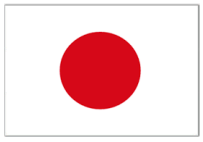



# D5.1 – Initial Report on the integration of the Testbeds, Experimentation and Evaluation

DG, UT, Fraunhofer, WU, ERICSSON, NESIC, MI, AU

| | |
|---|---|
| **Document Number** | D5.1 |
| **Status** | Final |
| **Work Package** | WP5 |
| **Deliverable Type** | Report |
| **Date of Delivery** | 16 March 2018 |
| **Responsible** | DG and FOKUS |
| **Contributors** | DG, UT, FOKUS, WU, Ericsson, NESIC, MI, AU |
| **Dissemination level** | PU |



*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723172, and from the Swiss State Secretariat for Education, Research and Innovation.*

| Authors in alphabetical order | | |
|---|---|---|
| **Name** | **Beneficiary** | **e-mail** |
| Nicklas Beijar | Ericsson | nicklas.beijar@ericsson.com |
| Ilias Benkacem | AALTO | ilias.benkacem@aalto.fi |
| Eleonora Cau | FOKUS | eleonora.cau@fokus.fraunhofer.de |
| Cédric Crettaz | Mandat International, MI | ccrettaz@mandint.org |
| Marius-Iulian Corici | FOKUS | marius-iulian.corici@fokus.fraunhofer.de |
| Fabian Eichhorn | FOKUS | fabian.eichhorn@fokus.fraunhofer.de |
| Eunah Kim | Device Gateway, DG | eunah.kim@devicegateway.com |
| Hiroshi Takezawa | NESIC | takezawa-h@sa.nesic.com |
| Tarik Taleb | AALTO | tarik.taleb@aalto.fi |
| Toshitaka Tsuda | WU | tsuda-toshitaka@nifty.com |
| Shu Yamamoto | UT | shu@iii.u-tokyo.ac.jp |
| | | |

# Executive Summary

This deliverable is the first deliverable of WP5 and includes the reporting on the work executed up to M18 for the deployment and the evaluation of the ignition testbeds and progress of the testbeds integration.

Since 5G!Pagoda aims towards the integration of a very high number of heterogeneous technologies within a multi-slice architecture, as a first step within tis development, a set of ignition testbeds were set up by different groups of partners aiming to determine their suitability to be run as slices on top of a common architecture. This deliverable includes the initial development of components, experimentation and evaluation of these testbeds. It gives the measurable means to assess both the progress into the testbed integration as well as the possible risks of implementation and mitigation actions that can be used for further refinement of the final goal of WP5.

Additionally, this deliverable includes a description of the different integration options to bring together two testbeds in Tokyo and Berlin. From these options a set of decisions were taken on the most suitable technologies to be used. A careful attention was given to the specifics of the integrated testbed especially towards the backhaul network characteristics, as the two central nodes are located at a very large network distance.

As a result of the ignition phase, it is foreseen that the 5G!Pagoda integrated testbed will provide a set of X highly heterogeneous slices proving the flexibility of the software network components developed in WP3 and the orchestration functionality developed in WP4. Additional slices could be created in a later pre-production phase towards addressing in a more specific way the use cases. This will be decided at the conclusion of the first integration phase, depending on the community requirements in Japan and in Europe.

# Table of Contents

**List of Figures**

**List of Tables**

# 1. Introduction

## 1.1. Motivation, Objective and Scope

This deliverable represents an accompanying document to the 5G!Pagoda testbed deployment performed within WP5. It includes the reporting of the work done in the first 20 months of the project and 8 months of the work package.

Specifically, this deliverable represents the first part of a set of 3 documents on the 5G!Pagoda testbed, which are usually used for the description of the practical work in terms of implementation and testbed deployment.

The main goal of the 5G!Pagoda testbed is to provide experiments and evaluations of the different 5G use cases and scenarios for 5G applications in IoT and human communication domains. In order to be able to perform these experiments in a relevant manner for Europe and for Japan, the 5G!Pagoda testbed has to pass through a set of steps: the testbeds are setup, technologies developed in WP3 and WP4 as well as third party ones are selected and integrated, an evaluation methodology is defined, evaluations are performed through experimentation campaigns, and the results are assessed and evaluated.

While there are several ignition testbeds built in parallel, two deployment nodes were selected to build an integrated testbed: one in Berlin based on and extending the Fraunhofer FOKUS 5G Playground; and one in Tokyo based on the UTokyo FLARE testbed and the O3 orchestration framework.

As a result of the testbed, the architecture proposed in WP2 will be validated. Furthermore, the technologies from WP3 and WP4 will be integrated into end-to-end scenarios, which will enable the assessment of their individual and composite capabilities as well as the building up of common dissemination and standardization activities as furthered by WP6.

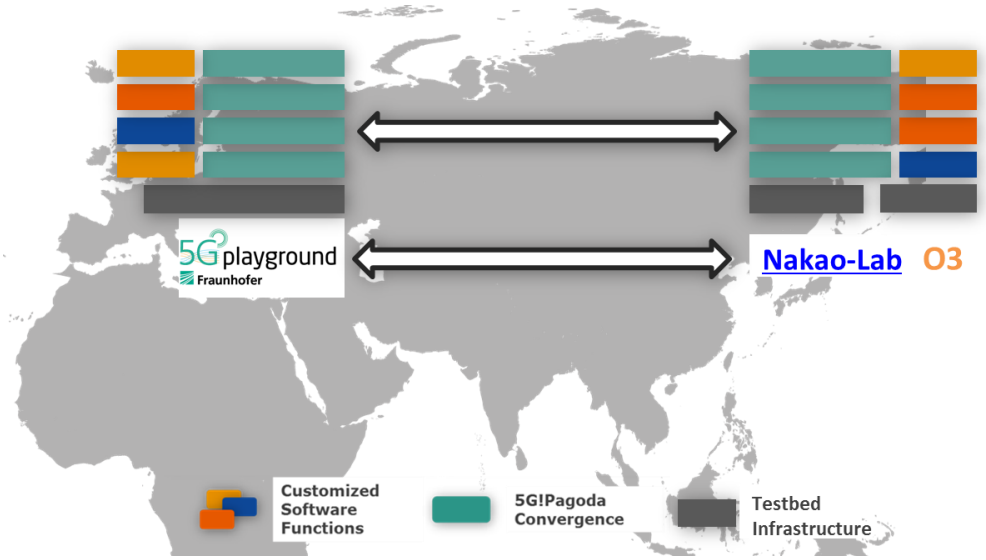


**Figure 1 – A harmonized testbed across the 5G Playground in Berlin and Nakao-lab in Tokyo as aggregation points for the project developments**

For technology transfer reasons, the 5G!Pagoda testbed will be using equivalent technologies in Europe and in Japan as part of the functional core testbed. This will enable the easy swapping of the different

deployments between the two central locations as well as the aggregation and the understanding of the technologies developed across the two continents.

The scope of the 5G!Pagoda testbed is to provide a comprehensive PoC for the multi-slicing architecture. This will be achieved by using a set of software components being developed within WP3 as well as orchestration solutions in WP4, brought together multiple use case to be run in dedicated slices on top of the same testbed. With this, 5G!Pagoda plans to validate the capability of running multiple independent software networks as slices on top of the same infrastructure. It is also meant to prove that the dedicated networks within the software slices can be customized for the specific needs of the use cases currently required in the context of 5G in Japan and in Europe. Albeit only a set of slices will be deployed in the testbed, a very wide set of communication technologies were chosen, enabling the coverage of the full spectrum of use cases. Specific optimizations and modifications of the slices are expected when going closer to production ready solutions, while the core technologies are expected to remain the same.

## 1.2. High-Level Integration Approach

Compared to other projects, 5G!Pagoda is required to integrate prototypes which come from two distinct R&D environments within a harmonized testbed infrastructure in Japan and in Europe. Because of this specificity, as well as due to the capabilities of the multi-slice architecture to handle heterogeneous networks, a new approach was taken for the integration of the testbeds, as illustrated in Figure 2.

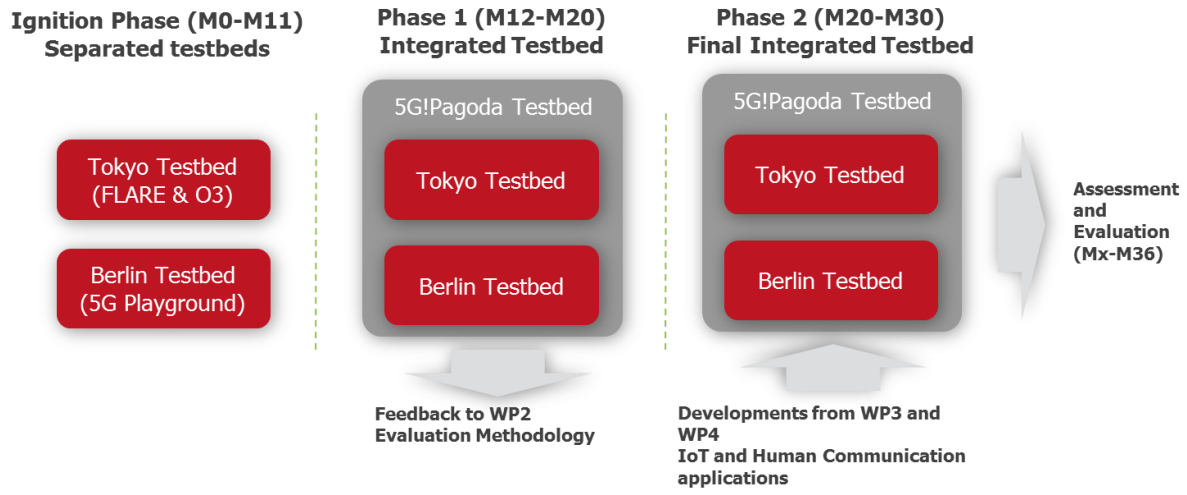


**Figure 2 - Testbed Integration Approach**

A new ignition phase was added to the integration approach. In this phase, a set of software network slices were developed based on the initial developments in WP3 and in WP4 to showcase specific use cases which are of interest towards the European and the Japanese market. The main role of this ignition phase is to lay ground for further developments of the integrated testbed by providing initial evaluation of the feasibility of the use cases to be deployed as dedicated slices on top of the multi-slice architecture.

Based on this information, an evaluation methodology suitable for the assessment of the capabilities of specific network slices to be deployed on top of a common infrastructure and to respond to the requirements of the specific use cases is to be further developed. The aim is double folded: to provide an evaluation methodology for the 5G!Pagoda testbeds and as well as an evaluation methodology for the software network slices in general.

The ignition testbeds include the 5G Playground in Berlin and the FLARE & O3 in Tokyo testbeds as well as other initiatives from the partners to build dedicated slices. These ignition testbeds are meant also to be integrated as part of the 5G!Pagoda testbed.

This operation will be executed in two phases: one phase up to Month 24 of the project and a second one with optimizations up to Month 30. The different slices will pertain to one or to the other of the two integration phases depending on their software maturity as well as on the interest of the market on such integrations. For the slices integrated up to Month 24, a further optimization towards customization for use cases will be taken up to Month 30, allowing for the initial slices to be adapted to the needs of the market.

As the WP4 orchestration framework will be completed only in Month 24, the first integration phase will include different orchestration solutions depending on the location. Furthering the integration of the orchestration framework, additional interoperability tests may be needed.

Furthering the Month 30 integration, an evaluation stage will be considered using the methodology developed up to Month 24. The evaluation stage may bring additional optimizations to the deployed slices as well as to the orchestration framework enabling them to be more fit to the requirements.

# 2. Initial experimentation and evaluation

In this section, the different ignition testbeds are described. The ignition testbeds were developed before the start of the integration work of WP5 aiming at:

- Providing an initial assessment of the capabilities to run specific dedicated networks in form of slices.
- Providing an initial evaluation of the requirements towards the integrated testbeds to be able to be on-boarded as specific dedicated slices.
- Providing an initial feedback from the third parties interested into the specific use cases on the possibilities to deploy their networks as software on top of a common network architecture.

As this initial experimentation and evaluation shown in Table 1 was executed before the start of the integration of the project, the ignition testbeds were rather diverse. Their selection was done based on the interests of 5G!Pagoda partners among the selected use cases defined in D2.1, in fulfilling the developments of WP3 and WP4 as well as in providing significant R&D developments for their market. Through collaborative working, these use cases were polished towards becoming easy to integrate in a common testbed infrastructure. This represents, on a practical level, the main goal of 5G!Pagoda: to provide a comprehensive network architecture in which a massive number of heterogeneous slices can be deployed addressing in a customized manner the different use cases of 5G.

**Table 1 A list of 5G!Pagoda ignition testbeds**

| Ignition Testbed | Services Provided |
|---|---|
| ICN-CDN | Combined content delivery solution. |
| IoT and Video slices | Dynamic slicing for emergency situation in IoT testbed and demonstrating with video streaming / real-time services. |
| Healthcare | Demonstration of the healthcare service provided by an MVNO focusing on its resource management and security enhancement. |
| Deep data Plane Programmability | Creation of a testbed for deep data plane programmable core network (DDPPCN) slices. |

For each of the ignition testbeds, a short description is added in the following sections including the architecture of the ignition testbed, the functionalities included, the description of the components and of the interfaces. The ignition testbeds were used for assessing the feasibility of the use cases to be deployed on top of common infrastructures in the form of slices. This initial evaluation completes each of the sections.

## 2.1. Ignition Testbed 1 - ICN-CDN Combined Contents Delivery

Waseda University and Aalto University developed the ICN-CDN testbed and its use case. This use case scenario aims to realize an efficient content delivery service by combining the Dynamic CDN slice (CDNaaS) and the ICN/NDN slice based on 5G!Pagoda architecture. Contents delivery, especially video contents, is

already a major traffic, occupying around 70% in mobile communication, and still increasing. It is expected that this traffic increases with the operation of 5G and will cause traffic congestion. The ICN-CDN ignition testbed aims at solving this problem by reducing the contents traffic of core network with dynamic CDN slice creation, using the optimum location assignment of content servers, and the use of ICN slice in regional distribution part of the network, combining dynamically two types of slices. This use case also intends to demonstrate some of the innovative features of 5G!Pagoda such as deep data plane programmability, slice stitching（slice chaining) capability, multi-domain orchestration, optimum VNF allocation, and slice resource scalability. This scenario also aims at connecting Japan and Europe to demonstrate the service.

### *2.1.1. Architecture description*

The overall expected network system for CDN/ICN combined content delivery service is shown in Figure 3. For the goal of efficient network resource usage and high performance communication network, Aalto University is responsible for the dynamic allocation of content server on demand through dynamic CDN slice provisioning across plural number of domains reflecting on user demand based on Internet, while Waseda University manages the autonomous distributed content cache using ICN nodes. The dynamic CDN is realized as a slice connecting beyond country boundary, and the dynamic CDN server is connected to ICN slice via the ICN gateway for the purpose of regional distribution in each country as shown in Figure 4.

In the dynamic CDN slice, IP protocol is used. The CDN servers (cache) are dynamically allocated at the suitable CDN servers close to the respective users, e.g. Finland, Germany, and Japan illustrated in Figure 3. This demonstrates the multi-domain orchestration capability. The contents that are cached at each location are smartly selected by the statistical analysis of users' preference of each location. To serve a variety of user terminal devices, transcoders are located at the appropriate locations to convert original video contents into several different standards if requested. Together with the allocation of the CDN servers, this demonstrates the optimum VNF allocation capability.

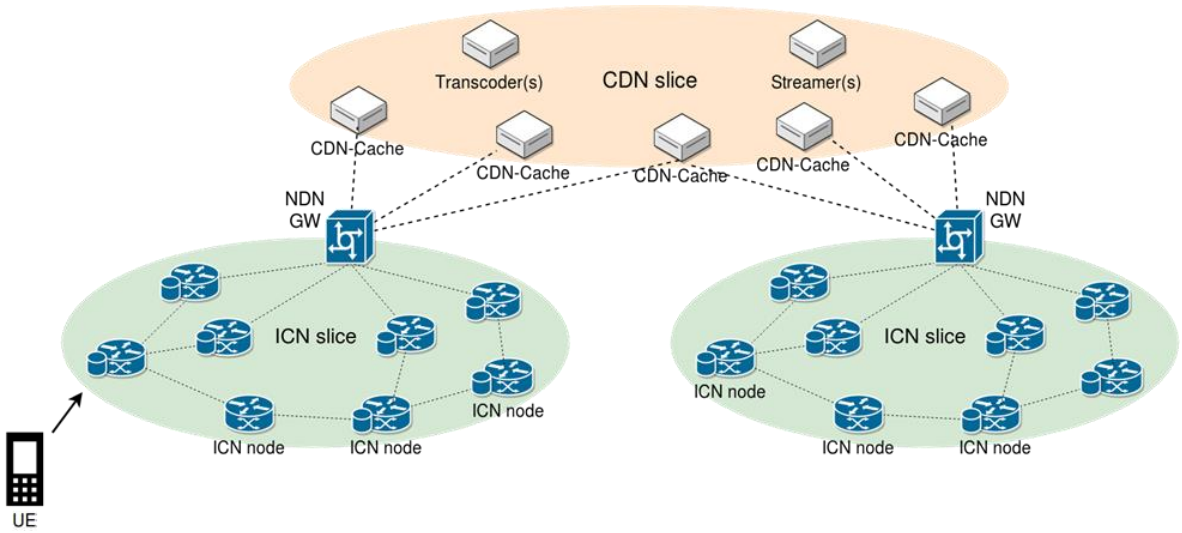


**Figure 3 - Dynamic CDN slice and interaction with ICN slice for regional distribution**

As the final stage of content delivery, the content data is transferred to the ICN slice on a request coming from the ICN slice. ICN is the emerging network architecture that uses the different routing protocol and the packet format. This means that the different data plane function from the IP is necessary. To interact with the dynamic CDN slice, a gateway function is deployed in order to communicate with the CDN slice and convert the content into the ICN/NDN compatible packet format. The gateway function is provided in

the ICN slice. From the viewpoint of traffic reduction, it is better to place the gateway function at the closest ICN node to the CDN server of the dynamic CDN slice. Since the location of CDN servers dynamically changes, it is possible that the number of the operational gateways changes also according to the creation of new Dynamic CDN slices. In addition, when the ICN node is assigned to function as a gateway, it is preferred to enlarge the cache capacity of the node to improve the cache hit ratio to the content request served with in the ICN slice, and avoid the duplicated interaction with the dynamic CDN slice. As is explained, implementing the ICN slice can demonstrate the deep data plane programmability and the scalability management of slice resources.

The implementation to realize this architecture for 5G!Pagoda Project will be conducted with FLARE that is developed by the University of Tokyo as an open deeply programmable network node architecture for next generation network. It aims to enable future network architecture that can be quickly and flexibly programmed to adapt to various user demands. In FLARE architecture, the data plane and control plane are programmable, which meets the requirement of implementing a new network architecture such as ICN. It can also realize the multiple slices, which is the fundamental requirement of 5G!Pagoda.

In summary, this use case scenario can demonstrate a solution for one of the 5G network problems on handling the major (and still increasing) contents traffic. It provides an efficient content delivery service covering a wide geometrical area that extends across the countries by the strength of pre-planned allocation with large cache capacity that the Dynamic CDN slice will provide. It also provides less volume of the regional traffic and shorter response time by the use of the limited node cache capacity but finer grain and dynamic contents allocation capability of the ICN slice. The use of ICN also make it easier to access the content for the user, and the protocol sequence will become simpler which helps the network to reduce the protocol traffic.

This use case can demonstrate some of the innovative features of 5G!Pagoda, such as the deep data plane programmability, the slice stitching (slice chaining) capability, Multi-domain orchestration, optimum VNF allocation, and slice resource scalability.

### *2.1.2. Components*

In this section the components of the slices are described, starting with the CDN and finishing with the ICN.

