tr>
<td>UL Latency (ms)</td>
<td>&lt;50</td>
<td>200&lt;X&lt;500</td>
<td>&gt;200</td>
<td>&lt;50 20-120</td>
<td>Network latency is satisfying, application latency is as expected. Need to differentiate network and application next cycle</td>
</tr>
<tr>
<td>Consistency/Coverage</td>
<td>&gt;95%</td>
<td>95% &gt;90%</td>
<td>X &gt;90%</td>
<td>&gt;95% &gt;95%</td>
<td>In-lab full coverage</td>
</tr>
</tbody>
</table>
3.3.6 Planning for Cycle 3 trials

In cycle we plan to take closer look on outdoor tests with multiple smartphones.

Furthermore, the planning is to check outdoor performance in different distances from the base-station.

Following are the planned tests:

<table>
<thead>
<tr>
<th>#smartphones</th>
<th>Delay</th>
<th>Configuration</th>
<th>Indoor/Outdoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>Generic</td>
<td>Indoor</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Uplink</td>
<td>Indoor</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Generic</td>
<td>Outdoor close</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Uplink</td>
<td>Outdoor close</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Generic</td>
<td>Outdoor far</td>
</tr>
<tr>
<td>2</td>
<td>1.8</td>
<td>Uplink</td>
<td>Outdoor far</td>
</tr>
<tr>
<td>2</td>
<td>1.8</td>
<td>Generic</td>
<td>Indoor</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Uplink</td>
<td>Indoor</td>
</tr>
</tbody>
</table>

3.4 UC4.4

3.4.1 UC test objective, design and deployment

This use case is about media-related experience in mainly generating professional and semi-professional content for various purposes including live news coverage, live sports and other entertainment coverage, telemedicine related live transmission such as video from rural medical centres to medical experts or live from moving ambulances, real time security cameras, automotive-generated content such as multiple cameras enabling teleoperation or uploading huge amounts of sensory-generated data.

For the broadcast and news coverage transmissions, this continues the application of replacing the traditional satellite trucks (disruptive) while allowing many more use cases (e.g. news coverage from indoors, from underground, at no-time notice, on the move, from drones, etc.).

When either very reliable transmission is needed or high video quality (in the past full HD, nowadays 4K and in the coming years even 8K), and/or during mobility, then multi-link bonding may be used. In this use case the professional or semi-professional content generator minimizes the risk for transmission failure from anywhere and regardless of potential degradations in any single network, aggregates bandwidth from one or more networks and in our project – also trialling aggregating bandwidth from two 5G slices of the same Mobile Network Operator, to provide professional grade transmission. Testing this multi-slice bonding depends availability of multiple slices at the UoP VINNI platform (currently not supported).

The single modem transmission and the various bonding combinations that will be tested are detailed as separate use case scenarios. They include transmission of encoded video over a single 5G modem, of bonding multiple 5G connections of the same slice type, of different slice types, of 4G with 5G, of 5G with WiFi, and more. Bonding different types of slices shall include a generic default slice with a dedicated slice for upstream video (such as
defined in the NEM-5G “Media Slice”). This special slice, if it will be supported by the VINNI UoP facility, may be set up dynamically on demand (real time or in advance reservation). We shall measure the dynamics of such operations, while considering various factors such as:

- **Event type:** Basically, any event or “non-event” can be the source generating this content in both live video and non-live uploading. News happens anywhere, anytime. Sports and entertainment event also happen everywhere, including in the most remote and wild areas.
- **Location:** Anywhere, including areas of very poor 5G or even 4G coverage, by any single MNO.
- **Live video generation:** In this UC live/real time content is generated at various qualities. For the professional TV and especially Sports rights holders, 4K becomes the standard. For professional news coverage, HD is used many times while 4K is more and more needed. Video is compressed though, with HEVC H.265 real time encoders and then transmitted to the TV studio or a cloud virtual receiver where it is decoded and handled.
- **Sizing:** Scalable. In many cases there would be more than one broadcasting teams at the same location covering the same event. So, the number ranges from a single camera to tens of professional and semi-professional TV cameras.

The following figure illustrates a real-life production involves all components needed for generating content from a live complex event. It includes examples of equipment (as labelled) for contribution from the field using various equipment types, production of the content and distribution via many channels.

![Production Workflow Diagram](image)

Figure 16: Real World complex Production: Epic Mountain Bike, multi-source, remote areas, multiple feeds.

Details for the external In-Lab testing performed for this UC can be found in Section 6.1 of the Annex.

### 3.4.2 UC Architecture

The test architecture is in the diagram below.

The camera feed provided by Blackmagic payout player is feeding with the stream (equal to camera feed but with the advantage of complex scenes). The unit is controlled either by the manual operator or from remote – via the LiveU web-based control SW or in this test case – via the CDSO orchestrator (Nokia).
The CDSO function here is to (a) initiate loading the right service/slice (according to pre-configured parameters defined with CTTC) into the 5G-VINNI platform (b) “coordinate” and control the test flow by commanding the right LiveU unit to start/stop transmission while informing of that to the Visualization system (AppART).

Using the coordination and control information, the Visualization system knows when to start collecting the platform/network information. This enables it to correlate the LiveU application data with the network/platform data.

During the test, the LiveU log files record the application level performance data such as bandwidth, latency and error rate, per each of the links/modems it uses. After the test ends, the LiveU application log files are manually collected, pre-processed with a special proprietary SW so to extract the relevant information, CSV files are created and manually uploaded into the visualization system portal.