**Table 2 - CDN Slice Components**

| Components name | Description | Major functionalities |
|---|---|---|
| Multi-Domain Orchestrator | Essential component that has the global view on the overall topology. It presents the Multi-domain orchestration plane. The Orchestrator manages the lifecycle of virtual slices and virtual resources and performs actions including instantiation and termination of virtual network functions with | **Web server and Database management:**<br>• User-friendly Web interface designed for our subscribers that would like to manage their own slices and also an admin interface for the whole platform (That manages users etc.)<br>• This component brings retrieve the subscribers needs and store it in the database to be exploited by running algorithms implemented in other agents.<br>**OSS rules**:<br>• The Orchestrator ensures the System orchestration and manages the policies while receiving the user needs. For example: The |

| | | |
|---|---|---|
| | desired flavours. The orchestrator updates constantly the NFV Manager concerned regarding any action made to VNFs under its direction. | slice should have only one Coordinator and at least one instance for each network function (Streamer, Transcoder, and Cache) in order to ensure the highest quality of service.<br>**VIM agent:**<br>● For every available administrative cloud domain, we have a VIM agent that communicates with the respective VIM in order to instantiate new VMs or delete ones.<br>● When new machines are instantiated, the agent receives an acknowledgment from the VIM with all the information needed including Public IP and Private IP and other information.<br>**Southern API with Coordinator:**<br>● An active agent constantly in contact with the manager of each slice.<br>● Update the coordinator in every new instantiation of a VM.<br>● Populate the CoordinatorDB about the available machines with their information including the Image ID, public and private IP, hosting cloud provider, the credentials and such other essential information. |
| Slice-specific Coordinator<br>(VNF Manager) | The main component of the CDNaaS considered as the brain of the CDN slice. It presents the virtual resource management layer. The Coordinator ensures the communication and SFC between isolated collections of VNF instances. The Coordinator manages the uploaded videos through the Coordinator web interface, manages end-users and has access to the dashboard for monitoring the slice resources, content popularity and access statistics. | **Web server and Database management**<br>● User-friendly web interface designed for the Owner of the slice and other interface for end users who just want to watch videos.<br>● Manage resources.<br>● Allow owners to manage contents uploaded videos.<br>● Select which CDN-cache, transcoder server and desired streamer for each content.<br>● Data visualisation of access statistics, content popularity and VM distribution.<br>**Queuing server**<br>● By the mean of an Advanced Message Queuing Protocol, we ensure the Task distribution management over the VNF nodes.<br>● Load balancing in FIFO order. Who finished first takes the next job.<br>**VNF transcoding agent**<br>● Write the task in the appropriate transcoder queue.<br>● Receive acknowledgment from the respective transcoder node.<br>● Receive feedback: Transcoding percentage progress. |

| | | |
|---|---|---|
| | | • The agent will provide the progress to the web front end to be displayed as a progress-bar.<br>**VNF-streaming agent**<br>• Load balance the streaming tasks over the VNF streamer nodes.<br>• Receive the access information from streamers.<br>**NorthernAPI with Orchestrator**<br>• Receive constantly updates from the Orchestrator regarding any change in the slice topology. |
| VNF-Transcoder | As a virtual network function transcodes remotely videos from virtual cache servers in order to make the content available in different qualities and resolutions. | **Transcoding service:**<br>• Consume jobs from its own queue.<br>• The server in charge of remote virtual transcoding. It consumes extremely the virtual computing resources.<br>• Always listening to the coordinator orders by the mean of a queuing system, it picks up the video from the cache server concerned, starts transcoding and sends the feedback to the coordinator with a real-time progress.<br>• Acknowledgment: The end-user will be notified if the operations are successfully completed. |
| VNF-Streamer | As a virtual network function for load balancing and receiving end-user requests for playing a specific video and redirecting the request to proper cache server to show the video content using available resolutions. | **Streaming service**<br>• Virtual server is essentially based on Nginx server<br>• Take care of load balancing and receiving end-user requests for playing a specific video and redirecting the request to proper cache server to show the video content using available resolutions.<br>**Reporting access information**<br>• The server tracks also the video accesses and sends them back to the coordinator to measure statistics and analysis in order to improve the Business Intelligence of a CDN slice and understand more our customer needs and expectations. |
| CDN-Cache | The server that cache content. It is considered as the Content Publisher (CP) in the ICN slice. | **Caching service**<br>• Cache static content and stores uploaded videos by users.<br>• When a user requests to watch a video with a desired resolution, the nearest edge server to the user will deliver the content, ensuring the shortest distance, therefore reducing the latency and providing the best QoS possible. |

**Table 3 – ICN Slice components**

| Components name | Description | Major functionalities |
|---|---|---|
| Dynamic NDNGateway | Enables the interaction between the Dynamic CDN slice and ICN slice | • NDN Gateway is assigned dynamically at each ICN slice based on distance either to the CDN cache or client<br>• The ICN node that has been elected as gateway is assigned higher virtual resources via Orchestrator's provisioning function<br>• Provision of the protocol translation function between ICN and IP when needed.<br>• The function which reads out the content cache in CDN server then reformats the content and transmits it chunk-by-chunk |
| NDN node | Network node with full function of ICN protocol so that the forwarding, caching functions and data exchange with content repository are handled in a manner that satisfies network resource, configuration, policies, etc. | • Utilize PIT (Pending Interest Table) and FIB (Forward Information Base) as the fundamental data structures for forwarding process<br>• Store the cached content in CS (Content Store)<br>• Request aggregation function: ICN nodes are equipped with the function to aggregate requests to the same content objects to reduce network traffic and server load.<br>• Subscription: ICN node provides a mechanism for a consumer to register a name to identify one or more content object in which the consumer is interested. |
| NDN node components (three fundamental data structures) | PIT (Pending Interest Table): records retrieval path and aggregates requests of the same content | • Indicate the return path of the requested content<br>• Request aggregation function when the same content request arrives |
| | FIB (Forward Information Base): forwards requests based on content name | • Name based forwarding |
| | CS (Content Store): acts as the cache storage of the ICN node to reduce the duplicated traffic of the same content objects, and also shorten the response time | • Cache the contents which went through the node, and when the request to the cached content, the content is provided by the content store of the node |

| | | |
|---|---|---|
| UE service module | Provides the content service module for UE | In the user equipment, the following set of functions is requested connected to each other to provide the user with the content with ease.<br>• IP based Dynamic CDN access function<br>• Light NDN protocol functions<br>• Viewer function of video content |
| ICN control function | Ensures that the key functions of ICN: routing, naming scheme and mobility support are controlled in an effective way. | • Content object registration function:<br>A content object is registered with a unique name or ID in ICN so that the consumers can access to it.<br>• Content object availability function: availability information of content objects is disseminated to help choosing a right direction of request forwarding.<br>• Network selection function: appropriate network interfaces are selected to forward requests in order to reach a specified content object.<br>• Content cache function: ICN nodes are equipped with content cache to reduce duplicated traffic of the same content objects. |
| ICN Management function | Responsible for network management functions | • Handles the key network management functions, such as: configuration, network performance, QoS, congestion control, fault tolerance managements. |
| Security function<br>(optional) | Ensures that the efficient secure mechanisms, including: availability, authentication and integrity function are enabled. | • Access control: ICN is equipped with a mechanism to examine and confirm the authenticity of consumers and that content object is accessible only by the authorized consumers.<br>• Network security function from malicious attack:<br>ICN has a mechanism to protect its functions from malicious network attacks.<br>• Content object availability<br>ICN provides a mechanism to ensure that the content objects published in network are available for authorized consumers.<br>• Content authentication and integrity:<br>ICN equipped with a mechanism to examine and confirm the authenticity and integrity of content objects. |

### *2.1.3. ICN/CDN Component interface*

The interfaces among the components are listed in Table 4.

**Table 4 ICN/CDN component interface**

| Components | Protocol | Major functionalities |
|---|---|---|
| Orchestrator Web server/OrchestratorDB | HTTP REST<br>MySQL database | • Interface the subscribers<br>• Populate database |
| VIM agent | Cloud provider API:<br>- Amazon<br>- OpenStack<br>- (MS azure)<br>- (Rackspace) | • Interface the administrative cloud domains<br>• Resource management<br>• Dashboard for available cloud domains |
| Queuing Server | Advanced Messaging Queuing Protocol (AMQP): RabbitMQ server<br>Socket based | • A server for each CDN slice<br>• Server manage all the queuing and load balance the tasks over the all network function nodes |
| Southern API with the slice Coordinator | AMQP<br>IP | • One tunnel queue between the Orchestrator and each Coord. |
| Coordinator Web Interface/CoordinatorDB | HTTP REST<br>MySQL database | • Interface the slice owner and the end users. |
| Streaming agent | Nginx server<br>AMQP | • Communicate with streamer nodes |
| Transcoding agent | AMQP<br>SSH | • Communicate with transcoder nodes |
| Northern API with Orchestrator | AMQP<br>IP | • Acknowledgement when changes applied in respective slice. |
| Caching service | Nginx server | • Store contents |
| Transcoding service | FFMPEG | • Transocde videos to different resolutions |
| Streaming service | Nginx | • Stream contents to end users |
| Access Reporting service | SSH protocol | • Populate the Coordinator database regarding all access information.<br>• The database will be used for future data analysis and business intelligence including data analysis, content popularity, VNF placement and such other smart applications. |
| Network interface among ICN nodes | ICN (NDNx protocol) | • ICN slice provides the integrated appropriate interfaces among ICN node for distributing content (name data objects) in ICN interconnections via in-network caching |

| | | |
|---|---|---|
| | | mechanism as a base of network protocol for large-scale |
| UE-NDN edge node interface | ICN | • Supports APIs for NDO distribution and retrieval, enable applications to retrieve the valid information and response to the request accordingly. |
| UE-coordinator interface | IP | • Suitable interfaces transparent for end-user are selected to guarantee efficient network connection for users accessing the networks<br>• Enable applications to retrieve the valid information and response to the request accordingly. |
| Orchestrator-NDN GW interface | ICN | • Provide best-fitted interfaces for the access network, as well as managing the virtual network and resources in ICN |
| Orchestrator-coordinator interface | IP | • Interact with CDN slice via its coordinator |

## *2.1.4. Initial Evaluation*

### 2.1.4.1 Scope of the evaluation

In this use case scenario, several new aspects are required to be implemented, such as the new VNF realization, the introduction of emerging network architecture (ICN), slice stitching between CDN slice and ICN slice, and the implementation of slice on FLARE architecture such as the deep data plane programmability. Therefore, it is necessary to take a step by step approach toward the final implementation of this use case scenario on the integrated test bed, starting with the development and evaluation of each VNF, partial interconnection test by manual operation, and finally the fully automated operation. The initial evaluation intended to validate the first part of the stepwise implementation.

### 2.1.4.2 Evaluation Criteria

As the first step, the evaluation is focused on the functional check of the components and the following aspects were intended for evaluation:

- VNF developments and evaluation.

In the CDN slice,

- CDN coordinator, Contents cache, Transcoder, and streamer.

In the ICN slice,

- ICN node and ICN Gateway as Docker image for FLARE.
- Slice creation by manual operation.
- First stage inter-slice connection.
- Control sequence creation.

### 2.1.4.3 Expected results

It was expected that the activities bring the following results:

- Basic VNFs are ready for use.

- Basic slice operation is verified.
- Simple inter slice connection mechanism is tested.
- Total sequence diagram is designed.

#### 2.1.4.4 Experimentation Results and Evaluation

Following is the results of the initial evaluation:

**Table 5 Experimentation steps and its evaluation results**

| Steps | Test sequence | Verdict | |
|---|---|---|---|
| | | Pass | Fail |
| 1 | For the CDN slice, the key VNFs are developed as Open Stack images, and the CDN slice is created by connecting developed VNFs. The NFVs are created in Data Centres (DCs) in several geometric areas and use IP networking. The video content delivery experiment was done based on the CDN slice scenario, and it is verified that the intended operation is achieved. | X | |
| 2 | For the ICN slice, the NDN node Docker image is developed as a preparation of the first stage implementation on FLARE architecture. Developed NDN nodes are interconnected through Ethernet, and the correct exchange of packets is observed. By this test, fundamental functional operation of the ICN slice is achieved. | X | |
| 3 | As for the NDN gateway, the data format conversion part is developed as an Open stack instance aiming at the initial inter-slice connection check. | X | |
| 4 | Inter slice connection mechanism has been tested and evaluated: the Open stack server is set up in Waseda University for the use of CDN slice by Aalto University. NDN nodes and gateway are also implemented on the same server. Aalto University remotely implemented necessary VNFs to provide the CDN service. The video content is uploaded on the CDN cache VNF. Then, the NDN gateway retrieved the contents from the CDN Cache. The packet format conversion is performed on the NDN gateway, to make the contents into the format (Data chunk) that can be handled in the ICN slice. By sending the content request to the ICN slice, the intended content is successfully provided to the requester. The experiment is done from Aalto university across the continent, and the result is presented in the 5G!Pagoda face to face meeting in September 2017 at Aalto university which confirmed the functional feasibility of the NDN/CDN combined content delivery service scenario. | X | |

| | | | |
|---|---|---|---|
| 5 | A total sequence diagram is designed for the final stage system integration enabling the automated CDN and NDN slice creation, and the inter slice connection under the control of the orchestrator. Also the interaction of user and the NDN/CDN combined content delivery system sequence is developed for the delivery of content on user request. | X | |

### *2.1.5. Conclusions*

As the first step of the implementation of the ICN/CDN combined content delivery scenario, the basic part of each slice is successfully implemented, and also the initial trial to connect both slices is done. With these activities, it can be said that the feasibility of the proposed service scenario is shown. The research and implementation will be continued until the final implementation. The followings are the major action items for the further development:

- Automation of the total procedures by the binding with Orchestrator.
- Implementation of the ICN slice on FLARE platform.
- Development of UE functions.
- Enhancement of NDN gateway functions.
- Automated collaboration between CDN slice and ICN slice.
- Interaction with the orchestrator.

## 2.2. Ignition Testbed 2 – IoT and Video Slices

DG, ERICSSON, MI and FOKUS built an ignition testbed for the use case of IoT and Video slices that shows the impact of dynamic network slicing of 5G!Pagoda in an emergency case in an IoT site. The selected scenario for the testbed is to handle an emergency situation in IoT enabled buildings such as factories, data centres, etc. The UE is able to connect to multiple slices and request the creation of a slice with increased capacity in case of alarm. The trigger for creating a slice can come manually from the user or automatically based on input from an IoT application.

Dynamic slicing allows a slice to be created without manual configuration. This allows slices to be easily set up for various purposes, such as specific applications, services, events and situations. The orchestrator exposes an interface through which users or services can request a slice to be created. The scenario is to demonstrate the benefits of the dynamic slicing mechanism in order to automatically set up a slice in reaction to an event from an IoT system. The other demonstrated feature is the allocation of resources such as bandwidth to specific slices. Slices have a set of resources allocated. Additionally, slices may have priority in case there is a lack of resources in the network. A high priority slice may obtain its required resources from a lower priority slice.

This ignition testbed and the use case have been demonstrated and validated during the IoT Week 2017 in Geneva. In the demonstrating scenario, it showed the reactions to a fire alarm in a data center caused by artificially heating up a temperature sensor. The rise of temperature measured by sensors in a data center triggers an alert via the IoT gateway to the IoT application server, which requests a new slice from the orchestration service. The new slice is set up with dedicated services and a reserved bandwidth for video streaming, allowing the critical event to be observed with high quality. When the fire has been controlled,

the slice is released. In addition, this scenario aims to demonstrate the Lightweight Control Plane (LCP) for slice customization, that represents the minimal set of functionalities of the core network needed to run the current 5G scenarios, as addressed by WP3.

### *2.2.1. Architecture Description*

The architecture is composed by vertical and horizontal dimensions, being different slices and data centers, respectively. As shown in Figure 4, functions are deployed between central and edge clouds where different types of UEs can connect, while the horizontal layers are characterized by a common slice that groups the shared/common components and by other specific slices such as the on-demand video slice, the IoT slice, the mIoT slice and the streaming video slice that will be detailed in this section.

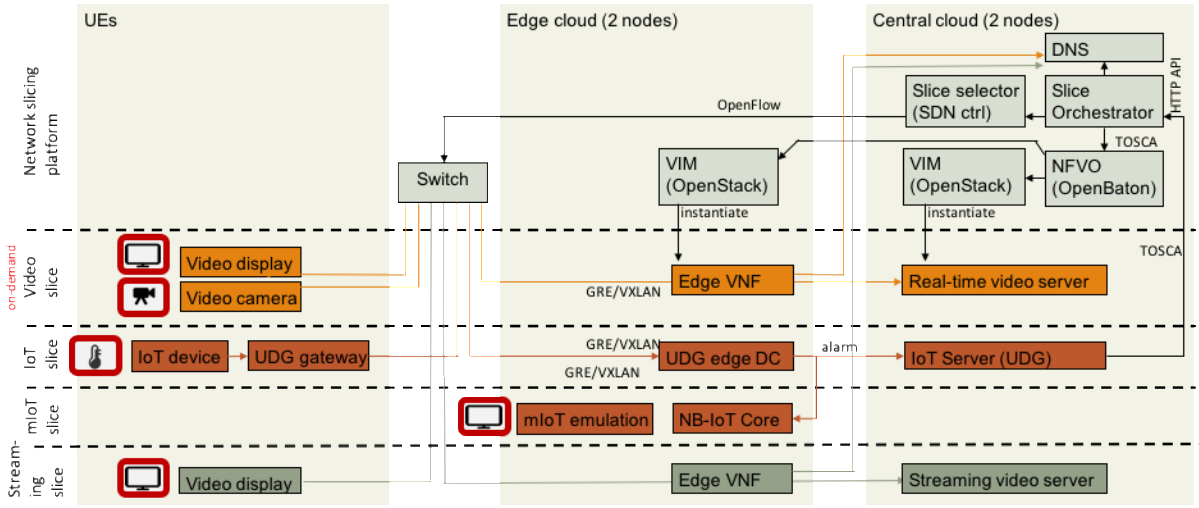


**Figure 4 Components building blocks for the IoT use case**

**Network slicing platform**

The network slicing platform contains the NFVs shared among all slices. It contains the infrastructure components implemented by Ericsson. These are either virtualized components running as Virtual Machines (VMs) or in Docker containers. The components include the Slice Orchestrator, the Slice Selector, the NFV Orchestrator (NFVO) and one Virtual Infrastructure Manager (VIM) in each cloud. Additionally, it hosts a DNS server to which the Slice Orchestrator registers the IP addresses of each publicly accessible VNF so that they can be located by the client. The functions of the components are presented in Table 6.

**Streaming Video Slice**

The streaming video slice provides a video service to which users can connect to in order to watch on-demand videos from a video server.

The video server is implemented as a web server that serves video files pre-encoded in various bit-rates from 216 kbps to 582 kbps. The files are served via the MPEG Dynamic Adaptive Steaming over HTTP (DASH) protocol which allows a client to request a suitable video bit-rate according to the available bandwidth. To allow clients to switch bit-rate during playback, the video files are split into 2 second long segments.

The client connects to the video server via the MPEG-DASH protocol which allows the client to adapt the bit-rate according to the available bandwidth. As a client we use the DASH Industry Forum Reference Client, which uses the reference algorithms for bit-rate adjustment according to the playback buffer length. In addition to the video stream, the client also displays the buffer length and the bit rate as a continuously

updated graph. The client runs in a Linux based laptop connected to the slice via Wifi. The client locates the video server using DNS, to which the video service VNF is registered by the Slice Orchestrator on instantiation.

The streaming video slice has a lower priority than the real-time video slice. It utilizes the remaining bandwidth with a low minimum guaranteed bandwidth. The purpose is to represent the best effort traffic in the network that utilizes the remaining of the available bandwidth. Using a video stream for this purpose allows visualization of the impact of a higher priority slice on the other traffic, since the available bandwidth is observed (indirectly) as both the visual quality and the video bit-rate graph on the client.

The slice consists of two VNFs. The Edge VNF located in the edge cloud is responsible for interfacing the clients to the slice and provide the networking for the clients. In this case, the client addresses are connected to the network via a Network Address Translator (NAT). The Edge VNF also contains a Slice Agent that is responsible for registering the slice and controlling the QoS. The Video Service VNF is located in the core network cloud.

**Real-time Video Slice**

The real-time video slice connects two devices, a camera device and a display device, via a video server. It allows the devices to locate each other and the video stream to be transported between the devices.

The video server is implemented using VLC, which reads the video stream from the camera and re-encodes the stream and may perform protocol conversion (depending on the used camera device). The video server then provides the stream over a HTTP server to which multiple clients can connect. The server provides the video over a high-quality fixed bit-rate stream, which consumes a high bandwidth. The display client connects to the video server with HTTP. The client runs in a Linux based laptop connected to the slice via Wi-Fi. The client locates the video server using DNS, to which the video service VNF is registered by the Slice Orchestrator on instantiation.

When real-time video slice is created, bandwidth is allocated to it at the cost of bandwidth taken from lower-priority slices. The real-time video slice has a higher priority than the streaming video slice, thus, the real-time video will display with full quality while the streaming video will have its quality reduced.

The slice consists of two VNFs. The Edge VNF connects the clients to the slice and provides the networking for the clients. Also in this case, the client addresses are connected to the network via a NAT and a Slice Agent is responsible for registering the slice and controlling the QoS. The video server VNF is located in the core network cloud while the edge VNF is located in the edge cloud.