The visualization system analyses and correlates the application performance and the network/platform data. It will show, per modem/IP connection, the bandwidth, latency and error over time from the application level and from the core level. Statistical analysis such as standard deviations will also be done.

In cycle 2 we extended the test cycle 1 basic configuration:

1. We used a “standard” 5G slice as well as “Uplink” slice - NEM UL slice.
2. We used two modems per unit and enabled bonding

The control and data flows are illustrated in the figure below:

Figure 17: UC4.4 architecture
Table 13: UC4.3 Network Configuration Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Generic Configuration</th>
<th>UL oriented Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq</td>
<td>3.6GHz (n78)</td>
<td>3.6GHz (n78)</td>
</tr>
<tr>
<td>BW</td>
<td>50 MHz</td>
<td>50 MHz</td>
</tr>
<tr>
<td>DL</td>
<td>2 × 2 (MIMO)</td>
<td>2 × 2 (MIMO)</td>
</tr>
<tr>
<td>UL</td>
<td>1 × 1</td>
<td>1 × 1</td>
</tr>
<tr>
<td>DL / UL slots</td>
<td>6:3</td>
<td>2:7</td>
</tr>
<tr>
<td>Modulation</td>
<td>qam64 / qam64</td>
<td>qam256/ qam256</td>
</tr>
</tbody>
</table>

3.4.3 Trials description and measured KPIs

Table 14: UC4.3 test cases

<table>
<thead>
<tr>
<th>Unit S/N</th>
<th>#modems</th>
<th>Delay</th>
<th>Configuration</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>802031-12931</td>
<td>2</td>
<td>0.8</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>802031-12931</td>
<td>2</td>
<td>1</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>802031-12931</td>
<td>2</td>
<td>1.2</td>
<td>Generic</td>
<td>No</td>
</tr>
</tbody>
</table>
### Table 15: UC4.3 KPIs

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Justification</th>
<th>Measurement details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate UL per user (Kbps)</td>
<td>20-40Mbps</td>
<td>LU600 upload speed, Generic configuration target: overall 20Mbps, Uplink configuration target: overall 40Mbps</td>
<td>Uplink throughput measurement at application server and network</td>
</tr>
<tr>
<td>UL Latency</td>
<td>Application 1.8sec, network: &lt;200ms</td>
<td>overall 5G (RAN+Core) UL latency and application latency</td>
<td>Uplink throughput measurement at the application server</td>
</tr>
<tr>
<td>Consistency</td>
<td>&gt;99%</td>
<td>Broadcasters and their viewers are looking for consistency in QoS and QoE</td>
<td>% of the time that UL rate was above the acceptable rate to stream</td>
</tr>
<tr>
<td>Reliability</td>
<td>&gt;99.5%</td>
<td>High to satisfy user experience, but it’s an entertainment, not a critical application</td>
<td>% of data delivered without data corruption</td>
</tr>
</tbody>
</table>

### 3.4.4 Lessons learned from the Deployment phase

In Figure 19, the achievable UL rate and the measured latency of LiveU’s LU800 unit during the streaming process are depicted. Both generic and UL-oriented network configurations are examined, while the value of $d$ is set to 1.2 sec. In order to have more realistic conditions, load generation is enabled. The achievable rate of each internal 5G modem individually but also the total rate are presented, proving the efficiency of bonding. In terms of achievable UL rate, the improvement is evident, when the configuration adapted to UL transmission is employed. In case of the generic configuration, an average rate of 12 Mbps is observed, in contrast with the case of the UL-oriented configuration where an average rate of 80 Mbps is reported. Apart from the total rate, it is also observed that the behavior of each modem is more constant during the streaming process, despite the introduced load by load generation, in the case of UL-oriented configuration. Latency is the other KPI metric.
examined in this work, which is approximately consistent throughout the test period in the case of UL-oriented configuration. However, when the generic configuration is utilized, small fluctuations are observed. Both cases achieve the target value of latency, which equals to 300 msec. Latency is defined as the delay between the time of the captured video frame until the same frame is displayed to the end user. The reported time includes video capture, compression, transmission to cloud/studio and decoding.

![Graphs showing achievable rate and latency](image)

Figure 19: LU800 unit achievable rate and latency

### 3.4.5 Results comparison with Cycle 1

Performance results outperform the corresponding results of Cycle 1, where the maximum achieved UL rate was 30 Mbps. The comparison is not fair though, due to the fact that in Cycle 1 only one model was utilized not being able to exploit the bonding functionality. In addition, load generation was enabled in cycle 2.

### 3.4.6 Results comparison with 5G targets

<table>
<thead>
<tr>
<th>Trial KPI Targets</th>
<th>Target by the DOA</th>
<th>Commercial Target (MNO PoV)</th>
<th>Measured (representative example as of snapshot)</th>
<th>Conclusion / Issues / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tech KPI</strong></td>
<td>Passed</td>
<td>Partially Passed</td>
<td>Not Passed</td>
<td>Satisfying results most of the time with some drops in transmission in different scenarios</td>
</tr>
<tr>
<td>Data Rate UL per user (Mbps)</td>
<td>&gt;40</td>
<td>40&lt;X&lt;20</td>
<td>&lt;20</td>
<td>20-40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Table 16: UC4.4 results comparison with 5G targets
D6.6-D6.3B: LL performance evaluation and lessons learned v2

3.4.7 Planning for Cycle 3 trials

In cycle 3 we plan to repeat the tests with new configurations:

1) Add outdoor tests.
2) Add 2K resolution.
3) Add another modem to 3 modems tests.
4) Focus on one low app delay configuration - 0.8 sec.

Table 17: UC4.4 Planned tests

<table>
<thead>
<tr>
<th>#modems</th>
<th>Resolution</th>
<th>Delay</th>
<th>Configuration</th>
<th>Load</th>
<th>Indoor/Outdoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>No</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>Yes</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>No</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>Yes</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>No</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
<td>Outdoor</td>
</tr>
</tbody>
</table>

3.4.7.1.1

3.5 UC4.5

3.5.1 UC test objective, design and deployment

Immersive gaming is an upcoming trend due to the increase in popularity of VR devices and cheaper availability. Playing multiplayer immersive game required new capabilities for the network as the gamers are more prone to noticing lag than in non-VR games. This requirement also stems from the need to run the game at 90Hz than normally 30 or 60Hz. The Usecase works on testing and showcasing the capabilities of 5G to allow smooth
multiplayer immersive gaming with minimal lag. The use case is running a multiplayer shooting game on a VR Headset which can be played by multiple users at the same time to shoot aliens in a martian environment. Details for the external In-Lab testing performed for this UC can be found in Section 6.2 of the Annex.

3.5.2 UC Architecture

The following figure presents the architecture for UC4.5.

![UC Architecture Diagram](image)

3.5.3 Trials description and Measured KPIs

The trials for the UC took place at Patras facilities. The main KPIs measured are the following:

The main Network KPIs:

- Connection testing
- BitRate
- Total Bytes

The main Application KPIs

- Time to connect to server
- Time to start the server
- Average Ping to server
- Average input lag

3.5.4 Lessons learned from the Deployment Phase

The main lessons learned are the following

- Use of 5G CPE to connect to the 5G Network
- Optimization of KPI Integration with the
3.5.5 Results comparison with 5G targets

3.5.6 Planning for Cycle 3 trials

The main plan for Cycle 3 is to integrate SLA from CTTC as well as work towards sending a stream of KPIs from the UEs.

3.6 UC4.6

3.6.1 UC test objective, design and deployment

Cooperative Media Production, or At-Home Production or Remote Integration (REMI) has become the new standard for production companies, broadcasters and sports organizations of all sizes.

Wireless At-Home/Cloud Production solutions allow broadcasters to reduce costs by producing live shows from a centralized studio control room instead of on-site production and satellite trucks. These on-site trucks involve huge costs both in terms of duplicated equipment and personnel, in very low efficiency due to the overhead involved such as travel time of the teams and the equipment, quality variance between events, production faults, lower efficiency in multi-site simultaneous events management etc. At-home production solves all of these inefficiencies and reduced support of live events by allowing sending only the camera and audio teams to the field, whereas the production is done either in the professional studio facility or anywhere else as seen fit.

Sports and event producers can deliver multi-camera live events while eliminating the need to spend a fortune on production vehicles, satellite uplinks and travel expenses. In this use case several cameras are each connected in the field to cellular-based transmission devices, including bonding devices to provide the utmost reliability and bandwidth. These field devices then transmit the video stream over the cellular and the standard public internet (ISP) to a single receiving/decoder server with several physical SDI outputs.

LiveU shall provide several (probably 4) bonding video encoders-transmitters which will use its Precision Timing feature to allow synchronization of the video streams at the receiving end – the remote production software.

The multiple video streams in the station or production facility are then used for the actual remote production. Video quality is up to 4K from each camera.

This UC applies mainly to:

- *Sports coverage – real time mainly, from venues, outdoors, first tier sports all the way to minor and local events and leagues*
- *Real time coverage of additional events. Brands campaigns, cultural and entertainment events, enterprises events, political events etc.*

3.6.2 UC Architecture

The test architecture and flow are defined elsewhere is in the diagram below. Basically, it is similar to UC4.4 except that we will use two LiveU units transmitting simultaneously. Both will be set to the minimal possible latency that also provides stable fixed-latency transmission using the LiveU synchronized bonding and fixed latency transmission capabilities. The data and control flows are illustrated below:
As the Patras summer Festival was cancelled due to the COVID-19 pandemic, the tests had to be carried out in-lab, and without load or real world “action” festival. Nevertheless, we managed to test bonding, use the uplink slice and simulate load using load application developed at UoP testbed.

Figure 21: UC4.6 Information flow

Figure 22: UC4.6 Architecture

Table 18: UC4.6 Network configuration table
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Generic Configuration</th>
<th>UL oriented Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq</td>
<td>3.6GHz (n78)</td>
<td>3.6GHz (n78)</td>
</tr>
<tr>
<td>BW</td>
<td>50 MHz</td>
<td>50 MHz</td>
</tr>
<tr>
<td>DL</td>
<td>2 × 2 (MIMO)</td>
<td>2 × 2 (MIMO)</td>
</tr>
<tr>
<td>UL</td>
<td>1 × 1</td>
<td>1 × 1</td>
</tr>
<tr>
<td>DL / UL slots</td>
<td>6:3</td>
<td>2:7</td>
</tr>
<tr>
<td>Modulation</td>
<td>qam64 / qam64</td>
<td>qam256/ qam256</td>
</tr>
</tbody>
</table>