**IoT Slice**

A specific slice dedicated for IoT devices is put in place through the switch where UDG (Universal Device Gateway) from Device Gateway is connected. This slice is in fact a Virtual Extensible Local Area Network (VXLAN) which is similar as a VLAN but dynamically managed by OpenFlow in the frame of SDN. So this slice is completely independent and isolated from the other network slices. In the IoT slice, there is an instance of UDG obtaining a local IPv4 address from OpenFlow. A border router is also connected to UDG through a USB port and represents the gateway between the IoT slice and the wireless sensor network. As the protocols used by the temperature wireless sensor are in this scenario IEEE 802.15.4 and 6LoWPAN, the wireless sensor network is deployed on IPv6 with global IPv6 addresses which guarantee an end-to-end connectivity. After receiving the temperature provided by the wireless sensor, UDG sends the requested alarm to the mIoT slice using UDP and LWM2M. A TOSCA file is also sent by UDG to the slice orchestrator. Finally, the TOSCA file is transmitted to Open Baton. All IoT devices (sensors, actuators, gateways, etc.) added in the IoT slice obtain IPv4 addresses corresponding to the IoT slice defined by OpenFlow.

**mIoT Slice**

The massive Internet of Things use case is one of the key scenarios for 5G. Consequently, the mIoT slice has been built following the requirements to support a very large number of devices in a small area, therefore, a very large device density. To achieve this goal, the Fraunhofer Open5GCore NB-IoT extension has been adopted as part of the slice, being the first implementation of the essential 3GPP NB-IoT features (Release 13 – TS 23.682) and enabling the demonstration of low energy IoT communication. It addresses the current stringent needs of the mIoT use case to provide low power, low cost efficient communication for a massive number of devices.

Furthermore, the NB-IoT Core Network extension feature represents the first version prototype of the Lightweight Control Plane designed to run as a Common Sub-Slice (according to the architecture defined in WP2) and mainly focused on reducing the complexity of the overall core network architecture: network functions are reduced to a minimal set of functionalities in terms of mobility, session, QoS, etc. in order to create a lightweight version of the core network. Only the skeleton of the Open5GCore remains with the only addition of the specific C-IoT extension for providing minimal data connectivity over control plane to IoT sensors. Furthermore, the application server represented by the OMA LWM2M (Lightweight M2M) Server is connected to the SCEF component, exploiting its capability of exposing 3GPP functions to third parties' components.

UEs are emulated through a benchmarking tool capable of generating workloads and to acquire and process resulting data. It was designed to assess the performance of a virtualized packet core solution for a massive number of subscribers (up to 10.000) and different number of eNBs, both LTE and NB-IoT types of communication. Furthermore, the UDG gateway placed in the IoT slice registers as a real device to the Open5GCore and is able to forward the alarm notification also to the core network that will then notify the LWM2M server of the emergency situation.

### *2.2.2. Components description*

The components involved in this use case are described in Table 6 **Error! Reference source not found.**and Table 7. **Error! Reference source not found.**Table 7

**Table 6 Platform components in common slice**

| Components name | Description | Major functionalities |
|---|---|---|
| Slice Orchestrator | Orchestrates the slices across several domains (in this case edge and core domains). | • Provides an API towards slice user for dynamically creating a slice<br>• Determines in which domain (here in edge and core) the components should be placed<br>• Interfaces one or several NFVOs (here one) for setting up the components in data centres<br>• Interfaces the slice selector for registration of the slice<br>• Interfaces DNS for registration of services accessible to slice users<br>• Controls tunnels toward Edge VNFs<br>• Provides scalability by multiple parallel orchestration instances |

| | | |
|---|---|---|
| Slice Selector | Controls the connectivity from UEs to slices | • Defines the connectivity rules between UEs and Edge VNFs<br>• Implements an SDN controller for communicating the connectivity rules to access network switches using OpenFlow<br>• Interfaces the UE database for retrieving information about slice access rights<br>• Controls bandwidth allocations<br>• Provides an API for UEs to select their slice |
| NFVO | NFV Orchestrator as defined by ETSI NFV. In this testbed Open Baton is used. | • Controls the orchestration of NFVs<br>• Sets up a slice consisting of multiple NFVs using one or several VIMs<br>• Manages the life cycle of the NFVs<br>• Defines the networks internal to a VIM |
| VIM | Virtualized Infrastructure Manager as defined by ETSI NFV. In this testbed OpenStack is used. | • Manages the VMs on which NFVs run<br>• Manages the internal network between VMs<br>• Distributes the VMs across several physical machines |
| Access Network Switch | Virtual switch in the access network. In this testbed Open vSwitch is used. | • Routes traffic between the UE and the edge VNF<br>• Implements the rules defined by the slice orchestrator<br>• Enforces bandwidth allocations<br>• Terminates tunnels toward edge VNF |
| Edge VNFs | The NFV within the slice that interfaces the UEs | • Contains a virtual switch (Open vSwitch)<br>• Terminates tunnels toward edge switches<br>• Implements the routing toward slice services, in the prototype it implements network address translation between the UE addresses and the service address space<br>• Enforces bandwidth allocations<br>• Registers its information (e.g. addresses) to slice selector |

**Table 7 Application components in the service slices**

| Components name | Description | Major functionalities |
|---|---|---|
| IoT device<br>(temperature sensors) | Wireless temperature sensor | • This mote is continuously measuring the temperature in a room. |

| | | |
|---|---|---|
| | | • The communication protocols used by this mote are IEEE 802.15.4 and 6LoWPAN. |
| UDG gateway | IoT gateway and trigger for NFV | • Collects the temperature data provided by the wireless sensor nodes<br>• When the temperature is higher than the defined threshold, UDG sends an alert to the LwM2M server and NFVO. |
| DNS server | Provides names for services in the slices. In this testbed BIND is used. | • Receives dynamic DNS update from slice orchestrator<br>• Allows UEs to query IP addresses based on service names |
| UDG Edge Data Centre | Lightweight Computers to form a Portable Openstack cluster. | • Provides Virtual environment for the other components included in the Edge Cloud and Data centre. It consists of 2 set of Intel Nuc Minicomputer configured in an OpenStack cluster. |
| Streaming video server | Server providing streaming video | • Serves video streams over MPEG DASH |
| Real-time video server | Server providing real-time video | • Allows clients to connect in order to watch the video streams from the camera<br>• Video protocol conversion |
| mIoT emulation | NB-IoT UEs and base stations emulation | • Addresses the connectivity of a multitude of devices (up to 1000)<br>• Capable of generating workloads for a massive number of IoT emulated devices |
| NB-IoT Core | Software based NB-IoT Core Network | • Fully virtualized core network for the 5G environment<br>• Support for NB-IoT, both IP and NON-IP delivery solutions<br>• Service Capability Exposure Function (SCEF) provides the API for LW |
| Lwm2m server | Application server | • Provides device management functionality over IoT or cellular networks<br>• Allows the registration of multiple groups of devices with pre-provisioned management objects: Device, Temperature Sensor, Transport Management Policy, etc. |

### *2.2.3. Component interfaces*

The following table shows the components interfaces for this ignition testbed.

**Table 8 Component interfaces**

| Components | Protocol | Major functionalities |
|---|---|---|
| Slice orchestrator – NFVO | HTTP REST | • Creating network service description<br>• Uploading TOSCA<br>• Creating network service record (instantiating)<br>• Retrieving status information and configuration<br>• Removing network service record and network service description |
| Slice orchestrator – slice selector | HTTP REST | • Registration of slice |
| Slice orchestrator – DNS server | DNS | • Registration of network services accessible to UEs |
| Slice orchestrator – Access network switch | OVS-DB | • Control of tunnels between access network switch and Edge VNF |
| Slice selector – Access network switch | OpenFlow | • Definition of forwarding rules |
| Slice selector – UE | HTTP REST | • Registration of UE |
| DNS server – UE | DNS | • Lookup of network services available to UEs |
| NFVO – VIM | HTTP REST | • Instantiation of VMs<br>• Instantiation of virtual networks<br>• Management of images<br>• Resource management<br>• QoS management |
| Edge VNF – Slice selector | HTTP REST | • Registration of detailed slice information |
| Edge VNF – Access network switch | OVS-DB | • Control of tunnels between access network switch and Edge VNF |
| Access network switch – UE | IP | • User plane traffic |
| Access network switch – Edge VNF | IP over VXLAN/GRE | • User plane traffic |
| Streaming video server – UE | MPEG DASH | • User plane traffic (video) |
| Real-time video server – UE | MPEG4 | • User plane traffic to display (video) |

| | | |
|---|---|---|
| Real-time video server – UE | RTSP | • User plane traffic from camera (video) |
| IoT server (UDG) – Slice Orchestrator | HTTP REST | • Instantiation of new slice (TOSCA file, containing the required information for the management of the video flow)<br>• Removal of slice |
| UDG gateway – LwM2M server | LwM2M | • UDG transfers the sensor data into LwM2M format and sends the LwM2M payload to LwM2M server |
| Sensor – Border router - UDG Gateway | IEEE 802.15.4<br>6LoWPAN | • Sensor data transmission |
| Lwm2m server | UDP | • User plane traffic from/to NB-IoT UEs<br>• Alarm signal from UDG gateway |
| NB-IoT Core<br>- Service Capability Exposure Function (SCEF) | UDP | • Delivery of NON-IP data from/to the AS<br>• Exports functionalities of the core network towards the AS |
| NB-IoT Core<br>- MME | S1AP/NAS | • S1AP signalling service from eNBs<br>• Forwarding of NON-IP data packets after NAS decapsulation |
| mIoT emulator<br>- NB-IoT base stations | S1AP | • Signalling service to the MME<br>• Carrier of NAS user plane traffic from/to NB-IoT UEs |
| mIoT emulator<br>- NB-IoT UEs | NAS | • Small amount of user data (Kb) is encapsulated into NAS messages and sent over control plane |

## *2.2.4. Initial Evaluation*

### 2.2.4.1 Scope of the evaluation

In the ignition testbed, the evaluation is mainly performed as functional evaluation. As this use case demonstrates a dynamic slice creation for a real-time video service triggered by an emergency situation from the IoT slice dedicated to IoT services, the functional evaluation focuses on:

- Functional check of the configuration of the all network components.
- Delivery of a new service request from the IoT slice in its emergency situation.
- Success of dynamic slice creation.
- Functional check on slice prioritization.

- Functional check on slice selection and quality control for the real-time video slice.
- Functional check on removal of slice when the emergency situation has been solved.

#### 2.2.4.2 Evaluation Criteria

The initial functional evaluation is measured by two types of criteria: Functional criteria for checking the behaviours and data exchanges of the components and non-Functional criteria that contains quantitative measurement for demonstrating the quality of the systems and qualitative measurement from the user experience. The following Table 9 and Table 10 show the defined measurement criteria.

**Table 9 Functional Criteria**

| # | Criteria | Description | Realization in testbed |
|---|---|---|---|
| 1 | Component configuration | All involving components must be configured and verified the functionalities. | Test messages are sending for the configuration check and verification of their communication among all involving components in the initial slices (IoT slice components, mIoT slice components, video streaming components, core platform components, and edge VNF components). |
| 2 | Slice configuration | All involving slices must be configured followed by the initial slice policy. | Slice orchestrator sets up initial slice policy and all involving UE slices are ready and running. |
| 3 | Emergency detection | The emergency on IoT testbed must be immediately delivered to IoT server. | The UDG IoT gateway sends alarm signals to the UDG IoT server and the NB-IoT core. |
| 4 | Emergency handling 1 | The IoT server should inform the alarm situation to the Slice Orchestrator. | The UDG IoT server sends slice creation request (detailed defined in TOSCA format) to the Slice Orchestrator. |
| 5 | Emergency handling 2 | NB-IoT core should maintain network connectivity and indicate the situation to the users using mIoT services. | NB-IoT core receives the emergency alarm and indicates the situation to the users using mIoT services. |
| 6 | Slice description | The user should be able to describe the functions of the slice through a slice descriptor. | The VNFs of a slice and their relationships are described in the TOSCA modelling language. |

| | | | |
|---|---|---|---|
| 7 | Dynamic slice creation | A slice can be created based on the provided slice description | The slice description provided via the API will be set up based. |
| 8 | Edge placement | VNFs can be placed in either the core or the edge cloud | The edge VNF is placed in the edge cloud while the server VNFs are placed in the core network cloud. |
| 9 | Traffic isolation | Traffic should be isolated between slices | It is not possible to reach a device in one slice from a device in another slice directly without connectivity setup. |
| 10 | Bandwidth control | The minimum available bandwidth should be specified and guaranteed for each slice | Each slice gets the requested bandwidth allocation provided that the capacity of the system is not exceeded. |
| 11 | Slice prioritization | Slice priority should be observed if multiple slices compete for the same resources | If the capacity requested by the two video slices exceeds the available capacity, the higher priority slice (the real-time video slice) is assigned the requested capacity while the lower-priority slice (the streaming video slice) gets the remaining capacity. |
| 12 | Slice selection | It should be possible to specify for each user device to which slice it should be connected | The slice selector provides an API to which the slice assignment of a client device is specified. |
| 13 | Slice removal | The dynamically created slice for the emergency situation should be removed and the streaming video service should return to the normal service. | The UDG IoT gateway sends a slice removal request with NSR ID to the Slice Orchestrator.<br>The Slice Orchestrator removes the slice and corresponding resources. |

**Table 10 Non-Functional Criteria**

| # | Criteria | Description | Realization in testbed |
|---|---|---|---|
| 14 | Slice setup time | The slice setup time is feasible for the purpose of dynamical setup of slices. | The setup time of a high-quality slice for remote emergency supervision is less than 2 minutes. |
| 15 | Video quality | The user should experience smooth video transmission for | Related to the Criteria 8, the high-priority real-time video slice is |

| | | | |
|---|---|---|---|
| | | the real-time video from the emergency place. | assigned the requested bandwidth capacity and the user's experience should match on it. |

#### 2.2.4.3 Expected results

It was expected that all components were ready to use and APIs among components are correctly functioning for handling the use case that requires low-delay high-quality real-time service.

#### 2.2.4.4 Experimentation Results and Evaluation

The experimentation has been made step-by-step following the described scenario and Table 11 summarizes the major steps of the demonstration and its functional evaluation results.

**Table 11 Experimentation steps and its evaluation results**

| | | | |
|---|---|---|---|
| **Simplified Inter-component** | Sensors -> IoT gateway -> IoT server -> Slice orchestrator -> NFVO -> Slice selector -> Edge VNF | | |
| **Test purpose** | Ensure that {<br>When {the slice orchestrator receives a request from IoT server for its emergency situation}<br>Then {The slice orchestrator creates a slice for real-time video transmission based on the predefined SLA}<br>When {the emergency situation has been handled}<br>Then {The IoT server sends a slice removal request to the slice orchestrator}<br>Then {The slice orchestrator removes the dynamically created slice.} | | |
| Pre-test condition | Slice creation policy, Slice prioritization policy | Trigger | Temperature sensing value |

| **Step** | **Test sequence** | **Verdict** | |
|---|---|---|---|
| | | Pass | Fail |
| 1 | The platform components and edge components (Slice orchestrator, slice selector, NFVO, Edge VNF, DNS server, network switches, etc.) and APIs are configured and ready (criteria 1). | X | |
| 2 | For maintaining IoT slice, sensors, the IoT gateway, IoT server, edge DNS, edge Data centre and border router are configured and connection with NB-IoT core and slice orchestrator are tested (criteria 1). | X | |
| 3 | For the lightweight network signalling support in mIoT slice, UEs perform simplified signalling procedures in order to attach, detach, etc to the core | X | |

| | | | |
|---|---|---|---|
| | network (specified in 3GPP TS 23.682) and provides non-IP data delivery during the segmentation process of the control part of the plane (criteria 1). | | |
| 4 | The slice orchestrator configures UEs' slices with initial slice policy and the UEs' slices are set up and ready (criteria 2). | X | |
| 5 | The IoT gateway detects abnormal temperature of the building and sends an emergency alarm to the IoT server and NB-ioT core (criteria 3) | X | |
| 6 | The IoT server sends a slice creation request (defined in TOSCA format) to the slice orchestrator (criteria 4 and 6). | X | |
| 7 | The NB-IoT core receives the alarm signal from the IoT gateway and indicates it to the User GUI (criteria 5). | X | |
| 8 | The slice orchestrator receives the slice description in TOSCA format from the IoT server (criteria 7). | X | |
| 9 | The slice orchestrator dynamically sets up a network slice by interacting with the NFVO (Open Baton), the SDN controller and the DNS server (criteria 6, 9 and 12). | X | |
| 10 | The NFVO interacts with the VIMs (OpenStack) both in the edge cloud and the core cloud and instantiates VNFs on both clouds (criteria 8). | X | |
| 11 | There are pre-defined Service Level Agreement (SLA) in bandwidth allocation for each service. Slice orchestrator applies the SLA to each slice within its bandwidth capacity. The dynamically created Real-time video slice for the emergency situation has high priority and the video streaming service is based on the best effort service. The video quality of the two services in different slice demonstrates the different video quality (criteria 10, 11 and 15) | X | |
| 12 | The IoT server receives the end of the alarm situation and sends a slice removal request to the slice orchestrator. The slice orchestrator removes the slice and the related resources (criteria 13). | X | |
| 13 | As the bandwidth reserved for the real-time video transmission has been released, the video quality for the video streaming service increases (criteria 10, 11 and 15). | X | |

The slice setup time has been measured for the evaluation of the Criteria 14 as shown in the Figure 5. In case of bare Ubuntu image, the required software for edge and core VNF are installed when a new network slice is setup dynamically. For pre-built case, the image used for VNFs is already installed with all required software for a specific VNF. To obtain the time difference between slice set-ups with the two VNF images, slice setup times were recorded for both the pre-built and bare VNF image cases. The resulting cumulative distribution function (CDF) of slice setup in the two cases is shown in Figure 5. It can be seen from the figure that slice setup time with pre-built VNF image take less time to start a new network service as compared to a VNF launched with bare Ubuntu image. On the average, it takes around 41.05 seconds to successfully

launch a network service using pre-built VNF image. While in case of VNF with bare Ubuntu image, the average slice setup time is 101.64 seconds, a difference of more than 50%.

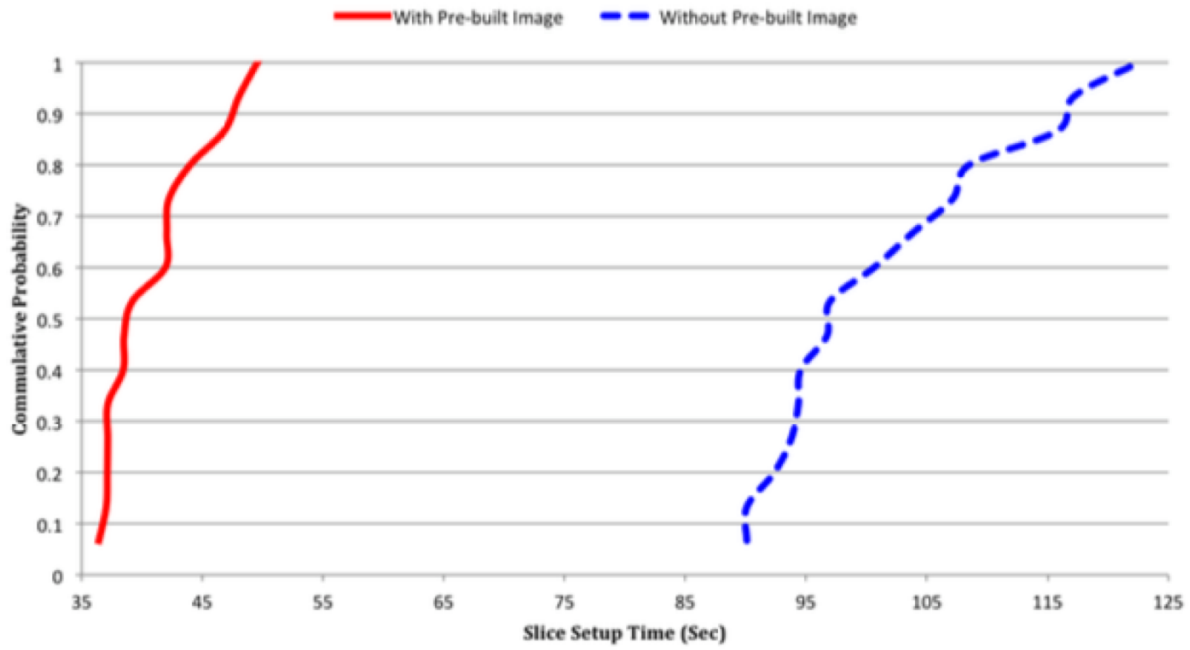


**Figure 5 Slice setup time with pre-built image and without pre-built image.**

### *2.2.5. Conclusions*

The first iteration of the dynamic slice creation for the IoT and video slices has been implemented and successfully demonstrated. The dynamic slice creation by a user request can bring new opportunities for opening new services in UE slice owners and telecom operators. It continues the development and enhancement of the scenarios and functions in both the IoT server and slice orchestrator, and will add more detailed evaluation measurement results (e.g., detailed video QoS measurement). It will ensure that the implementation is aligning with the architecture and functionalities defined in WP2, WP3 and WP4.