### 3.6.3 Trials description and Measured KPIs

#### Table 19: UC4.6 tests

<table>
<thead>
<tr>
<th>Unit #1 S/N</th>
<th>Unit #2 S/N</th>
<th>#modems in each unit</th>
<th>Delay</th>
<th>Configuration</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>0.8</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1</td>
<td>Generic</td>
<td>No</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1.2</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>0.8</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1.2</td>
<td>Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>0.8</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1.2</td>
<td>Uplink</td>
<td>No</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
</tr>
<tr>
<td>601650-60404</td>
<td>802031-12931</td>
<td>2</td>
<td>1</td>
<td>Uplink</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### Table 20: UC4.6 KPIs

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Justification</th>
<th>Measurement details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Rate UL per user</td>
<td>20-40Mbps per unit</td>
<td>LU600 and LU800 together upload speed, Generic configuration target: overall 20Mbps, Uplink configuration target: overall 40Mbps</td>
<td>Uplink throughput measurement at application server and network</td>
</tr>
<tr>
<td>UL Latency</td>
<td>Application 1.8sec, network: &lt;200ms</td>
<td>overall 5G (RAN+Core) UL latency and application latency</td>
<td>Uplink throughput measurement at the application server</td>
</tr>
<tr>
<td>Consistency</td>
<td>&gt;99%</td>
<td>Broadcasters and their viewers are looking for consistency in QoS and QoE</td>
<td>% of the time that UL rate was above the acceptable rate to stream</td>
</tr>
</tbody>
</table>
### 3.6.4 Lessons learned from the Deployment Phase

In Figure 23, the achievable UL rate and the latency metric are depicted for the UC4.6, where parallel streaming is deployed. The generic network configuration is employed, while load is not enabled. The delay value is set to 0.8sec. Measurements for each LU unit and for each modem are depicted separately. The efficiency of bonding is also observed in this case. LU800 unit has slightly outperforms LU600. Specifically, in terms of UL rate, LU600 achieves on average 3.5 Mbps approximately compared to LU800 which achieves around 6 Mbps. The reported UL rates are not very high, due to the utilization of the generic configuration. Furthermore, latency in both units is mostly under the target value of 200 msec, but in the case of LU600 frequent fluctuations are observed.

![Achievable rate and latency graphs](image)

**Figure 23: UC4.6 achievable rate and latency**

### 3.6.5 Results comparison with Cycle 1

The comparison is not fair, due to the fact that in Cycle 1 only one model was utilized not being able to exploit the bonding functionality. In addition, load generation was enabled in cycle 2.

Cycle 2 performance doesn’t reflect improvement in the overall bandwidth and latency. Nevertheless, the reliability has improved due to the bonding mechanism inherent in LiveU’s units and network allowance.
3.6.6 Results comparison with 5G targets

Table 21: UC4.6 results comparison with 5G targets

<table>
<thead>
<tr>
<th>Tech KPI</th>
<th>Trial KPI Targets</th>
<th>Target by the DOA</th>
<th>Commercial Target (MNO PoV)</th>
<th>Measured (representative example as of snapshot)</th>
<th>Conclusion / Issues / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Passed</td>
<td>Partially Passed</td>
<td>Not Passed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Rate UL per user (Mbps)</td>
<td>&gt;40</td>
<td>40&lt;X&lt;20</td>
<td>&lt;20</td>
<td>&lt;1000</td>
<td>40 per unit</td>
</tr>
<tr>
<td>UL Latency (ms)</td>
<td>&lt;50</td>
<td>200&lt;X&lt;50</td>
<td>&gt;200</td>
<td>&lt;100</td>
<td>Application 1.8sec, network: &lt;200ms</td>
</tr>
<tr>
<td>Consistency/Coverage</td>
<td>99.9%</td>
<td>99.9% &gt; X</td>
<td>95% &gt; X</td>
<td>&gt;99.9%</td>
<td>&gt;99%</td>
</tr>
<tr>
<td>Reliability</td>
<td>&gt;99%</td>
<td>99% &gt; X</td>
<td>98% &gt; X</td>
<td>&gt;95%</td>
<td>&gt;99%</td>
</tr>
</tbody>
</table>

3.6.7 Planning for Cycle 3 trials

In cycle 3 we plan to repeat the tests with new configurations:

1) Add outdoor tests.
2) Add 2K resolution.
3) Add another modem to 3 modems tests.
4) Focus on one low app delay configuration - 0.8 sec.

Following are the planned tests:

Table 22: UC4.6 Planned tests

<table>
<thead>
<tr>
<th>#modems in each unit</th>
<th>Resolution</th>
<th>Delay</th>
<th>Configuration</th>
<th>Load</th>
<th>Indoor/Outdoor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>No</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>Yes</td>
<td>Indoor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>No</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
<td>Indoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>No</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Generic</td>
<td>Yes</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>No</td>
<td>Outdoor</td>
</tr>
<tr>
<td>3</td>
<td>2K</td>
<td>0.8</td>
<td>Uplink</td>
<td>Yes</td>
<td>Outdoor</td>
</tr>
</tbody>
</table>
4 Consolidated feedback from Cycle 2 deployment phase and lessons learned – Recommendations for Cycle 3 trials and the testbed

Cycle 2 deployment phase provided a lot of valuable lessons in terms of tackling integration problems and KPI performance of the various UCs. During Cycle 2 integration of all relevant components that support the UCs architecture was the focus for most LL4 UCs, with the integration being successful and providing additional functionality e.g. ZTA, testing northbound APIs towards CDSO and implementing VNFs and VNF chains. Multiple end to end trials have been performed as part of Cycle 2 and each UC made important observations and conclusions both for architecture changes that might be needed or improvements in the KPIs that were achieved as part of those trials.

Further changes will be made respectively for each UC in Cycle 3, based on the data gathered and the lessons learned from the previous Cycle. Most changes focus on details specific for each UC like algorithm fine-tuning, additional parameter exploration and scenario enhancement. Furthermore, multiple UCs plan to take advantage of the outdoor capabilities of the testbed in Cycle 3, with additional outdoor tests. Those tests will provide valuable comparisons with the indoor ones and will pave the way for concluding the effect the different testing scenarios will have on each UC.
5 Conclusions and Next Actions

In this deliverable the objective, architecture, deployment, measured KPIs, lessons learned, result comparison with Cycle 1 and 5G targets and future Cycle 3 planning for the LL4 UCs for Cycle 2 trials were analyzed. Through these trials all UCs gathered valuable experiences and lessons that will be utilized to mature the deployment and results of the UCs in the forthcoming final Cycle.

During Cycle 2, UCs were deployed and tested successfully to the UoP testbed and were integrated with additional 5G-SOLUTIONS services like ZTA and CDSO.

Results comparison, lessons learned and the updated architecture for each UC that are described in this deliverable provide a solid ground and reference point for the upcoming Cycle 3 trials of the 5G-SOLUTIONS verticals and especially LL4.
6 Annex

6.1 External In-Lab testing for UC4.4

This use case is about media-related experiences in generating professional and semi-professional content for various purposes, including live news coverage, live sports and other entertainment coverage, telemedicine-related live transmission such as video from rural medical centers to medical experts or live from moving ambulances, real time security cameras, automotive-generated content such as multiple cameras enabling teleoperation or uploading huge amounts of sensory-generated data.

This section gathers the details about the in-lab implementation of a Network Service (NS) able to balance the load of incoming video streams as to assess multi-link reliability (at least 20 Mbps for each uplink continuous transmission) and reduce slice reconfiguration timings (< ~3 minutes).

The implemented NS relies on a Container Orchestration Engine (COE) as PaaS, in the same configuration as proposed for UC 4.1 ODS. COE takes care of Life-Cycle Management (LCM) operations of service components, particularly automatic horizontal scaling operations subject to predefined telemetry thresholds associated to Container Network Functions (CNF). In summary, the NS implements an IP Virtual Server (IPVS) as load balancer for a dynamic pool of endpoints, i.e., Video Sinks as CNF. As the resource-stress (e.g., %cpu usage, memory usage) increases, the NS dynamically scales-out/in the number of Video Sinks, therefore dynamically changing the total capacity of supported User or Machine Generated Content: allocating more resources when flows increase, while destroying unused resources when no longer needed.

6.1.1 Leveraging virtual infrastructure elasticity

Figure 24 shows the proposed architecture of the NS. The proposed NS is taking advantage of IP Virtual Server\(^3\) and Horizontal Pod Autoscaler (HPA)\(^4\) for service load balancing and automatically management of resources such as Video Sink Pod’s\(^5\) creation and deletion. The involved main components in the architecture are as following:

- **IPVS**
  - IP Virtual Server (IPVS) is Transport layer load balancer which directs requests for TCP and UDP based services to real servers. It is part of the Linux Virtual Server (LVS) project. As with IP-tables, IPVS is built on top of Netfilter. IPVS also supports 3 load balancing modes: NAT, DR (L2 load balancing via MAC rewriting) and IP Tunneling. In a nutshell, IPVS provides better performance (Hashing vs. Chains), more load balancing algorithms, supports server health checks, connection retries and sticky sessions.
  - IPVS implements by default a set of load-balancing algorithms such as round-robin, weighted round robin, least Connection, weighted least Connection, and resource based.

- **Horizontal Pod Autoscaler (HPA)**
  - HPA is implemented as a Kubernetes API resource and a controller that provides automatically scaling up or down Pods (where a deployment such as a Video Sink is hosted) according to defined (or custom) metrics such as CPU, memory, packets-per-second, requests-per-second, http-requests, queue-messages-ready, etc.

By utilizing these two main components, the proposed NS can balance the streams among a dynamic pool of Video Sinks, making a more effective use of network resources. The manner of the load balancing is implemented leveraging IPVS, while the scaling-out of Video Sink is subject to HPA and monitoring. That is, a new instance of Video Sink is created if the average active streams per Video Sink surpasses a threshold, s.

---


\(^4\) https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

\(^5\) Kubernetes Pods and CNFs will be used interchangeable throughout this section.
6.1.2 Experimental Scenarios

The development of this use case will allow testing different combinations of load balancing algorithms and various thresholds determining scaling out of Video Sinks.

Figure 25 shows an example workflow from an experiment where an OSS requests to upstream \( a+b+c+\ldots+z \) streams, but only \( a+b+c \) streams are initially admitted. Then, HPA operations extend the pool of Video Sink Pods, which in turn admit the remaining \( d+e+\ldots+z \) UGC streams.