## 2.3. Ignition Testbed 3 – Healthcare

NESIC, UT, Hitachi and KDDI develop this ignition testbed. This use case scenario aims to realize an efficient slice creation and modification by service requests. It is targeted to generate a slice through Hitachi's orchestrator on the University of Tokyo's FLARE platform and to realize a change of slice characteristics. The MVNO market trend in Japan within which MVNO tries to offer the dedicated services to various industries was stated in D2.1. In this use case, we focus on a case where MVNO provides a healthcare service and there are two key points:

The first point is to create slices dedicated to services. It is necessary to define the end-to-end communication quality. For example, slices are created by freely combining resources permitted to control from the MNO. It is possible that this MMO may become plural in reality. For MVNO that needs to realize various services among limited resources, the efficient use of resources is crucial, and it is necessary to efficiently utilize resources and form slices.

The second point is to combine resources provided by MNO and resources owned by MVNO to aim for an appropriate slice creation for services. In this scenario, in order to realize the security function,

indispensable for the realization of the health care service, we aim to arrange and operate the security NFV prepared by MVNO on the slice we form. In reality, NFV installed by MVNO should be able to operate on various points, such as edge to core.

### *2.3.1. Architecture description*

In this use case, we will consider mobile terminals used for healthcare applications. In Japan, the mechanisms that enable to access medical information remotely have begun to be adopted against the background of telemedicine and government-initiated reforms of working methods. For example, it is used to confirm electronic charts from a mobile terminal, collaborate with a nurse call system, confirm patient's vital information, and grasp the state of a patient from a distance. A security function is essential in these types of services in order to protect the personal information of the patients and the confidential information related to life. Thus, the security NFV establishing its security function from the network channel can strengthen the authentication. Terminals, SIMs and applications are targeted as authentication elements.

There are two points of this architecture. The first is to create an end-to-end slice from MVNO. In the current architecture, MVNO can only control MVNO equipment. In order to form an end-to-end network slice, suitable for each service, it is necessary to control the facilities managed by MNO as well. The second one is that it makes it possible to allocate a service dedicated NFV not only on the resources that MVNO holds, but also on resources provided by MNO and cloud service provider. We aim to be able to choose the location of NFV that is suitable for the characteristics of the service. Therefore, a programmable network node is required, and FLARE platform is used to provide programmable network functionalities.

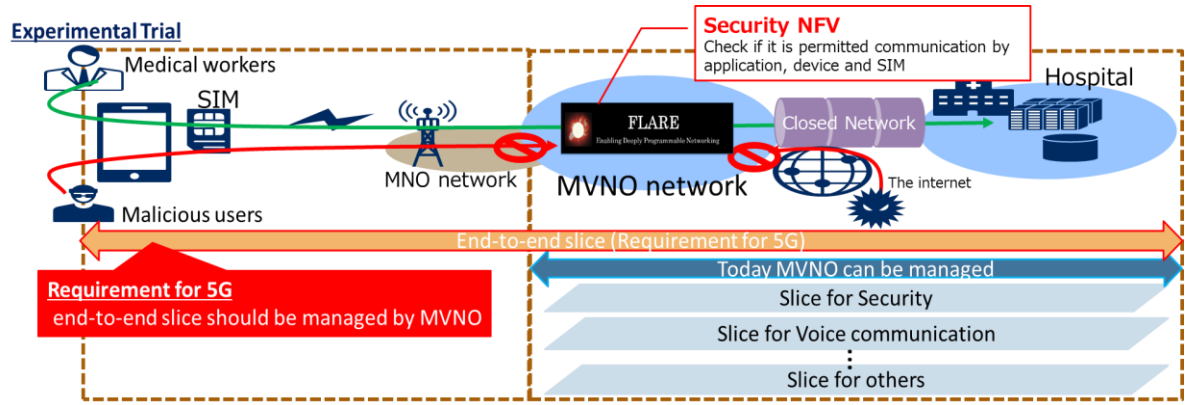


**Figure 6 – Security NFV**

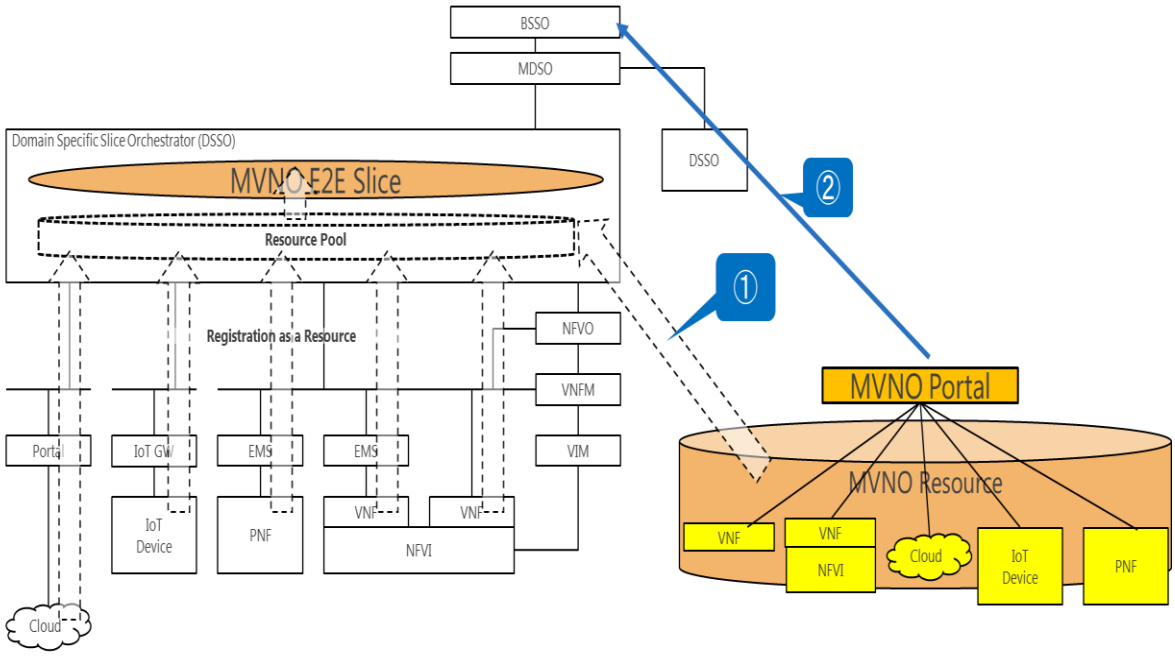


**Figure 7 – System Architecture from MVNO's viewpoint**

In order that the MVNO controls the API provided by the orchestrator from the MVNO's portal, the following methods are used:

(1) Register the resources owned by MVNO / MVNE in the resource pool on the DSSO. At the registration stage, the ranges of the open resources to the others are defined for each resource. Resources defined as open resources can be assigned to other MNOs and MVNOs.

(2) Request the resources necessary for the service deployment, network slice, and SLA for network slice from the portal held by MVNO to the BSSO. Appropriate resources are selected from the resource pools and those are provided for MVNO through MNO, MVNE, etc., and the slices for which MVNO secured the end-to-end SLA are generated.

### *2.3.2. Components Description*

In this section the components of the Healthcare service on a MVNO slice are described.

**Table 12 – Components of a Healthcare service on a MVNO slice**

| Components name | Description | Major functionalities |
|---|---|---|
| Health care application | Application that medical staff uses on their device | • Application for medical purposes such as e-medical record, voice communication and vital monitoring, etc. |
| Mobile terminal | Mobile terminal used by medical staff | • Usable within a medical area or outside of the area<br>• Mobile communication<br>• Various applications are installed and works on a terminal. |

| | | |
|---|---|---|
| MUNO network | Utilize MNO's mobile network to provide mobile service as MVNO. | • MVNO's PGW, PCRF and authentication node to interface with MNO's SGW. |
| Closed network | Network service isolated from the internet physically and/or logically. | • Block a illegal access from the internet |
| FLARE | Network node developed by U –Tokyo. Many-core network processor is used for the data-plane while a x86 architecture processor is used for the control – plane. | • Refer to section 3.6.1.1 for the FLARE node description |
| Security NFV on FLARE | Softwarized security function for U-Plane.<br>It provide authentication function for terminals and applications | • The binary compiled from the source code programmed by C or C++ high-level language can be executed in the user-plane with PMD bypassing kernel similar to DPDK.<br>• Authentication function for terminals and applications by inspecting the contents of data. |
| Management server | To configure the security rules on the security NFV | • Security rules configuration from GUI<br>• display the current configuration |

### *2.3.3. Component interfaces*

The following table describes the interfaces among involved components in this testbed.

**Table 13 Components interfaces for Healthcare testbed**

| Components | Protocol | Major functionalities |
|---|---|---|
| Slice orchestrator – NFVO | HTTP REST | • Creating network service description<br>• Uploading TOSCA<br>• Creating network service record (instantiating)<br>• Retrieving status information and configuration<br>• Removing network service record and network service description |
| MVNO orchestrator - BSSO | HTTP REST | • Resource visualization<br>• Visibility of slice state<br>• Slice formation<br>• Characteristic change of slice |

| | | |
|---|---|---|
| MVNO orchestrator - FLARE | HTTP REST | • Refer to section 3.6.1.4 about the Orchestrator |
| UE – security NFV on FLARE | Original protocol developed by U-Tokyo | • To inspect data from UE to identify the applications and terminals |

## *2.3.4. Initial Evaluation*

### 2.3.4.1 Scope of the evaluation

The scope of the evaluation of this scenario is to form a slice securing the realization of healthcare services and realize the following step by step:

- Realizing security NFV by Deep data plane programmability.
- Form end-to-end slice from MVNO.
- Implement resources, security NFV, etc. held by MVNO on end-to-end slice.
- Interaction with the orchestrator.

### 2.3.4.2 Evaluation Criteria

The evaluation criteria are described below

- The resources available from MVNO can be visualized.
- To be able to combine selection of visualized resources and resources owned by them and to form slices for each service.
- End-to-end communication quality (bandwidth, speed, etc., QoS) can be set when forming a slice.
- These controls shall be realized by control through MVNO portal, orchestrator.

### 2.3.4.3 Expected results

It was expected that the activities bring the following results:

- Total sequence diagram is designed.
- Security NFV are ready for use.
- Simple interfaces with orchestrator are ready for use.

### 2.3.4.4 Experimentation Results and Evaluation

As a first step, a single demonstration test of security NFV was conducted in the hospital field. We targeted mobile terminals used by the medical staff in hospitals. Applications for browsing electronic medical charts, applications in conjunction with nurse call systems, and applications for hospital internal calls are installed in this mobile terminal. Hospitals and MVNO facilities are connected via a closed network and security is guaranteed, but it is necessary to prevent unexpected communication due to loss of the terminal, loss of communication SIM, viruses, and applications which are not permitted to use.

We installed a software module that works with security NFV for terminals used at hospitals, and strengthened the authentication of terminals, communication SIM, and applications to communicate. In this demonstration, it was demonstrated that the customization of the security function is performed from

the controller to the FLARE, the UE and the application to be used are discriminated, and the prohibition of unexpected use, access control, traffic control, etc. can be performed without affecting the service. The next step is to integrate this within the 5G!Pagoda network slice architecture, design a sequence for service development on the orchestrator and slice architecture and try to implement it.

### *2.3.5. Conclusions*

As a first step, we demonstrated the functional verification to strengthen security certification with healthcare use cases based on the first prototype based on Figure 7. In WP2, WP3, WP4 we will define a generalized architecture as the MNO / MVNO interface based on Figure 7 and demonstrate it as a testbed in WP5.

## 2.4. Ignition testbed 4 - Deep Data Plane Programmability Testbed

This use case scenario aims to create a testbed for deep data plane programmable core network (DDPPCN) slices and realise the 5G!Pagoda architecture in Tokyo and Berlin testbeds through facilitating the integration. The use case will be demonstrated through an orchestrated deployment of FOKUS' Open5GCore and UTokyo's FLARE platform in the Tokyo and Berlin testbeds. It aims to evaluate the compatibility and fitness of the Tokyo and Berlin testbeds on a technical level, as well as to provide the initial work towards the integration of the testbeds and their components. Once realised, it will be a showcase for features of the 5G!Pagoda architecture, such as deep data plane programmability, multi-domain orchestration, slice stitching and slice resource scalability.

### *2.4.1. Architecture Description*

This section provides a short overview of the architecture. For a more detailed description, please refer to D3.1 Section 5.4.

To create the deep data plane programmable core network (DDPPCN), the components will be deployed in three kinds of slices:

1. Core slices for the core network components
2. Control slices for the components interfacing between 1 and 3
3. Data Plane slices for the programmable data plane components

These slices will be deployed and managed by a slice Management and Orchestration (MANO) system. An overview of this can be seen in **Error! Reference source not found.**.

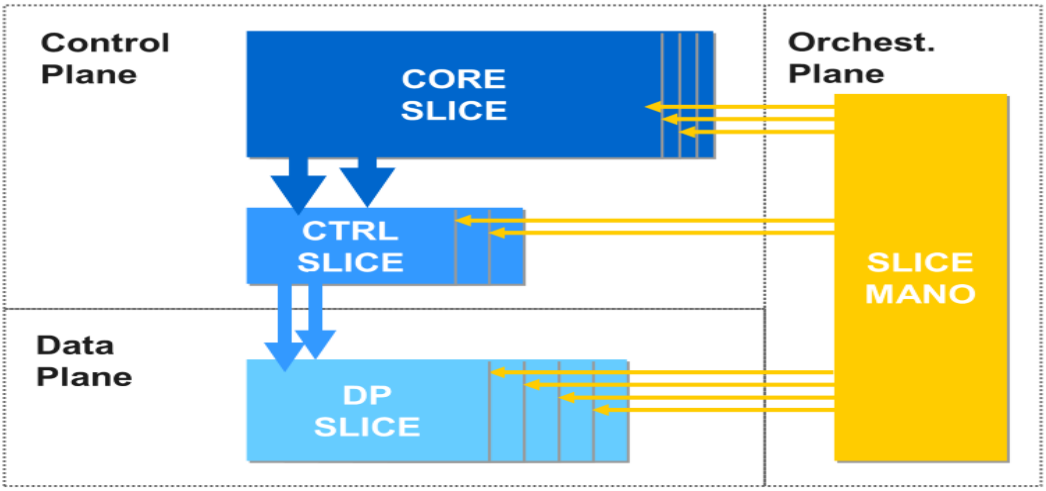


**Figure 8 DDPPCN Slice Orchestration**

The slicing architecture of the DDPPCN is presented in more detail in **Error! Reference source not found.**. The three types of slices will contain different components of Open5GCore. The Core slice will contain the components required for the 5G Core Network operation: AMF, SMF, PCF, UDM FE and UDR. The Data Plane slice will contain the User Plane Function (UPF). Lastly, the Control slice will facilitate access of Data Plane slices for Core slices by containing a controller component.

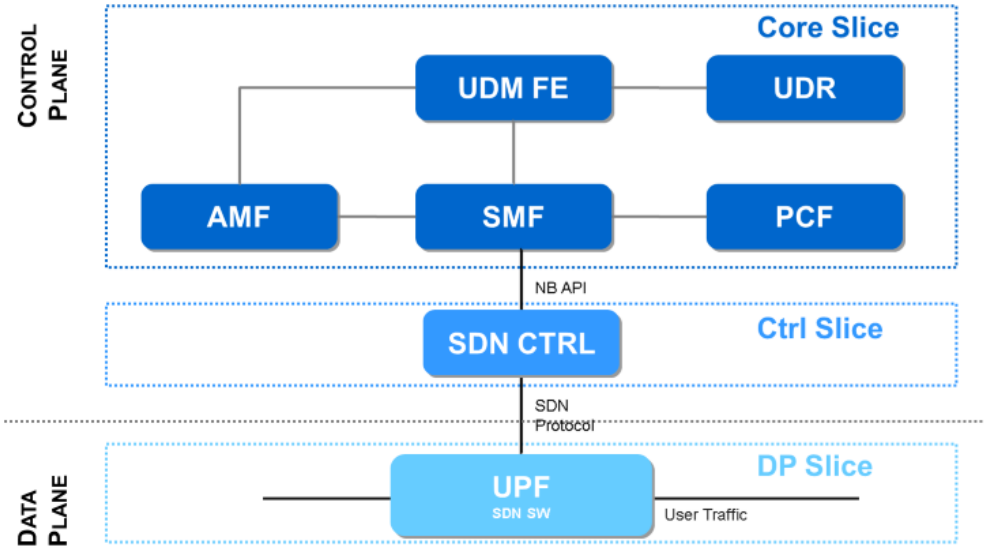


**Figure 9 DDPCN Slicing**

The slices can be deployed on the FLARE Platform as depicted in **Error! Reference source not found.**. The deployment takes advantage of FLARE's NPU to create the deeply programmable data plane, while Core and Ctrl slices operate on the CPU part of FLARE.

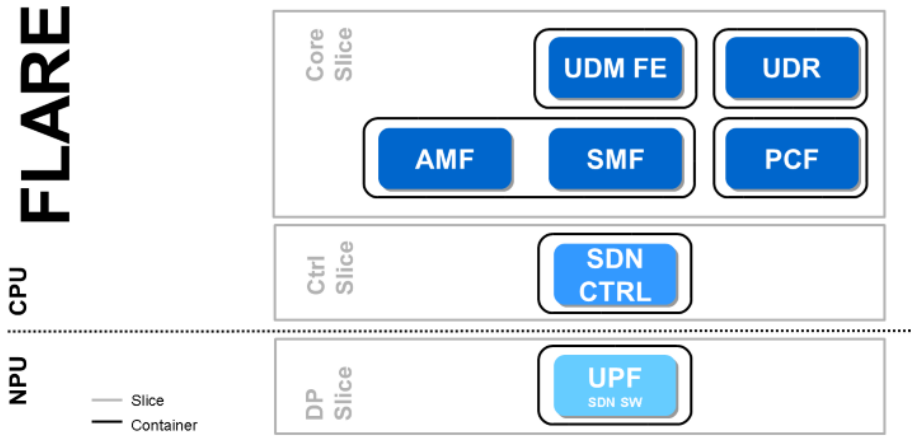


**Figure 10 Deep data plane programmability testbed**

## 2.4.2. Components

The following table lists the minimally required components for the use case.

**Table 14 DDPPCN components**

| Component | Description | Major functionalities |
|---|---|---|
| DDPP Platform | Specialised hardware platform for VNF deployments with deep data plane programmability support | • High performance networking capabilities for data plane traffic<br>• Data plane programmability support<br>• Generic VNF deployment support |
| 5G Core Network | Fully virtualised 5G Core Network implementation that can be customised to support deep data plane programmability and advanced slicing. | • Standard aligned 5G core network implementation<br>• Strong support for sliced deployments<br>• Highly customisable |
| MANO | Generic Management and Orchestration Solution for the 5GCN slice deployment on the DDPP platform. | • Ability to manage and orchestrate DDPP platform and core network in a user friendly manner |

## 2.4.3. Initial Evaluation

### 2.4.3.1 Scope of the evaluation

During the initial phase, it needs to be evaluated whether and how well the respective testbeds and components work together, to support the DDPPCN slice deployment.

#### 2.4.3.2 Evaluation Criteria

The initial evaluation criteria to consider are as follows:

- Compatibility between FLARE Platform and Open5GCore
- Slice-ability of Open5GCore

#### 2.4.3.3 Expected results

The activities were expected to yield the following results:

- Initial evaluation of components
- Identification of required adjustments to components for integration
- Implementation of these adjustments
- Initial deployment of Open5GCore integrated with the FLARE Platform

#### 2.4.3.4 Experimentation Results and Evaluation

- Components evaluation:
  - The FLARE Platform provides a proven infrastructure for the deployment of VNFs and is uniquely equipped for deep data plane programmability support, through its CPU/NPU architecture.
  - Open5GCore is a highly customisable and modular 3GPP aligned with 5G core network implementation. It has been used to provide VNFs for various research projects and testbeds before.
- Required Adjustments:
  - The FLARE Platform supports the deployment of docker containers. To integrate Open5GCore and FLARE, the need for a containerisation of Open5GCore components was identified as the primary adjustment.
- Implementation:
  - A generic software to create Open5GCore docker images was implemented.
- Deployment:
  - The resulting images were successfully deployed on the FLARE Platform.
- Additional:
  - An initial evaluation of compatibility between Open5GCore and the Open Air Interface eNodeB took place. This evaluation identified key issues for a potential future integration.

### *2.4.4. Conclusions*

The initial integration of Open5GCore and the FLARE Platform was completed successfully. The results should facilitate the integration of the Tokyo and Berlin testbeds and the creation of dynamically deployable DDPPCNs.

In the future, the Open5GCore needs to be fully integrated into the Tokyo testbed orchestration solution and a FLARE node should be deployed in the Berlin testbed. It is possible, that extra integration work is required for Open5GCore to support FLARE's NPU. Afterwards, the deployment of the sliced architecture can be realised and evaluated.