The following illustrates the experimental workflow:

Figure 25: Pushing a great number of streams to a dynamic pool of Video Sinks. HPA subscribing to \%cpu usage by Pod.
1. OSS/BSS sends request to the controller.
2. OSS/BSS get notified of a Video Sink from controller.
3. Before starting upstream request from OSS/BSS, some metrics such as CPU and memory utilizations must be specified by the Horizontal Pod Auto scaler (HPA) according to the identified KPIs.
   a. To be more specific, metrics such as CPU and memory limitation and request must be specified during Video Sink deployment over the Kubernetes Pod manifests (e.g., limits: cpu: 2500m requests: cpu: 100m). Then the interval of desired Pod replicas (minReplicas: 1, maxReplicas: 5); CPU and memory utilization percentage must also be specified in HPA deployment (e.g., \texttt{targetCPUUtilizationPercentage: 50}).
4. OSS/BSS request upload of \((a+b+c+...+z)\) streams to the IPVS.
5. The IPVS load balances the upstream \(a+b+c+...+z\) to an existing Video Sink Pod (VSP1).
6. The VSP1 accepts upstreams \(a+b+c\), but upstream \(d+e+...+z\) are denied due to overloaded VSP1’s CPU and memory utilizations, and get send back to the OSS/BSS.
   a. For example, when CPU utilization is 138% (Figure 27), VSP1 cannot accept other streams \((d+e+...+z)\).
7. After successful deployment of VSP and HPA, HPA scales out/in between the mentioned desired VSPs looking at CPU utilization metric, automatically.
   a. In this case, monitoring system gathers specified metrics (in this example %cpu usage) which are relayed to HPA. Finally, HPA determines scaling operations subject to the policy applied to the configured metrics. This process is illustrated in Figure 28.
   b. In the following, HPA in terms of CPU usage is shown. In Figure 26, HPA shows that Pod \texttt{myapi} CPU utilization is 5%, which is under the target CPU utilization (50%) triggering scaling operations. By increasing the load on the Pod to 192% (Figure 27) HPA automatically creates pod replicas and distributes the workload between the newly created pods, decreasing the average CPU utilization from 192% to under 50% (Figure 28). When the streams finish, all created replicas are removed (Figure 26).

\[
\begin{array}{|l|c|c|c|c|c|c|}
\hline
\text{NAME} & \text{REFERENCE} & \text{TARGETS} & \text{MINPODS} & \text{MAXPODS} & \text{REPLICAS} & \text{AGE} \\
\hline
\text{horizontalpodautoscaler.autoscaling/myapi} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 2dh \\
\text{pod/alarmmanager-prometheus-prometheus-oper-alertmanager-0} & \text{Deployment/myapi} & 192\%/50\% & 1 & 5 & 4 & 26dh \\
\text{pod/myapi-6b4dccc89bc-n58bn} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 207h \\
\text{pod/myapi-6b4dccc89bc-daqir} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 6h4n \\
\text{pod/myapi-6b4dccc89bc-fltw} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 115s \\
\text{pod/myapi-6b4dccc89bc-m58mje} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 6h4n \\
\text{pod/myapi-6b4dccc89bc-zx5wb} & \text{Deployment/myapi} & 5\%/50\% & 1 & 5 & 1 & 115s \\
\hline
\end{array}
\]

Figure 26: Experimental setup with HPA (metric: CPU)

\[
\begin{array}{|l|c|c|c|c|c|c|}
\hline
\text{NAME} & \text{REFERENCE} & \text{TARGETS} & \text{MINPODS} & \text{MAXPODS} & \text{REPLICAS} & \text{AGE} \\
\hline
\text{horizontalpodautoscaler.autoscaling/myapi} & \text{Deployment/myapi} & 49\%/50\% & 1 & 4 & 1 & 26dh \\
\text{pod/alarmmanager-prometheus-prometheus-oper-alertmanager-0} & \text{Deployment/myapi} & 192\%/50\% & 1 & 4 & 4 & 267h \\
\text{pod/myapi-6b4dccc89bc-daqir} & \text{Deployment/myapi} & 5\%/50\% & 1 & 4 & 1 & 320ms \\
\text{pod/myapi-6b4dccc89bc-fltw} & \text{Deployment/myapi} & 5\%/50\% & 1 & 4 & 1 & 320ms \\
\text{pod/myapi-6b4dccc89bc-m58mje} & \text{Deployment/myapi} & 5\%/50\% & 1 & 4 & 1 & 320ms \\
\text{pod/myapi-6b4dccc89bc-zx5wb} & \text{Deployment/myapi} & 5\%/50\% & 1 & 4 & 1 & 320ms \\
\hline
\end{array}
\]

Figure 27: Overloading CPU and pods creation

Figure 28: HPA achieved less than target utilization

8. Once HPA creates the new Video Sink Pod replica, requests to VSP are balanced by IPVS (using Round Robin algorithm), this is shown in Figure 29 and Figure 30.
6.1.3 Remarks

- IPVS can use different algorithms such as round-robin, weighted round robin, least Connection, weighted least Connection, resource based etc. for load-balancing behind the scenes.
- Based on fair distribution of network resources the round-robin algorithm provides the best results.
- For implementation cycle 3 we expect to:
  - Identify the most relevant metrics for HPA to achieve efficient scaling-out i.e., to provide best network throughput with less CNF instances.
  - Compare the required slice configuration for: i) scaling of PaaS resources, 2) scaling of CNF resources. Both yielding indicators of a global slice reconfiguration time.

6.2 External CCTC In-Lab testing for UC4.5

This use case will explore the effect of inter-slice interference on running services. Furthermore, it will effectively recreate scenarios where SLA/KPIs are threatened by inter-slices interference. That is: the resources’ consumption made by tenant Y’s slice will negatively affect the performance of tenant X’s service.

We will explore how to quantify this, and what are the measures commonly taken for mitigating the effect of external VNF interference, e.g.: VNF migration, scaling, placement, etc.

According to the Description of Work (DoW), CTTC commitment will be to:

provide effective innovative E2E dynamic inter-slicing algorithms to enable a seamless collaborative gaming experience by bonding resources across different network domains.

6.2.1 Inter-slicing algorithms

This refers to OSS/BSS-level algorithms able to control more than one Network Slice Instance (NSI) and redistribute their allocated resources (e.g., leveraging NFV MANO APIs) as to minimize SLA violations.

In the context of the DoW, CTTC will devise precise scenarios compatible with UC 4.5 (e.g., URLLC slices) and recreate them under an emulation environment compatible with UoP (i.e., CTTC end-to-end experimental platform).

6.2.2 Bonding resources across different network domains

Bonding is the aggregation of independent resources into a new unit. In radio access network (RAN), bandwidth may be increased by bonding more than one channel, e.g., 40 + 40 = 80 MHz.
The advent of virtualization extends this concept to Edge and Cloud segments of the network. In the context of the DoW, CTTC will devise and implement mechanisms that leverage 5G NFVI elasticity and programmability as to dynamically extend/reduce virtual resources based on service-level metrics (i.e., not solely those provided by the infrastructure).

6.2.3 Discussion

Currently, scaling-out/in operations based on service-level metrics (e.g., a variable) is a clear candidate for a bonding technique (as shown in Section 6.1.1).

- The envisioned scenario will emulate two independent Network Slice Subnet Instances (NSSI) sharing infrastructure resources (i.e., VNF interference scenarios).
- A service emulating application-level load will determine the suitability of the orchestrated resources via probing.
  o If resources are enough, nothing happens.
  o If resources are not enough, an observed metric is modified. Triggering bonding/scaling operations (this is mapped to Scenario 1 below).
- As the number of users increases, dynamic scaling operations will satisfy application-level requirements,
  o while OSS/BSS-level inter-slicing algorithm minimizes the possibility of inter-VNF interference by e.g., securing effective VNF placement or NFV MANO Object scaling\(^6\) (this is mapped to Scenario 2 below).

6.2.4 Scenario 1: Bonding resources based on service-level metrics

As the title suggests, this scenario will increase the resources of a service (scale-out) based on arbitrary service-level metrics. As e.g., service load increases, PaaS control plane will take care of scaling out resources to maintain SLAs.

UC Owner (NURO) is basing the scaling out of resources to the number of connected users, \(C\). Nevertheless, this metric is not suitable for scaling operations in cloud-native applications. Therefore we transform it to a gauge \(C^*\) which represents the number of connected per second.

\(^6\) These refer to the scaling of Container Infrastructure Service Instances (CISI), as defined in ETSI NFV-IFA 029 (see also ETSI NFV-IFA 040).
6.2.4.1 Implementation Layout

![Network Service Layout](image)

Figure 31: Network Service Layout

Figure 31 provides an overview of the components realising Scenario 1. In summary, Clients are emulated using HTTP load tools, such as Locust\(^7\). Clients then trigger Server Pod APIs in order to generate $C^*$ metrics. The latter is exposed to a Prometheus Instance, which in turn leverages libraries such as Prometheus Adapter to subscribe said metric to Kubernetes Metrics API. Kubernetes Horizontal Pod Autoscaler (HPA) is setup to observe $C^*$ and trigger the creation of new Server Pods accordingly.

6.2.4.2 Validation

The strategy should be validated by:

- Demonstrating Server Pod’s CPU usage increases with $C^*$.
- Showing how Server Pod’s CPU usage can be maintained below a threshold by applying the strategy. That is, by scaling out Server Pods we can evenly distribute the load generated by connecting Clients.

**Demonstration 1:** In this strategy 6 users are sending request to game-server. CPU target value is set up to 60% in Horizontal Pod Autoscaler (HPA) and min-max replica sets for game-server are set to 1-10. This means that when game-server’s CPU usage increases over 60%, HPA starts to create game-server’s replicas to decrease CPU usage to less than 60%. Figure 32 illustrates the set up HPA.

![HPA](image)

Figure 32: HPA (CPU target value 60%)

We test this approach over two scenarios. In Scenario 1, we test 6 users with spawn rate 1 per second. Each user increases the perceived CPU usage. As shown in Figure 33 and Figure 34, when CPU usage passes over 60% game-server replicas are created until CPU usage goes below the target value (60%) (Figure 35).

---

\(^7\) [https://locust.io/](https://locust.io/)
Figure 33: HPA value upper than target value

Figure 34: HPA value upper than target value

Figure 35: HPA value become less than target value

Figure 36 is showing the created game-server pods for bringing CPU target value lower than 60%.

Figure 36: Game-server pods

You can observe the overall HPA function for creating and deleting game-server replica sets in Figure 37.

Figure 37: HPA lifecycle

In the following Figures, all information related to number of users, total number of requests, and respond time are shown.
Demonstration 2: The same test is done with 1000 users with spawn rate 6 (users/second). Figure 41, Figure 42 and Figure 43 are showing the game-server CPU target value is increasing over 60% then HPA creates game-server’s replica sets to bring CPU usages lower than 60%.

The HPA lifecycle is shown in Figure 44.
All the information regarding the 1000 users load generation can be found in Figure 45, Figure 46 and Figure 47.