# 3. Integrated Testbed

The 5G!Pagoda testbed is the result of a process of customization, further development and integration of different prototypes, starting from the 5G Playground in Berlin and the FLARE & O3 in Tokyo testbeds with the addition of other initiatives from the partners to build dedicated slices and different use cases in order to present in a coherent manner the results of the practical implementation within the project as well as to enable the technology transfer between Japan and Europe.

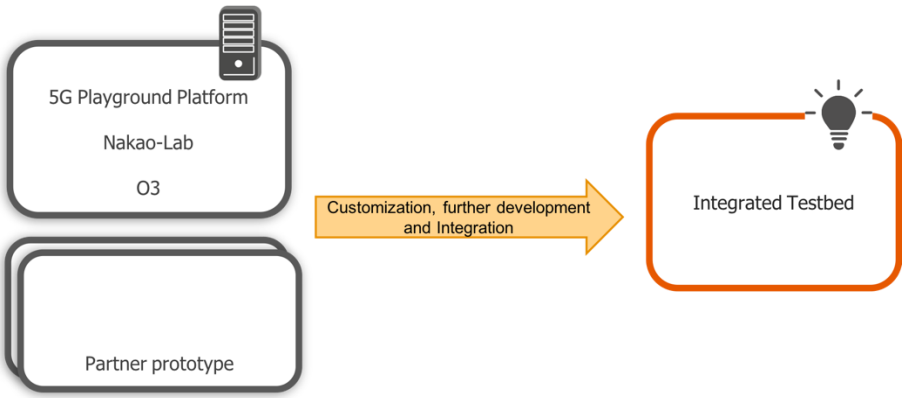


**Figure 11 Creation of integrated testbed**

As illustrated in Figure 11, the integrated testbed is based on the infrastructures deployed in the 5G Playground in Berlin and in the Nakao-lab and O3 in Tokyo. In the ignition phase, work was executed by a large number of partners to provide prototypes to be easily integrated into the testbed. As a second phase, the prototypes as well as the testbed nodes will be further customized, developed and integrated into a common testbed.

## 3.1. Integration processes

Different types of integrations can be performed, depending on the provider role that the partner has and on the equivalent type of components that need to be integrated.

- *Network Function Providers* – prototypes that provide functionality to run within the slices. The network function providers are contributing with pieces or complete slices to be deployed and managed within the integrated testbed.
- *Infrastructure Providers* – prototypes that provide functionality to run as part of the infrastructure. The infrastructure providers are contributing with software and hardware pieces on which the slices will be deployed. These infrastructure components come to compliment with a large amount of common hardware and virtualization software components (such as OpenStack) to provide additional value to the testbed.
- *Application Providers* – prototypes that are statically on-boarded as part of the slices. The applications are ported directly from their existing environments towards the different slices to provide means for demonstrations. As the project addresses mainly 5G networking, it is not in the scope of the project to optimize them.

### 3.1.1. Integration of network functions

In order to be able to integrate network functions, the process requires the following steps to be performed (steps 2-to-4 are highly dependent on the type of components deployed):

1. Embedding the network function into the NFV environment. For example, in order to deploy a service with Open Baton in Berlin, the definition of the Network Service Descriptors (NSDs) for the specific network functions are needed as well as the enablement of the VNFM for the specific network functions;
2. Providing benchmarking tests, including means to generate workload and eventually binding to monitoring functionality;
3. Providing individual assessments through the evaluation of the specific components into the environment;
4. Integration with other network functions, creating end-to-end network slices.

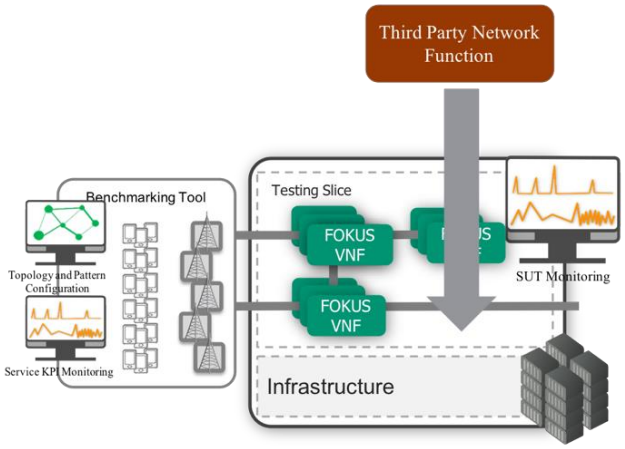


**Figure 12 Integration of NFs**

### 3.1.2. Integration of infrastructure components

On the other hand, the integration of infrastructure components requires a totally different approach:

1. Embedding the network function into the NFV environment. This involves inter-op testing with other infrastructure components and management alignment of the infrastructure component itself. The realization of this step needs a very careful consideration as the infrastructure may have side effects on the testing of the slices;
2. Testing with an existing set of slices to ensure the proper functioning of the infrastructure component;
3. Continuous testing while adding VNFs.

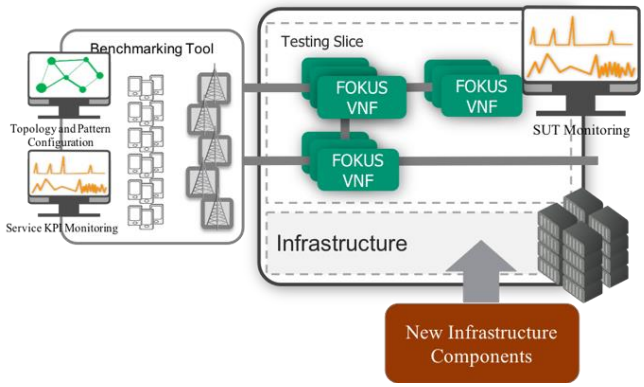


**Figure 13 Integration of Infrastructure Components**

### *3.1.3. Integration of third party applications.*

Finally, in case of "black-Box" applications the on-boarding of the product should be performed without any knowledge of its internal workings. As showed in Figure 14, the proper network connectivity needs to be provided through appropriate allocation of IP addresses as well as the consequent testing of the reachability of the application across the networking level VNFs needs to be performed.

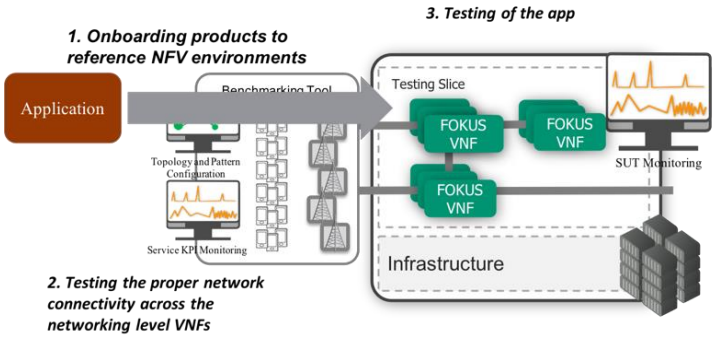


**Figure 14 Integration of black box applications**

## 3.2. Integration requirements

The followings are the minimum requirements for the testbed integrations between EU and JP:

- Tokyo and Berlin should have similar infrastructures, as any difference will require inter-operation at the other location. A comprehensive same infrastructure is not envisioned due to the differences in the approach towards the infrastructure and the virtualization technologies. However, an equivalent and interchangeable infrastructure is considered. With this, the integrated testbed enables for diversity at the infrastructure and virtualization levels as well as for the technology transfer. From an engineering perspective, the main insight which will be gained is on the level of equivalence needed for the infrastructures in order to be able to function into an integrated testbed. This will allow further deployments to determine the maximum level of heterogeneity feasible for a multi-slice infrastructure.
- Run a Service in Tokyo and separately from Berlin: same Network Service Images (VMs or containers) and the Network Service should be integrated with the local orchestrator. This means that the service should be initially tested in one location first and then to be deployed on the other.

For this, the infrastructure as well as the slice management components should be equivalent enough as not to require a large amount of integration. One of the expected results from this perspective is to determine the most appropriate balance between the heterogeneity of the infrastructure and the need of integration of the various services. As 5G!Pagoda has a rather large number of slices, a best-practice engineering result will show to the later deployments what is to be considered as cost of on boarding into the heterogeneous infrastructures of new slices which has to be added to the CAPEX of the new services.

- Run a service in Tokyo and Berlin at the same time: testbeds in Berlin and Tokyo should be interconnected and an end-to-end tenant should be established across the two environments. Due to the large delay between the different locations and due to the increased cost of the network connectivity, is deemed not suitable for the end-to-end data paths. It is easier to replicate the different data path components as to establish a proper end-to-end connection in the expected requirements. However, this solution provides a high potential for some of the control and management plane features such as maintaining a single subscribed database in the home environment while having “visiting network” deployments in the other locations.

## 3.3. High Level Architecture of the Integrated Testbed

In this section, a high level architecture of the integrated testbed is very shortly described in order to give to the reader the technology support and terminology to better understand the next sub-sections.

The high level architecture of the testbed is illustrated in Figure 15, including equivalent technologies in the testbeds in Berlin and in Japan.

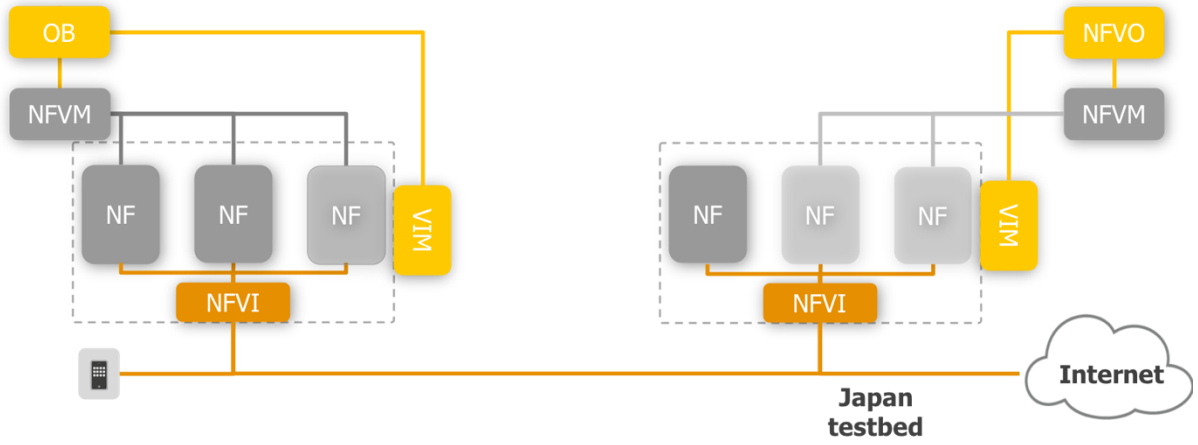


**Figure 15 Testbed high level architecture**

The following components are considered to be deployed in both locations:

- NFV Infrastructure (NFVI) – following the ETSI NFV terminology, the NFVI is composed of all the hardware infrastructure nodes and virtualization software which enables the presentation of the hardware resources as logically isolated virtual resources towards the slices including processing, storage and networking
- Virtual Infrastructure Management (VIM) – the VIM includes all the specific components which are needed to manage the NFVI including resource aggregation, brokering and scheduling towards the different virtual network components. Additionally, it includes management components enabling

to understand and to react to infrastructure modifications such as monitoring, fault management, etc.

- Network Functions (NFs) – the NFs are software components which run on top of the NFVI from which the slices are composed of. The NFs may be dedicated to specific hardware, such as in the case of legacy physical nodes or can be deployed transparently on different infrastructure nodes
- Virtual Network Function Manager (VNFM) – a set of management components which handle the management functionality of the NFs related to the deployment on top of the virtual environment including the new features for scalability and dependency management
- NFV Orchestrator (NFVO) – represents a set of high-level management components which enable the life-cycle management of the end-to-end slices according to the requirements from the slice tenant.

## 3.4. Interconnection Options

In the following section different network interconnection options are described for enabling the basic connectivity between the Berlin and the Tokyo node.

### *3.4.1. Best effort connectivity*

Best-effort is the standard service level in many IP-based networks (e.g. Internet). It is a connectionless model of delivery that provides no guarantees for reliability, delay, or other performance characteristics.

For the best-effort connectivity, available connections through the public Internet between Berlin and Tokyo will be used. Albeit multiple paths are available, the selection of the path is independent of the project as it mainly depends on the peering SLAs between the different Internet autonomous systems.

We assume that in most of the use cases, a best effort connectivity will be enough to support the specific communication requirements. This is mainly due to the fact that end-to-end data path connectivity should terminate in the local domain (delay constraints), while the connectivity between the different locations should be used for delay tolerant management and control plane operations.

### *3.4.2. Geant connectivity*

The connection of the two testbeds resided in Berlin and Tokyo requires the global network connectivity. For such connectivity, there are two options: (1) the use of the public network and (2) the research network connecting the overseas research institutes through the international links.

Fort the latter case, there are two research networks in Japan. The Science Information Network (SINET) is a Japanese academic backbone network for more than 800 universities and research institutions. SINET is also connected to the research network in Europe such as GEANT as shown in Figure 16[1]. The GEANT and SINET are connected at London with two 10Gb/s links. But this link is also shared by other science projects between Japan and Europe and congested. Although the full 10Gb/s cannot be obtained, the lower latency can be expected because this link can be connected to London access point of GEANT.

[1] SINET International link – [https://www.sinet.ad.jp/en/aboutsinet-en/interconnectivities]

In addition, there is another research network such as JGN, which is operated by National Institute of Communications Technology (NICT)[2]. JGN is designed to provide testing environment for researchers of advanced networking technologies. JGN has four international links to enhance collaborative R&D activities with global research institutions as shown in Figure 17. Using Japan to Singapore links, JGN can be connected to GEANT through Trans-Euraasia Information Network (TEIN)[3]. This route passes through Singapore and causes larger latency whereas this link is not much congested.

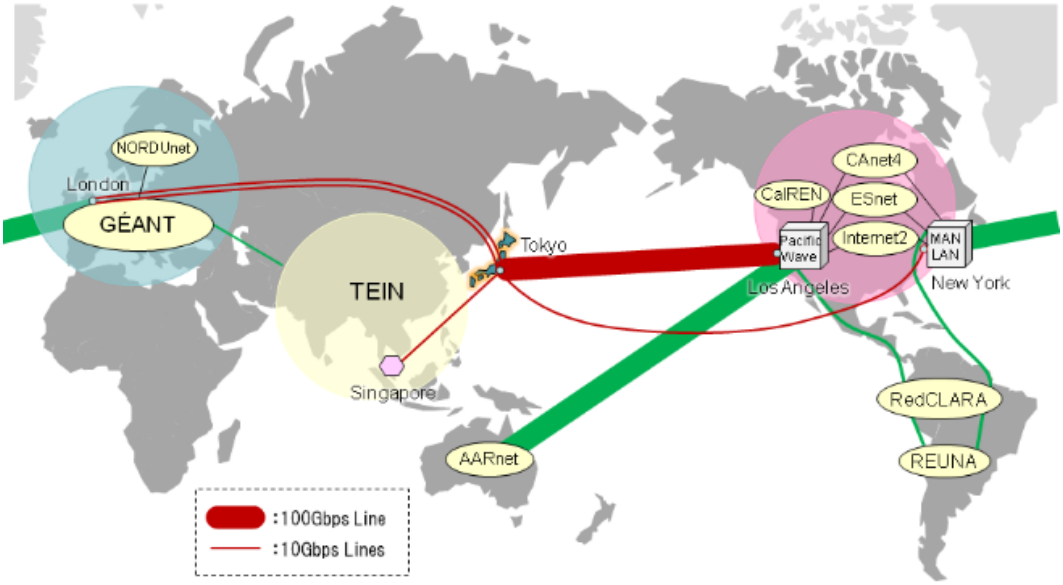


**Figure 16 SINET International links**

[2] National Institute of Communication Technologies – [http://www.jgn.nict.go.jp/english/networks/index.html]

[3] Trans-Euraasia Information Network - [http://tein4.net/tein4/about/history.do]

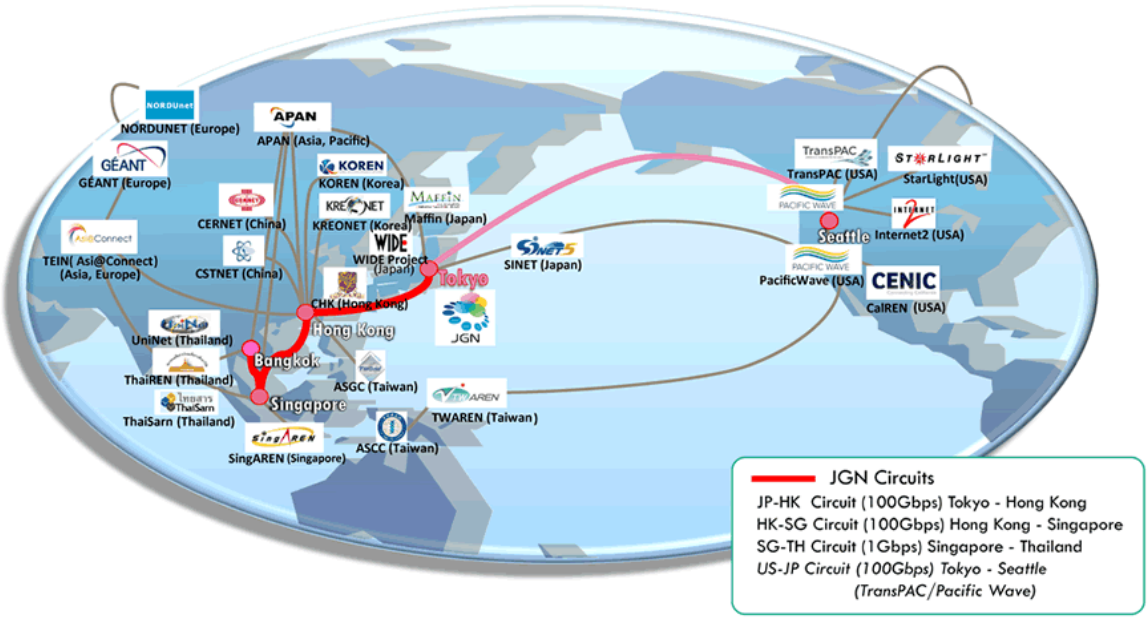


**Figure 17 JGN Global Networks**

The testbed resided in the campus of UTokyo can be connected to these networks. Figure 18 shows the campus networks of UTokyo to facilitate the connection to the testbed in Berlin. The testbed resided in UTokyo is connected to SINET which enables up to 10Gb/s connectivity to GEANT using L2VPN. To stitch multiple slices between Berlin and Tokyo, VLAN is desirable to enable the L2 connectivity and to segregate the link resources for data planes. SINET provides the L2VPN service for the interconnection of the two network domains. L2 connection of the data planes of the slices created in two testbeds can be obtained using research networks, while public networks as well as the research network can be used for the control plane.

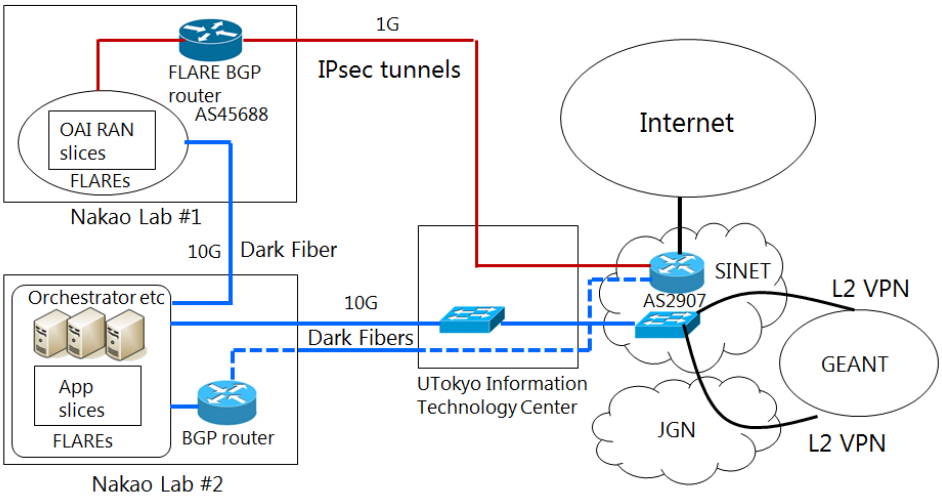


**Figure 18 UTokyo Campus networks**

### 3.4.3. IPsec Interconnection

The interconnection between testbed routers can be realized by using IPsec tunnels with pre-shared key authentication. The individual parameters and encryption algorithms can be chosen based on EU and JP VPN gateways capabilities to reach the optimal encryption strength.

With this solution, both the private infrastructures in Berlin and in Tokyo will be open only for the integrated testbed following a business-to-business collaboration model. Next to the security provided, the IPsec connectivity provides an IP level overlay making transparent the network between the two locations. With this, the end-to-end networking solution is simplified and thus, easier to debug in case of issues.

### 3.4.4. Selected Interconnection Option

Overcoming the best effort limitation of the public Internet, the GEANT facility allows to request QoS guaranteed network connectivity services between the deployed NFs. Furthermore, dedicated network services can be established on demand for each slice, with guarantees in terms of bandwidth, reduced jitter and service reliability.

However, an IPsec tunnel will be established between EU and JP locations first. Once the integrated testbed has reached a stable configuration, the GEANT connectivity is planned to be used.

Specifically, individual subnets interconnected by an encrypted IPsec tunnel, using IKE version 1 protocol, have been established between edge VPN gateways located in each testbed and the IPsec connection is currently using the Internet to create the connection.

## 3.5. Integration Options

There are multiple options to do the interconnection between Japan and European testbeds. The options differ in the complexity and in the features that are provided. The following sections will outline the three possible options.

### 3.5.1. Public IPs

The most basic type of interconnection between two testbeds is to assign public floating IPs to each Virtual Machine and then use a direct Internet access to create the connection between the public floating IPs of the machines. By using this approach, the traffic is routed through the Internet without encryption. By using direct Internet connection the usable protocols are limited to Routable IP packets only. If direct layer two connectivity is needed between the VMs then a tunnel has to be used.

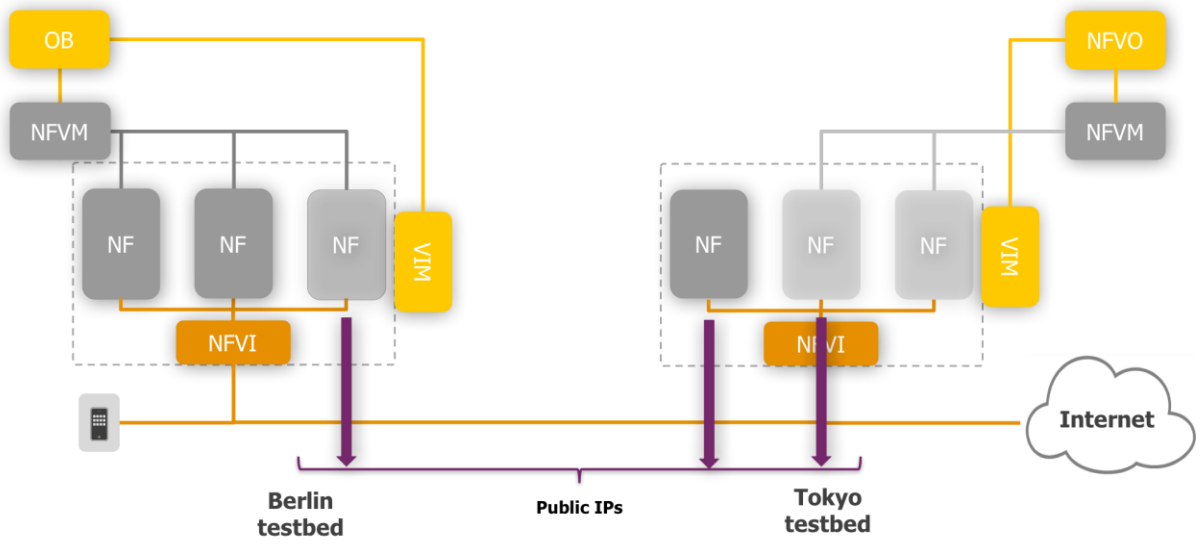


**Figure 19 Interconnection with public IPs**

### *3.5.2. Routable Network*

This solution uses dedicated subnets that are assigned to each of the testbeds. The private subnets are interconnected using a VPN tunnel which is established between dedicated edge routers at each testbed. The VPN router will create a tunnel connection to directly transport traffic between the testbed islands located in Japan and Europe. By using this approach it is possible to use IP protocols that do not work over the public Internet. It also provides confidentiality by transparent encryption of the traffic at the VPN router. Each VM in the testbed will receive an IP address from the testbeds assigned subnet. By using this solution, each testbed needs to have a dedicated non-overlapping IP subnet.

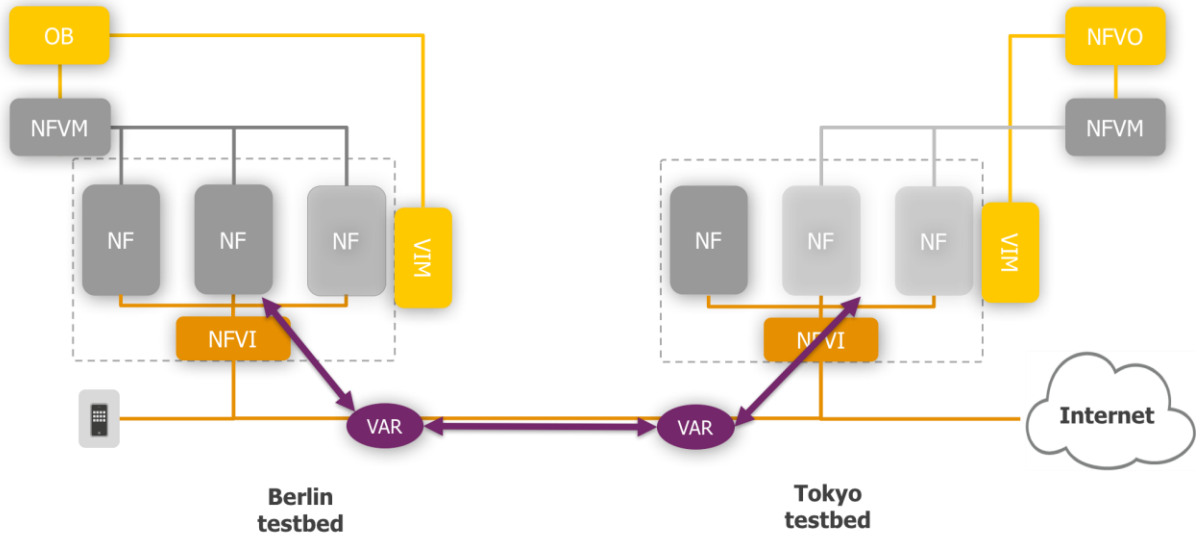


**Figure 20 Interconnection with routable networks**

### *3.5.3. Single Virtual Network*

This solution includes a single virtual network established at the European and Japanese locations through the VPN as a Service (VPNaaS) function of OpenStack and a single DHCP server for IP addresses allocation.

NFs can communicate transparently through a Layer 2 network connection and the result is that NFs are similar to being in the same location. However, this solution may not work because of the large delay generated by ARP messages sent to discover the link layer address associated with a given IP address.

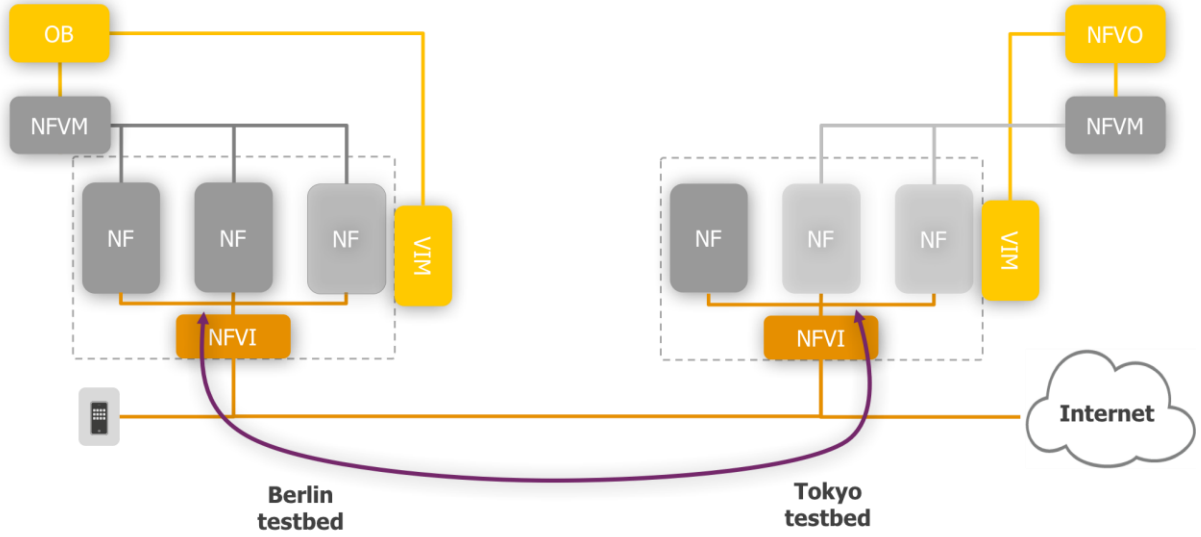


**Figure 21 Interconnection with single virtual network**

### *3.5.4. Selected Integration Option*

The evaluation of these different solutions has been done in the context of the testbed requirements described in Section 3.2.

Assigning Public IPs is the best solution in terms of easiness of realization, but only in case there is no need for specific private network forwarding. Since 5G!Pagoda testbed requires a private type of interconnection between Europe and Japan, a routable network or a single virtual network solutions are the only feasible options.

However, considering the network delay between Japan and Europe is so high, it does not allow operations of L2 protocols, like ARP. Thus, integration at L3 is the most appropriate solution and the services should be ready to adapt for this. For example, the EPC was designed with routing between the different entities and since they are IP based, can pertain to different networks.

## 3.6. Selected Technologies

Figure 22**Error! Reference source not found.** describes the EU/JP integrated testbed architecture. The real challenge is the construction of the end-to-end orchestration framework at the management layer according to the reference architecture designed in WP2 and WP4, allowing the seamless deployment of services in both European and Japanese locations.

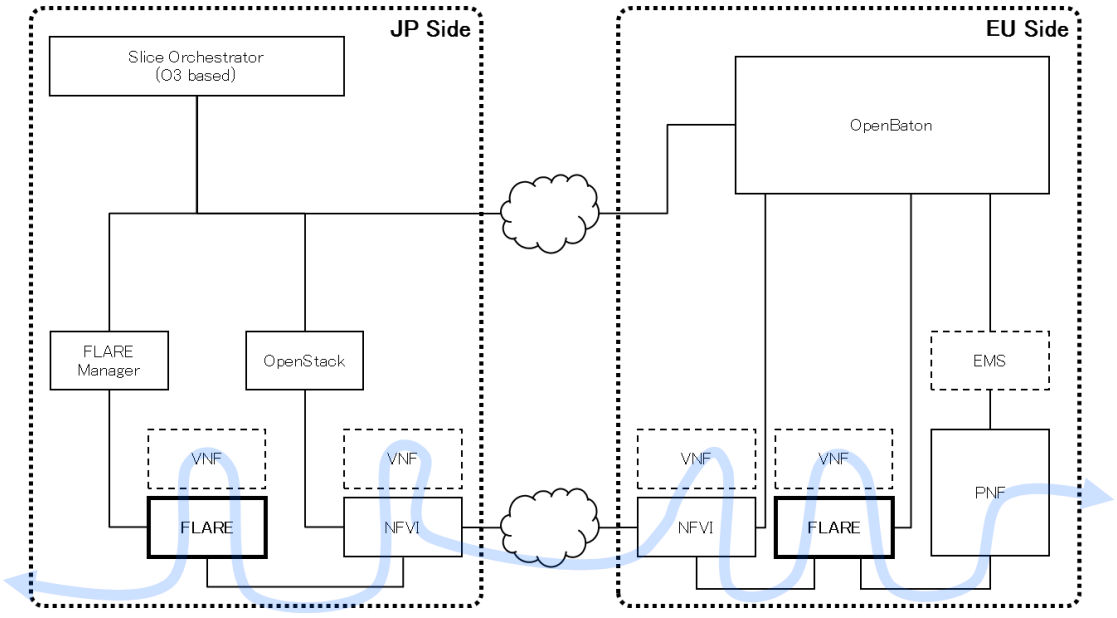


**Figure 22 Architecture of the testbed integration**

In the following sections, a detailed description of the UT and the 5GPlayground testbeds is provided from the architecture and technology perspective. A very wide set of communication technologies were chosen, enabling the coverage of the full spectrum of use cases.

## *3.6.1. Selected Technologies of UT Pagoda testbed*

5G!Pagoda testbed in UT is designed to be aligned with the documents in WP2 and WP3 to validate the 5!Pagoda specific featured as well as required NFV capabilities defined by SDO. The testbed composes of NFV core domain, orchestrator domain, and management domain.

5G!Pagoda testbed should also emulate the network domains such as RAN/front-haul, back-haul, transport, and likely cloud. The orchestrator should be also integrated with testbed in order to create the service slice based on the template/blue print.

The unique feature of the 5G!Pagoda testbed is to support VNF with data-plane programmability. In addition, the orchestrator can manage NFV consisting of such VNFs. UT proposes to use FLARE node to enable such VNFs. In this section, the testbed composing of FLARE nodes as NFVI and the associated orchestrator are addressed.

### 3.6.1.1 FLARE node

5G!Pagoda UT testbed uses FLARE nodes which were described in D3.1. FLARE node has a unique architecture different from the node using x86 general-purpose processors. In FLARE nodes, a many-core network processor is used for the data-plane while an x86 architecture processor is used for the control – plane. The chip used in FLARE node is TILE Gx multicore processor. Each core of TILE-Gx is a full-fledged 64-bit processor having local cache and supports a flexible virtual memory system. Any of the cores can independently run the operation system of standard Linux as well as SMP Linux as a group. Each core is interconnected through the non-blocking mesh based data baths instead of ring data bath. Them high throughput and low latency are enabled in the multi-core operation.

The binary compiled from the source code programmed by C or C++ high-level language can be executed in the user-plane with PMD bypassing kernel similar to DPDK. In addition, the physical interfaces are directly connected to NIC using Broadcom I/O chip. Hence in terms of the throughput, there is not the

bottleneck due to PCIe. Currently, in the testbed, 1GbE as well as 10GbE ports are supported in FLARE. But in the next version of FLARE, 40GbE QSPF interface will be supported in order to facilitate the 40G transport network.

In addition, in order to further improve the programmability, Click modulator router is ported in the many-core network processor platform. Click element providing network functions can be compiled by the processor SDK together with the new Click elements related to processor IO function specifically programmed. With these reasons, it provides the high performance without the special I/O optimization.

FLARE node can support the architecture with the separation of control-plane and the data-plane. FLARE VNF functions are instantiated in a pair of containers created in both processors, respectively with the interconnection of PCIe interfaces. The LXC created in x86 architecture processor used as the control plane can support Docker. Then multiple Dockers with different function can be operated in the control plane LXC.

**Performance of FLARE:**

First of all, the packet forwarding performance of FLARE data plane was shown in Figure 23**Error! Reference source not found.**. The packet forwarding was performed with Click configuration of IO Click elements connection by NULL element. Input and output IO click elements were dedicatedly developed for the many-core processor used in FLARE. For 64byte short packet, 14.88 M packets per second (PPS), which is equivalent to 10Gb/s, was obtained using three cores when the processor operated at 1GHz. For the long packet length, 10Gb/s could be obtained merely using one core.

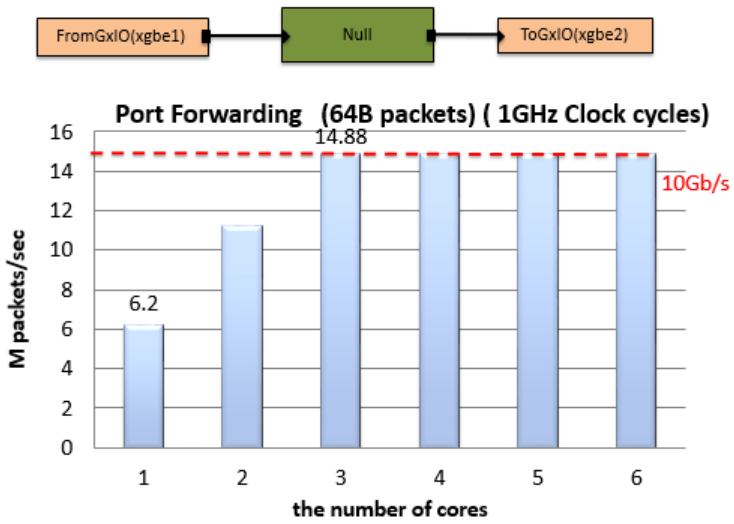


**Figure 23 Packet forwarding performance of FLARE data plane**

Next, the performance of L2 switch with MAC learning function was measured. L2 switch function can be implemented by using L2 switch element together with IO Click elements as shown in Figure 24**Error! Reference source not found.**. The measurement was performed as a function of the number of cores for 64byte short packets and one million flow entries. More than 10Gb/s throughput was obtained using 6 cores whereas for the long packet, 10Gb/s throughput was obtained by a single core.

The measurement was done for more complex Click configuration such as OpenFlow switch. In this case the control plane was used to communicate to the OpenFlow controller using OpenFlow controller. In the data plane, the click element of OpenFlow data path was developed and the OpenFlow function was implemented by the Click configuration using Click IO elements used in the L2 switch. The measured throughput was shown in Figure 25**Error! Reference source not found.** as a function of the number of cores

used in the slice for 1514 byte packets and 1 million flow entries. In OpenFlow function, six cores are needed to obtain 10Gb/s throughput for the long packet. However, throughput linearly increased with respect to the number of cores. In terms of the latency, 12*u*sec was measured while 3 to 5 *u*sec for the hardware OpenFlow switch.

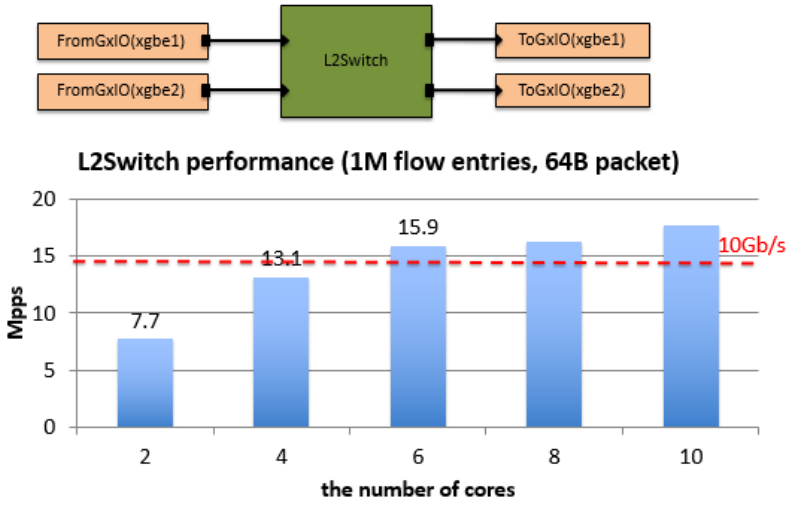


**Figure 24 Performance of L2 Switch with MAC leading function**

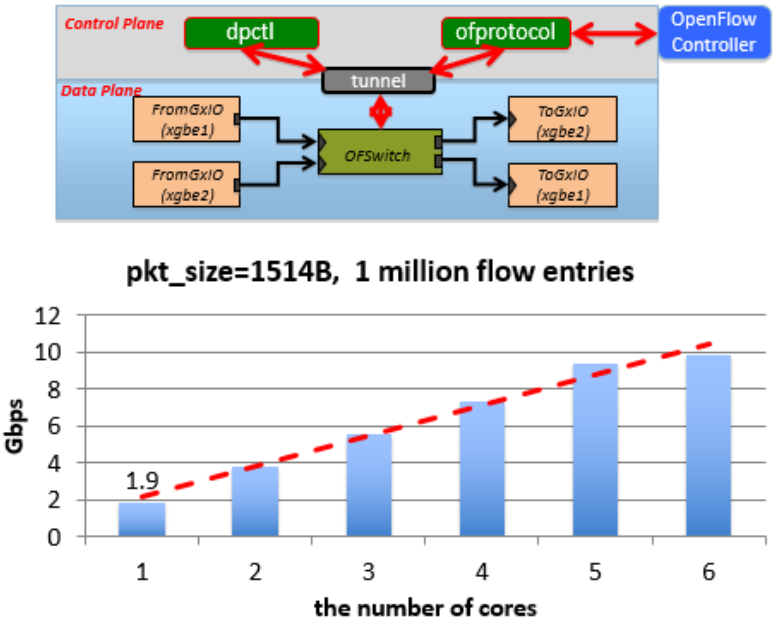


**Figure 25 Throughputs**

**Data-plane programmability:**

FLARE can support any flame packet. In order validate the programmability of data plane in FLARE, the measurement was performed for the customized ether flame packet. The length of MAC flame was changed to 12 bytes from 6 bytes. The L2 switch to support such extended MAC flame was implemented by Click element. The measured results were shown in 26. It can be confirmed that the throughput was not changed comparing the standard MAC length switch for the long and short packets.