Figure 44: HPA lifecycle with 1000 user test

Figure 45: Total number of users

Figure 46: Total requests per seconds

Figure 47: Response times
Demonstration 3: In this demonstration, we use custom metric (service-level based) for scaling out/in. The chosen service metric here is http requests per second, \( C^* \) to game-server. The target value is set to \( C^*=6 \) to game-server pod as shown in Figure 48.

<table>
<thead>
<tr>
<th>NAME</th>
<th>REFERENCE</th>
<th>TARGETS</th>
<th>MINPODS</th>
<th>MAXPODS</th>
<th>REPLICAS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-server</td>
<td>Deployment/game-server</td>
<td>33m/6</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>9m26s</td>
</tr>
</tbody>
</table>

Figure 48: http requests per second HPA

When target value passes over \( C^* \) HPA starts to create game-server replicas until current detected value of \( C^* \) goes below the threshold, as in Figure 49, Figure 50, and Figure 51. Figure 52 is illustrating 10 created game-server pods.

<table>
<thead>
<tr>
<th>NAME</th>
<th>REFERENCE</th>
<th>TARGETS</th>
<th>MINPODS</th>
<th>MAXPODS</th>
<th>REPLICAS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-server</td>
<td>Deployment/game-server</td>
<td>7489m/6</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>11m</td>
</tr>
</tbody>
</table>

Figure 49: http request target value increases over 6 users per second

<table>
<thead>
<tr>
<th>NAME</th>
<th>REFERENCE</th>
<th>TARGETS</th>
<th>MINPODS</th>
<th>MAXPODS</th>
<th>REPLICAS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-server</td>
<td>Deployment/game-server</td>
<td>14675m/6</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>12m</td>
</tr>
</tbody>
</table>

Figure 50: http request target value increases over 6 users per second

<table>
<thead>
<tr>
<th>NAME</th>
<th>REFERENCE</th>
<th>TARGETS</th>
<th>MINPODS</th>
<th>MAXPODS</th>
<th>REPLICAS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-server</td>
<td>Deployment/game-server</td>
<td>5872m/6</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>18m</td>
</tr>
</tbody>
</table>

Figure 51: http request target value become less than 6

<table>
<thead>
<tr>
<th>Pod Name</th>
<th>Replicas</th>
<th>State</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-server-8cbbbb5dd88-25rgq</td>
<td>1/1</td>
<td>Running</td>
<td>114s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-5dcqh</td>
<td>1/1</td>
<td>Running</td>
<td>114s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-8tbd7</td>
<td>1/1</td>
<td>Running</td>
<td>53s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-9hrjb</td>
<td>1/1</td>
<td>Running</td>
<td>34m</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-cnjwv</td>
<td>1/1</td>
<td>Running</td>
<td>53s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-cpxq6</td>
<td>1/1</td>
<td>Running</td>
<td>114s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-fm57n</td>
<td>1/1</td>
<td>Running</td>
<td>83s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-tf2j5</td>
<td>1/1</td>
<td>Running</td>
<td>2m25s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-vsxsj</td>
<td>1/1</td>
<td>Running</td>
<td>2m58s</td>
</tr>
<tr>
<td>game-server-8cbbbb5dd88-vqqjc</td>
<td>1/1</td>
<td>Running</td>
<td>84s</td>
</tr>
</tbody>
</table>

Figure 52: Created game-server’s pods

The HPA lifecycle can be seen in Figure 53.

All the information regarding the load generating for 1000 users with spawn rate 6 can be found in Figure 54, Figure 55 and Figure 56.
Figure 53: HPA lifecycle for http request per second metric

Figure 54: Total number of users

Figure 55: Total number of requests per second
6.2.5 Scenario 2 proposal for Cycle 3: Condition service placement (or migrate) to minimize inter-slice interference

During orchestration or even during operations, PaaS control plane will make sure requested resources are guaranteed. This scenario will show how service components can be migrated to appropriate locations withing NFVI, or its orchestration is conditioned by available resources\(^8\).

**6.2.5.1 Implementation Layout**

Figure 57 shows a scenario where CNF system resources’ requests are specified at orchestration time. These condition the placement of CNF-B to Network Functions Virtualization Infrastructure (NFVI) nodes with the specified availability.

In turn, Figure 58 produces a CNF-B migration due to NFVI nodes failing to provide the placement conditions specified at orchestration time.

Both mechanisms highlight 5G cloud/edge segments’ ability to dynamically distribute slices’ load on the underlying resources. This allows service owners to specify their requirements at orchestration time and leave (some) service lifecycle operations to the infrastructure itself.

---

\(^8\) [https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/](https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/)
Figure 57: Orchestration-time Placement Decisions

Figure 58: Runtime CNF migration
6.2.5.2 Validation

Scenario 2-A, which consist of orchestration-time decisions for placement will be validated according to the following procedure:

1. Based on Deployment manifests OSS/BSS will determine NodeAffinity conditions for the deployment.
   This procedure recreates scenarios where a suitable placement at the Edge is desired.
   a. Leverages available infrastructure metrics to make placement decisions.
   b. Employs templating tools to configuring descriptors.
   c. Currently conditions are based on:
      i. Node with minimum CPU usage time.
      ii. Node with most available RAM.

Scenario 2-B is validated by:

2. Ordering an on-demand migration of a CNF based on NodeAffinity.

Triggering an automatic migration due to cohabitation with other CNF.