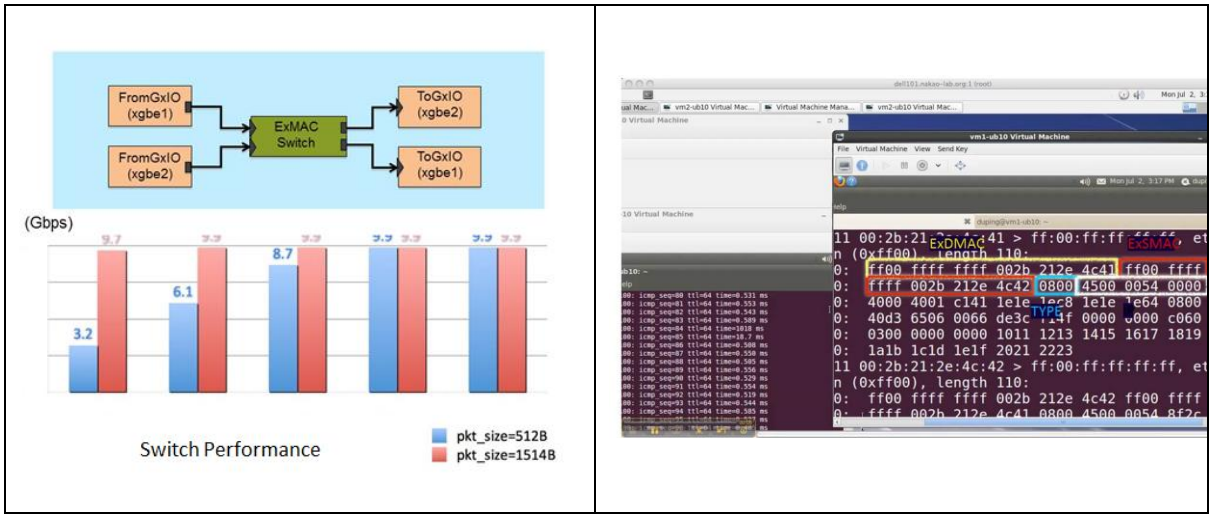


**Figure 26 Throughput of extended MAC switch and its verification of MAC Ethernet frame**

### 3.6.1.2 Implementation of RAN using FLARE

The demand of mobile resources changes dramatically in a day. In terms the efficient resource allocation to the mobile network, the softwarization of the mobile network enables the dynamic change of the resources using the virtualization platform together with the network slicing.

Mobile network can be decomposed to the front-haul and back-haul networks. In front-haul, the base station function consisting of RRH (radio remote head) and BBU. The radio packet processing function such as EPC in the case of LTE is resided in Mobile back-haul network or in the cloud as C-RAN.

In the initial evaluation, software LTE mobile network is implemented using OAI software. RAN slicing technology is required but still being developed. The research and development of advanced radio scheduling algorithm is required. The current OAI release does not support 5G Core but in the future.

**Sliced eNodeB:**

An OAI LTE eNodeB application is run by Docker instance in each LXC slice created using x86 processor. Each eNodeB in a Docker instance is isolated and can be replaced within a LXC slice. UEs are attached to an eNB instance through RRH of software radio platform USRP X310 which connects to an eNB with XGBE pass-through technology. The traffic flows of different slices are isolated using VLANs Open vSwitch (OVS) running in the host machine. Hence the packets from different eNodeBs are segregated each other using VLAN tag and forwarded to the designated EPC instance with VLAN ID.

The software PHY layer of the eNodeB is responsible for receiving packetizated radio signal frames from RRH of USRP X310, performing symbol level processing and delivering subframes to upper layer for further processing. Since each subframe has to be acknowledged back at the USRP X310 to allow for retransmission, there is an upper bound for the total processing delay for each subframe in the software eNodeB RAN functions.

To successfully provide frame/subframe timings, the execution environment of the software eNodeB has to support strict deadlines of the code execution. OAI community has the following realtime optimization issues for efficient signal processing over General Purpose Processors (GPPs) as opposed to the implementation by Application-Specific Integrated Circuits (ASICs).

- State-of-art Intel processor with vector processing instructions (AVX2) is used to optimize the turbo decoding and FFT processing of LTE subframes.

- Disabling CPU's automatic underclocking function so that the processing time will not be increased to save the power.
- Low-latency (or real-time) kernel is used to reduce the preemption delay in Linux scheduling.

According to the experiment, it is still found that the critical realtime issue in the above OAI's ingredients remained unresolved due to the inefficient or unnecessary actions. Some subframes cannot be processed in deadline due to the subframe processing threads which could be preempted by other threads even under a low-latency kernel. To solve this issue, we did the following extra optimization. The dedicated data-plane cores are assigned to subframe decoding/encoding threads to reduce processing delay by reducing context-switching.

So far the physical multiple RRUs are accommodated by eNodeB created in the network slice. EPC part can be implemented using the architecture of the data-plane and the control plane separation in the FLARE platform as stated in D3.1.

**Data-plane of FLARE EPC:**

The GTP-U channel creation and user data processing GTPV1-U kernel module in the original OAI software were separated and implemented in data plane of FLARE node. The separation of data plane is expected to offer the extensibility as well as to improve the performance. As the data plane of FLARE uses multi-core processor, the scalable performance improvement can be obtained by dividing functionality and enabling parallel processing across multiple cores. In addition, FLARE uses Click modular router to implement the network functions with its the easy network-programming framework. Click language abstracts the underlying architecture such as I/O function, inter-core communication and exposes the relevant necessary details to a set of reusable Click elements. In fact, SP-GW data-plane was implemented by chained Click elements. The packets receiving from eNB are forwarded to the designated slice by Slicer slice together with the classification of the signalling packets (e.g., GTP-C) and data packets (e.g., GTP-U). The signalling packets, namely control packets, is forwarded to control-plane while the data packets to data plane and processed by many-core processor in the data plane.

**Control plane of FLARE EPC:**

The control-plane of FLARE node is provided by LXC created in x86 processor. The signalling entities of EPC slice (e.g., MME and the control-plane of SP-GW) are executed by Docker instances in the LXC slice. In addition, HSS entity is executed by another Docker instance in the same LXC slice. These two Docker instances are isolated and can be replaced without interfering with others. The interfaces between the two dockers are implemented with internal Ethernet links. They can communicate with each other via TCP and SCTP protocols.

As the initial evaluation of the system, the throughput was measured by using Speed Test application using the commercial Android phone as the function of the bandwidth. The example of the downlink throughput measurement is shown in Figure 27. This is the initial result although the detail analysis and tuning have not been done yet.

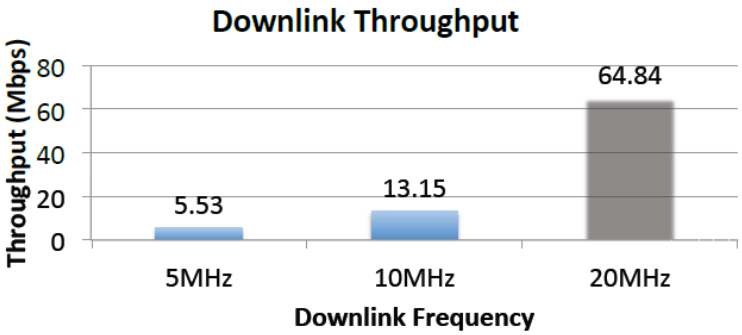


**Figure 27 Downlink throughput**

The parameters used in the experiment were shown in Table 15.

**Table 15 Parameters**

| | |
|---|---|
| Frequency | Down Link：variable<br>Up Link：variable |
| Frequency Bandwidth | Variable |
| Raido Format | X7W |
| Multiplexing Scheme | Down Link：OFDMA<br>Up Link：SC-FDMA |
| Modulation Scheme | Down Link：QPSK、16QAM、64QAM<br>Up Link：QPSK、16QAM |

**FLARE Mobile Network Testbed:**

Based on the FLARE software LTE architecture, the mobile network can be instantiated as NFV in the slice. **Error! Reference source not found.** Figure 28 illustrates the block diagram of LTE system. The LTE system can be decomposed to VNFs which can be independently provided from the repository. The *yaml* based instantiation is being studied.

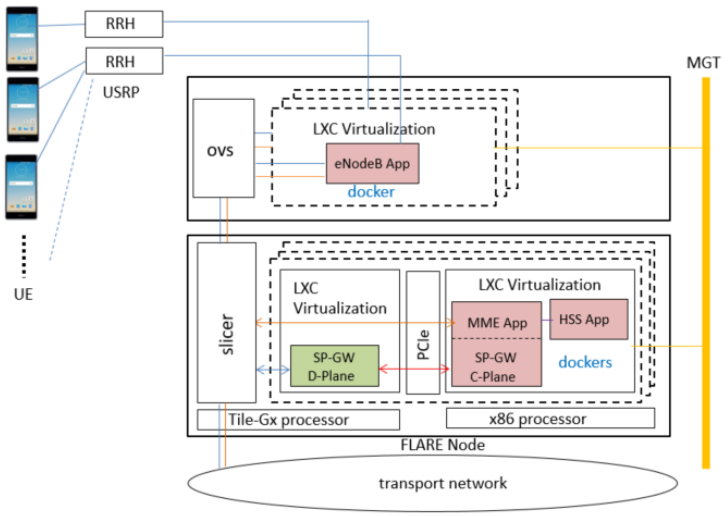


**Figure 28 Block diagram of FLARE LTE system**

#### 3.6.1.3 FLARE Testbed

The slices are created and deleted on demand basis. To support the request of the increasing number of slices, the slice should be created in a scalable manner. Then the scalability of FLARE slice is investigated. The two type of architecture were designed.

**Type 1: Aggregation of 1U FLARE nodes**

A 1U FLARE has two 10GbE ports and eight GbE ports for the data plane and transacts up to 40Gb/s internal packet processing. The total number of cores in TILE-GX36, which is currently used in 1U FLARE, is 36. For the packet processing for the medium packet length, 4 to 8 cores are required for each slice. Accordingly, three to six slices will be created in a FLARE node because eight cores are consumed to Slicer slice. To support many FLARE slices, the architecture to connect multiple FLARE nodes through the switch is proposed as shown in Figure 30. However, there is the limitation of the physical port count of the switch to connect the FLARE nodes especially using GbE ports. Therefore, this type architecture will be used for the medium node or the initial version of the testbed.

**Figure** 29 **FLARE node**

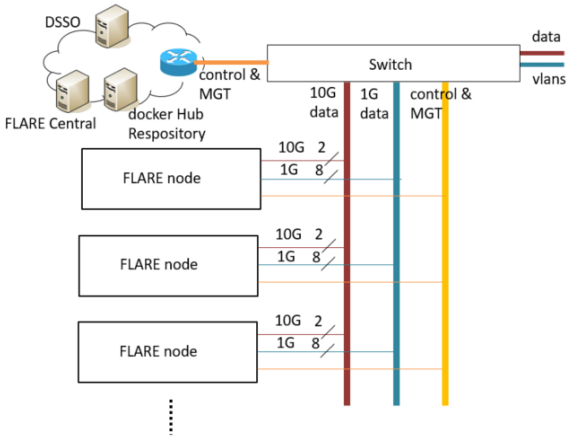


**Figure 30 Type 1**

**Type 2**

To improve the scalability of FLARE node testbed, Type 2 architecture is being developed. The PCIe card for the TILE-Gx processor is full-height size to support 10 physical interfaces. Due to its size, it is difficult to accommodate multiple cards in one server unit. At present, to avoid this problem, the low-profile PCIe card was designed using a 40Gb/s QSPF interface. Thanks to Low profile size, multiple PCIe card (up to 7) can be accommodated in a single server having multiple PCIe slots.

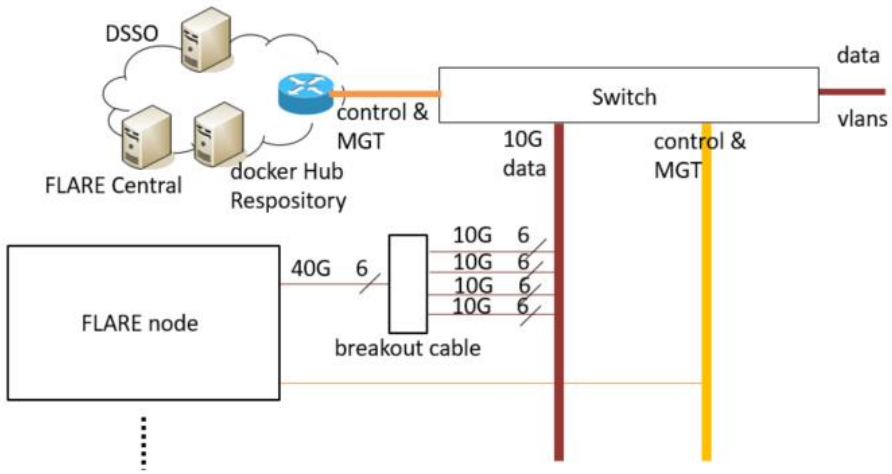


**Figure 31 Type 2**

#### 3.6.1.4 Integration of Orchestrator and FLARE node

Figure 32 shows the generic 5G!Pagoda architecture. The domain specific orchestrator in Japan is developed by Hitachi. FLARE node infrastructure is being integrated in this architecture.

A FLARE node has its own FLARE Manager, aka FLARE central. FLARE central has its APIs to talk to the client. The interface to communicate between FLARE central and DSSO was developed using Python scripts. Figure 33**Error! Reference source not found.** shows how to instantiate ICN/CDN service slice as an example. The following is the sequence to create ICN/CDN service in the slice by Hitachi Orchestrator:

(1) To define FLARE slice through FLARE manager.

(2) To log on FLARE manager.

(3) To assign slice user to the slice to be created.

(4) To assign FLARE node and to create the slice in the FLARE node.

(5) To create the interface using Click scripts in FLARE data plane.

(6) To create VMs in compute nodes of OpenStack.

(7) To register resources in the resource pool.

(8) To define VNFs of ICN/CDN to be created in the slice.

(9) To allocate slice resources required for the service deployment.

(10) To create ICN/CDN service in the slice by deploying VNFs.

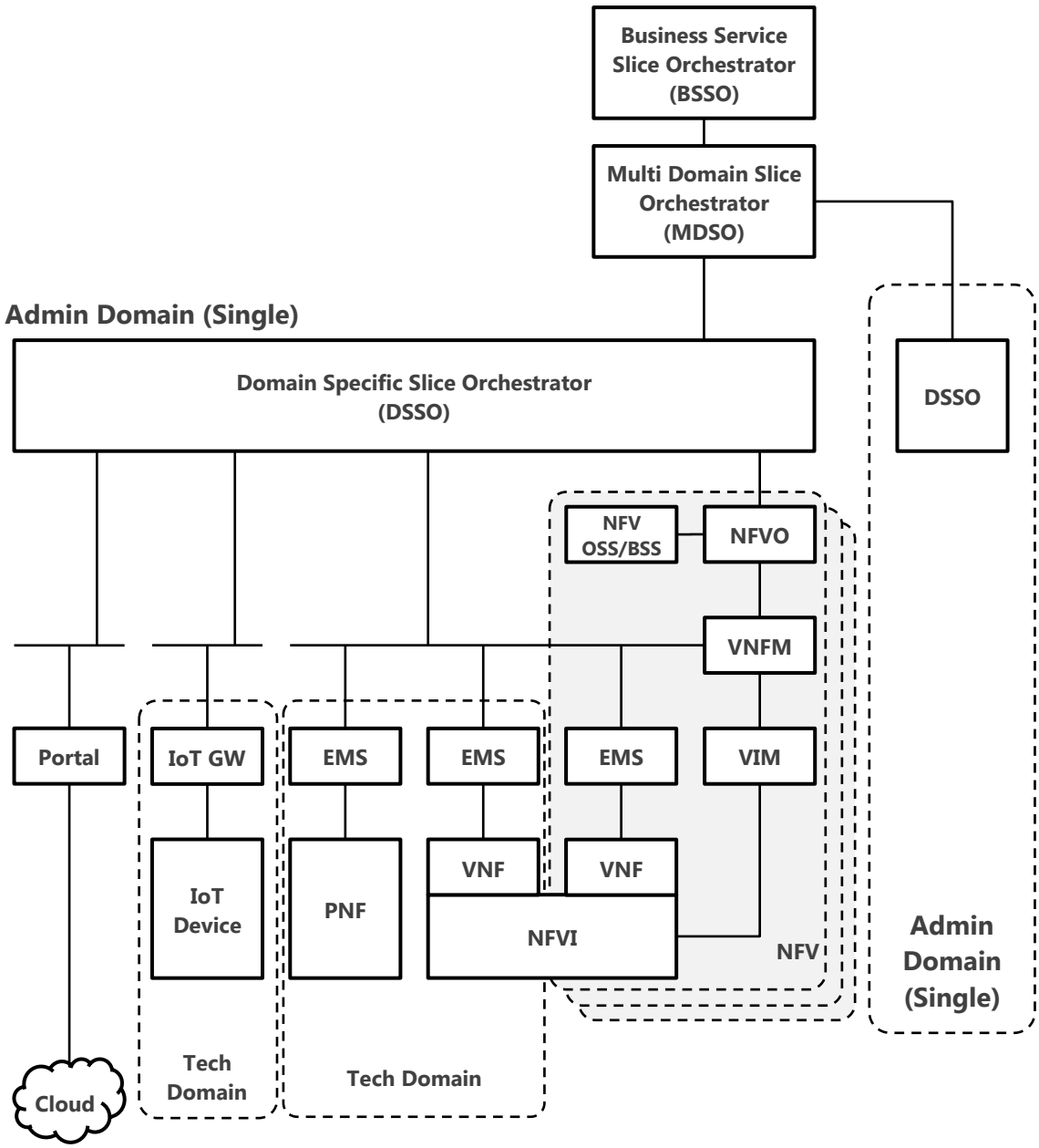


Figure 32 5G!Pagoda generic architecture

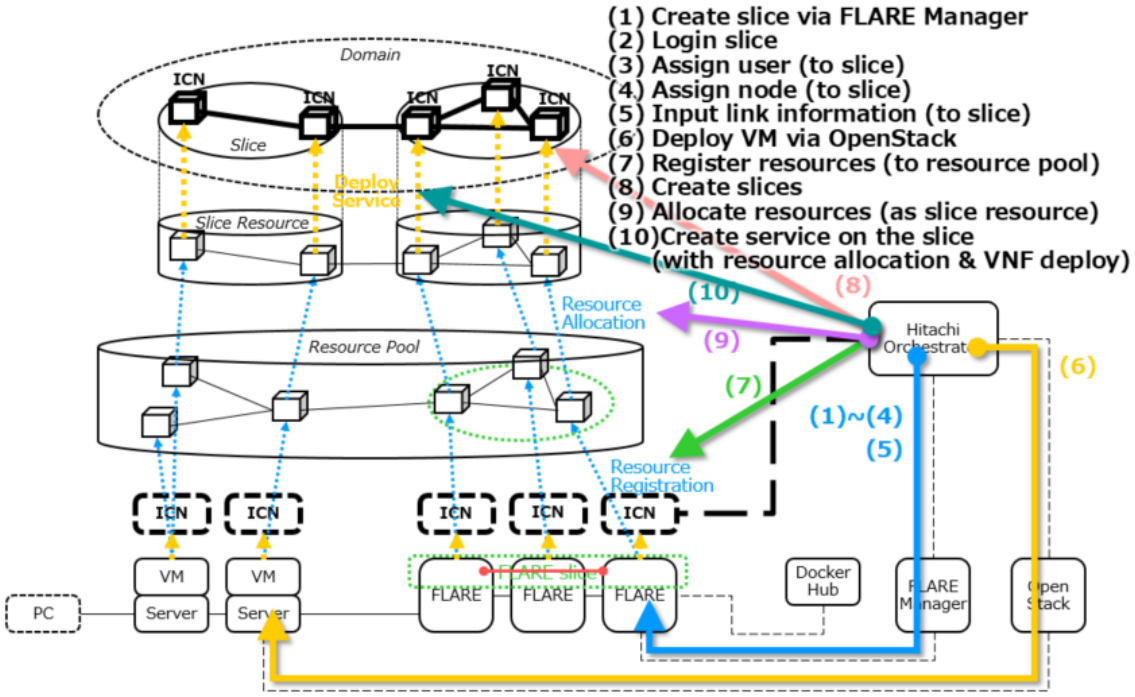


**Figure 33 Instantiation of ICN/CDN service slice**

## *3.6.2. Selected Technologies of 5G Playground testbed*

The 5G Playground testbed includes different components that are configurable depending on the specific use cases required and can be instantiated in parallel in multiple slices. As illustrated in Figure 34, the following infrastructure is considered for the deployment of the testbed:

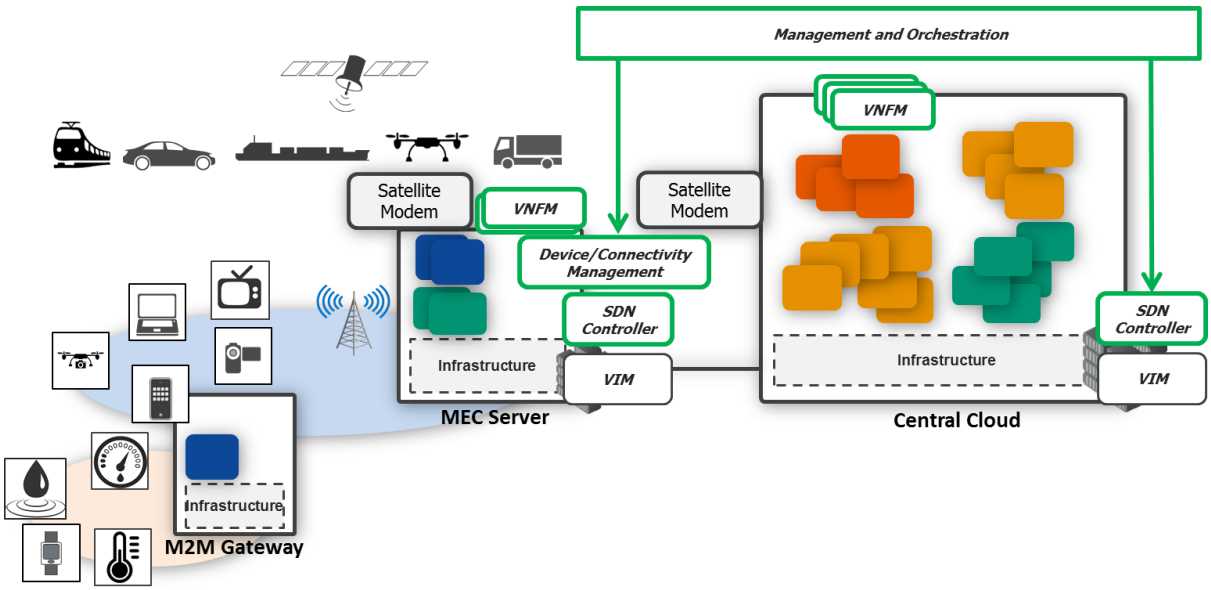


**Figure 34 5G Playground testbed infrastructure view**

- Hardware components – a set of data centre components which are installed in different combinations for responding to the specific use case requirements (grey).

- Virtualization enablers – the proposed testbed is based on OpenStack [] as a virtualization solution and it is based on the KVM hypervisor (grey).
- Management and orchestration – for the inter-data center communication an SDN solution based on OpenSDNCore is proposed. For interconnecting the devices and for their dynamic configuration, a device and connectivity management solution is proposed as included in the Open5GMTC toolkit. The testbed is orchestrating using the Open Baton toolkit. For this each of the components includes an ETSI NFV compliant VNFM.
- Radio components – the testbed integrates with a large number of off-the-shelf eNBs as well as with non-3GPP accesses such as WiFi. Several testbeds were executed in the past for the integration of the satellite networks, however, they were integrated not as access, but as backhaul.
- Devices and applications – the testbed is able to integrate with standard compatible devices and applications according to 3GPP 4G system architecture, ETSI NFV and OMA LWM2M and OneM2M.

#### 3.6.2.1 The Berlin 5G Playground

The 5G Playground in Berlin represents a live testbed instantiation of the Fraunhofer FOKUS toolkits integrated within a 5G network environment with other external components creating a unique comprehensive, open, flexible and easy to replicate testbed within the 5G research environment. The 5G Playground was designed to address use cases for: interoperability, product prototyping, remote experimentation and prototype evaluation and product calibration. It includes a dense wireless environment based on Open5GCore and third party radio, a multi-data center environment for SDN and NFV research, a massive device connectivity lab as well as a shared cloud infrastructure.

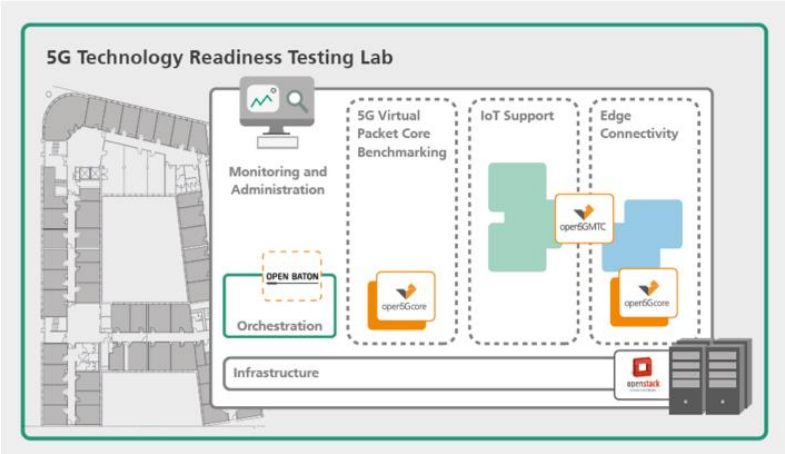


**Figure 35 5G Playground in Berlin**

As illustrated in **Error! Reference source not found.**Figure 35, the following toolkits are part of the 5G Playground. They include a comprehensive set of components to address dynamic network management using the OpenSDNCore toolkit and Open Baton for NFV and MEC orchestration.

#### 3.6.2.2 Open5GCore toolkit

Open5GCore **Error! Reference source not found.**is an R&D prototype for mobile core networks beyond 3GPP Release 13+, supporting 5G, 4G (LTE) and WLAN, including NB-IoT support and control-data plane separation as currently under standardization within 5G networks.

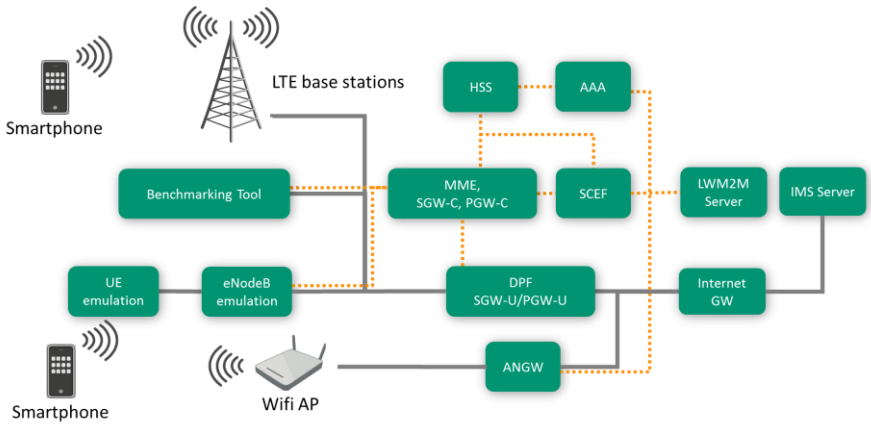


**Figure 36 Open5GCore Architecture**

The fundamental Open5GCore functionalities are:

- UE connectivity manager for Android and Linux
- MME/AMF – based on 3GPP EPC implementation (with optional data path selection)
- [SGW-C+PGW-C]/SMF – a grouped OpenFlow controller with GTP support able to allocate IP addresses and bearers
- HSS/UDM – including S6a Diameter interface
- eNB emulation with NAS overlay over IP communication
- UE emulation with NAS support
- A first implementation of the essential 3GPP NB-IoT features (Release 13 – TS 23.682) enabling the demonstration of low energy IoT communication
- Benchmarking for providing quantitative evaluations of different customized core networks on top of different resource infrastructures
- WLAN support with 3GPP standard connectivity for WiFi as Trusted Non-3GPP Access
- Elasticity support, Enabling graceful scaling, load scheduling and high availability
- VoLTE with the integration of Kamailio IMS, an open source implementation of 3GPP IMS

#### 3.6.2.3 OpenSDNCore toolkit

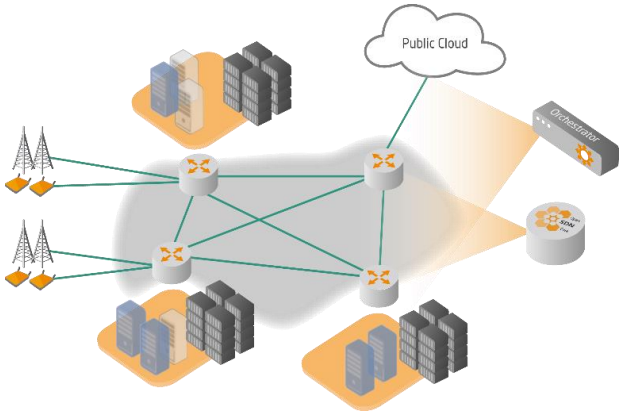


**Figure 37 OpenSDNCore Architecture**

OpenSDNCore is an extensive platform for SDN added value features for flexible routing, virtual environments and core network data paths. The components are: OpenFlow protocol version 4.1 controller and switch, together with a wide range of controller applications and a Network Configuration Protocol (Netconf) based configuration suite featuring both client and server.

The controller applications enable logic for establishment of dynamic data paths that can be used for backhaul control for dedicated networks, deep data plane programmability, Service Function Chaining and Mobile Mesh Networks using the BATMAN protocol.

The OpenSDNCore switch benefits from integrating with the Intel DPDK acceleration library, leading to performances of 8.9 Gb/s for forwarding traffic.

For a robust virtual network support, the toolkit also has a deep integration with the OpenStack Neutron routing component in which Neutron uses the OpenSDNCore switch to enable routing for provisioning or removing virtual machines or assigning floating IPs.

#### 3.6.2.4 Open Baton toolkit

The Open Baton Platform provides a comprehensive ETSI NFV Management and Orchestration (MANO)-compliant environment focusing his aim in increasing flexibility of the platform and extensibility of the functionalities.

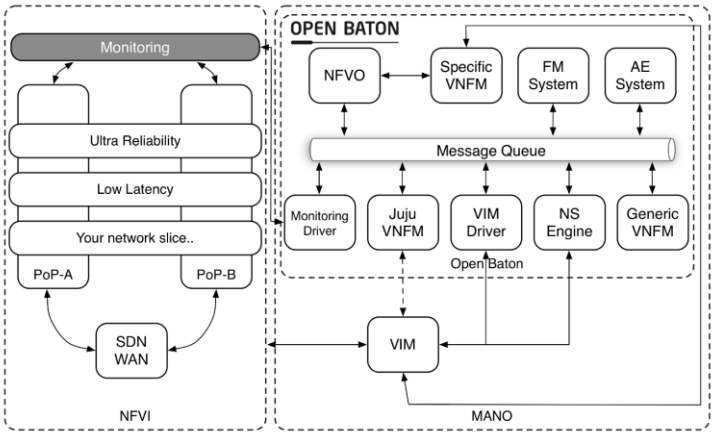


**Figure 38 Open Baton Architecture**

Open Baton enables virtual Network Services deployments on top of multiple cloud infrastructures and thereby builds a bridge between Cloud Computing Service Providers that have to understand Network Functions and Network Function Providers requiring the appropriate infrastructure support for their virtualization. In the last release, Open Baton significantly increased the number of available components that are part of the ecosystem and included new functionalities for simplifying the way Network Service developers deploy their services.

First of all, the Open Baton marketplace provides a public catalogue of VNFs that can easily be downloaded and started in your local environment. The marketplace has been integrated within the Open Baton dashboard for allowing the immediate deployment of complex Network Services (like the vIMS) with few clicks. Open Baton represents the first platform providing full interoperability between different VNFM solutions. It allows the instantiation of Network Services composed by Juju VNFs (basically charms) and Open Baton-compliant VNFs.

An extended set of new external modules (Services) are included in the platform and have been recently improved and extended such as a Network Slicing Engine capable of enforcing network requirements,

defined mainly as available bandwidth, onto virtual networks using SDN technologies deployed in the Network Function Virtualization Infrastructure (NFVI). The two other main services are the Fault Management System and the Auto Scaling Engine. They are respectively enforcing high availability and dynamic scaling, out of the box of a deployed Network Service.

#### 3.6.2.5 Integration of Radio Access Networks

Since the Open5GCore Release 1 in 2014 there have been multiple testbed deployments integrating commercial, off-the-shelf cells such as:

- Nokia Siemens Networks eNodeB
- Ericsson eNB (as part of 5Groningen)
- Huawei eNB (as part of 5Groningen)
- Ip.access LTE 245f Cell
- OpenAir Interface (both PCI-e card and PCB for radio hardware)
- AirSpan Airvelocity1200

To test the end-to-end system, programmable SIM cards with the milenage encryption algorithm are provisioned. A various range of android and apple mobile phones were tested with the end-to-end setups.

Regarding the Core Network setups, these have been deployed on VMWare, OpenStack and Open Baton and have been fully tested with the cell integrations. The Open5GCore has also been deployed on physical hardware directly ranging from blade servers to workstations and even ATOM powered devices such as Raspberry PI 3.

#### 3.6.2.6 5G Playground Testbed Instantiations

The software components are designed to run on any type of hardware and integrate with a multitude of access networks, as described in the previous section, the following setup is recommended to be used as part of the 5G!Pagoda end-to-end testbed, as already tested and run in Fraunhofer FOKUS.

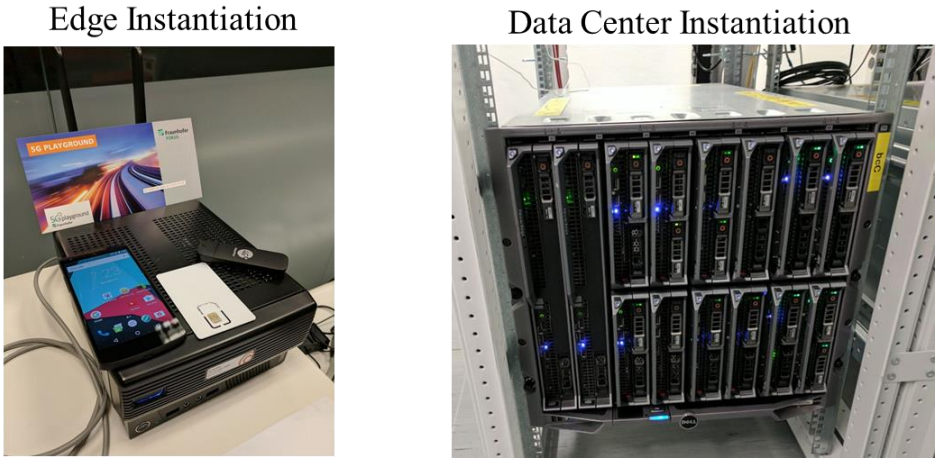


**Figure 39 Current testbed instantiations at Fraunhofer FOKUS**

These instances are currently extended with a FOKUS wide indoor network still with LTE access network for demonstration purposes. Additionally, an outdoor network using 5G radio technologies is planned in the next year and, depending on the availability, may be used for the final demonstrations of the 5G!Pagoda.

**Table 16 – Most significant technologies which integrate with 5G Playground**

| **Radio Access Networks** | |
|---|---|
| eNodeB | Commercially available from Nokia, Huawei, Ericsson and other vendors |
| Femto-cells | Commercially available from ip.access, airspan, etc. |
| Access points | CISCO Small Business |
| **Edge node** | |
| Small compute unit | Lenovo M900 Tiny |
| **Data center** | |
| Servers | PowerEdge M630 Blade Server |
| Casing | PE M1000e Rack (mit 1x CMC und 9x 12V Fans) |
| Switches | HP 5900AF-48XG-4QSFP+ Switch |
| OpenStack Support | RedHat Support Package |

The Fraunhofer FOKUS testbed provides infrastructure resources to execute different slices in such a way that they could interconnect with resources hosted on JP testbed.

## 3.7. Report on integration status

Both EU and JP testbeds provide a NFV infrastructure for allowing the on demand, deployments of virtual compute, storage and networking resources. This layer has been implemented using OpenStack on EU side, while the FLARE node on JP side as explained in Section 3.6.1.

Each testbed was assigned a private IP subnet that is used to provide connectivity to the virtual resources deployed for different use cases. The networks address ranges were assigned in such a way to avoid conflicts, so that each individual private networks can be interconnected within the 5G!Pagoda network.

VPN routers in Berlin and Tokyo are used to connect the subnets to the rest of the world and the rest of the 5G!Pagoda virtual network. In particular, on Europe side a VMWare ESXi virtual machine and the pfSense[4] VPN router VM have been used for the setup, while a hardware solution has been adopted on Japanese side with the commercial Yamaha RTX 1210 router. The VPN routers are using VPN tunnels via Internet to establish a direct connection between the subnets. These tunnels form a sort of overlay network on top of Internet. Firewall rules on the router protect the network from unwanted traffic and are used to enforce corporate rules of the private networks.

However, the current setup has showed some reliability issues regarding of packet loss. During the on-going investigation has been discovered that packets are lost on the way between Japan and Europe, which

[4] pfSense – OpenSource Security [https://www.pfsense.org/]

means that packets can be observed leaving the VPN device on the JP side but they are not arriving on EU side. While in the opposite direction, packets traveling from EU are reaching the JP side of the VPN.

FOKUS and UT are currently evaluating if these issues could be related to software incompatibilities between the used IPsec software and the Yamaha hardware router device. The planned solution is to replicate the same VPN software that is used on European side also on the Japanese side to overcome any compatibility issues.

# 4. Conclusion and Future plan

As the first output of WP5, this document gives the overview of the current status of the ignition testbeds and testbed integration between EU and Japan. Each ignition testbed has completed initial implementation and has been tested with use case scenarios. The functional validation and evaluation of the implemented components and interconnection among the components has been discussed in this deliverable. The development of the ignition testbeds will be continued to enhance the functional support for the usecase and to be compliant with 5G!Pagoda core platform design and development by WP2, WP3 and WP4. The setup of the testbed integration has been done and further integration for the components will be done applying 5G!Pagoda architecture and core components in order to support the use cases currently developing in the ignition testbeds.

## 4.1. Integration plan of the ignition testbeds

Section 2 introduced multiple ignition testbeds. Each of them is following the 5G!Pagoda architecture and adapting the core technologies being developed in WP3 and WP4. The synchronization of the use case development and core technology development is important and continuation of the co-development will be conducted. In order to have a high level overview of such co-development, the following Table 17 summarizes the future plans for the ignition testbeds showing:

- Use case to be addressed that is selected in D2.1 in WP2 and will be specified on evaluation in D5.2 in WP5;
- Included technologies mapping with the 5G!Pagoda architecture designed in WP2 and core technologies developed in WP3 and WP4, and
- Target location to be deployed (EU and/or Japan)

**Table 17 Integration plan**

| Slice Name | Use case addressed in the ignition testbeds | Included technologies | Target location (EU and/or JP) |
|---|---|---|---|
| ICN-CDN | Dynamic CDN As a service | Deep data plane programmability, Multi-domain orchestration, Slice stitching | EU/JP |
| IoT-Video slices | IoT and Video slices and a dynamic real-time slice for emergency handling on IoT site. | Dynamic slicing, multi-slice interoperation, Slice prioritization, Slice policy management, slice resource management, slice selection. | EU |
| Healthcare | Utilization of MVNO slice as an example of application to health care market | Realizing security NFV by Deep data plane programmability.<br>Form end-to-end slice from MVNO. | JP |

| | | | |
|---|---|---|---|
| | | Implement resources, security NFV etc held by MVNO on end-to-end slice.<br>Interaction with orchestrator. | |
| Deep data plane programmability | Deep data plane programmable mobile core network | FLARE, Open5GCore, Open Baton, O3, Multi-Domain Orchestrator | EU/JP |

## 4.2. Further development plan for the EU-JP testbed integration

The setup phase of the 5G!Pagoda testbed infrastructure was challenging as it involved the design of a federated architecture with different administrative domains and different network components. The fact that some of the network equipment used by FOKUS and UT developed some incompatibilities during the establishment of the IPsec connection, lead to extra steps that need to be implemented.

The deployment of the different customized slices for the different use cases will follow the guided architecture and use the heterogeneous networks in Berlin or in Tokyo. The integration process has already started with the ignition phase in order to prepare for further developments of the integrated testbeds. Thus, an initial evaluation of the feasibility of the use cases that will be deployed on top of the multi-slice architecture has been provided in this deliverable.

The ignition testbeds will be integrated as part of the 5G!Pagoda testbed up to months 24 of the project, while software optimization will be performed up to months 30 towards customization of the use cases in order to adapt the initial slices to the need of the market.

Furthermore, the orchestration framework defined in WP4 will be also integrated into the 5G!Pagoda testbed, firstly including different orchestration solutions depending on the location (Open Baton in Berlin and O3 in Tokyo) and afterwards with the development and integration of the end-to-end orchestrator. In this case, additional interoperability tests will be performed.