NEXT GENERATION INTERNET

Open Call 2

Integrating OpenIreland and COSMOS testbeds for delivering a cross-Atlantic Open Networking Solution
Deliverable 3: Experiment Results and Final Report

Authors	Frank Slyne, Diarmuid Collins, Sebastian Troia, Massimo Tornatore, Marco Ruffini
Due Date	28th of February 2022
Submission Date	28th of February 2022
Assigned Reviewers	
Keywords	

Contents

Table of Figures

Co-funded by the Horizon 2020 Framework Programme of the European Union

Tables

Equations

1 Abstract

Our vision in this project was to develop a federation, control plane, and monitoring framework for OpenIreland that is compatible with COSMOS, so that experimenters can easily operate between both testbeds. The first part of this vision involved upgrade of the Mininet-Optical emulation framework to support OpenIreland, so that users can easily move between emulation environment and real testbeds. We achieved this objective by integrating support for NETCONF SDN management protocol into Mininet-Optical, a prerequisite to using the OpenIreland testbed. In Experiment 1, which is detailed in this report, we have developed a GUI to simplify access and configuration of optical equipment in the OpenIreland testbed. This supports management, control, data collection and training of ML algorithms, and enables the possibility of performing experiments remotely. Experiment 2 focuses on the convergence of optical and wireless networks, and was carried out across the OpenIreland and COSMOS testbeds supported by the dedicated 10G testbed interconnection link made available in the later stages of the project. Overall, the tools developed, and experiments performed during this project will act as a foundation for great data extraction, supporting ML algorithm training and enabling future research into Optical and wireless networks using OpenIreland and COSMOS testbeds. The fact that researchers will be able to work seamlessly across the OpenIreland and COSMOS testbeds using these tools will boost the EU-US research activities and strengthen research outputs considerably. It is also envisaged that these initiatives, testbeds, and frameworks will bring closer together European and US researchers towards better international collaboration.

2 Project Vision

Our vision in this project was to develop a federation, control plane, and monitoring framework for OpenIreland that is compatible with COSMOS, so that experimenters can easily operate between both testbeds. The COSMOS team had already developed some tools, specifically Mininet-Optical, which was extended to OpenIreland by integrating support for NETCONF SDN management protocol into Mininet-Optical, a prerequisite to using the OpenIreland testbed. This extension was reported in Deliverable 2, at the midpoint of the project. Throughout this project, we have worked towards simplifying optical experimentation and optimisation by developing strategies for signal quality monitoring, so that data can be collected dynamically and fed to machine learning algorithms for the estimation of optical quality of transmission (QoT). One of the important aspects for the proposed collaboration with COSMOS is a focus on QoT, which is needed for efficient and reliable channel provisioning and the automation methods considered here are essential to achieve autonomous network operation—moving away from the mostly manual methods used today. In this report, Experiment 1 aims to simplify optical experimentation including management, control, support for data collection and training of ML algorithms, and enable the possibility of performing experiments remotely through the provision of a GUI. The GUI developed in Experiment 1 is the cumulation of this work, bringing together the individual entities that make up the OpenIreland optical testbed in a single graphical user interface. It is now possible to launch state-of-the-art machine learning Reinforcement-Learning agents supporting network optimization using this tool. Experiment 2 focuses on the convergence of optical and wireless networks, and was carried out across the OpenIreland and COSMOS testbeds supported by the dedicated 10G testbed interconnection link made available in the later stages of the project. This experiment involves resource orchestration and

Co-funded by the Horizon 2020
Framework Programme of the European Union

automatic service scaling across the OpenIreland and COSMOS testbed domains. We have carried out this experiment using 5G open source OAI5G gNB libraries with support for full stack connectivity and seamless bandwidth switching based on the 5G NR Carrier Bandwidth Part standard. This intelligent QoS controller function, running on the COSMOS testbed, can dynamically reserve capacity in the transmission networks, and spectrum in the wireless OAI 5G edge nodes running in the OpenIreland testbed based on observed diverging traffic patterns. Overall, the tools developed and experimented during this project will act as a foundation for greater data extraction supporting ML algorithm training and enable future research into Optical and wireless networks using OpenIreland and COSMOS testbeds. This will enhance European and US researchers towards better international collaboration. Furthermore, this work will enable European researchers to better interface with the international collaboration capabilities of COSMOS, under development in the COSMIC project.

3 Details on participants (both EU and US)

Prof. Marco Ruffini is Associate Professor and fellow at TCD. He is co-PI of both CONNECT Telecommunications Research Centre, and IPIC photonics integration centre. He is the PI of the OpenIreland research testbed. His recent work focuses on optical networks for 5G, including access network virtualisation, mesh access architectures for edge-cloud, access network sharing and QoT estimation. Prof. Ruffini has pioneered the concept of Virtual Dynamic Bandwidth Allocation for PON virtualisation, patenting its concept and bringing it to standardisation at the BroadBand Forum.

Prof. Dan Kilper is Chair of Future Communications at TCD and Adjunct faculty at the College of Optical Sciences at the University of Arizona. He is also an adjunct faculty member of Columbia University and a co-PI on the NSF COSMOS PAWR platform. He worked for more than a decade at Bell Labs where his pioneering work on optical performance monitoring contributed to the first transcontinental deployment of an optically switched (ROADM) fibre optic transmission system. While at Bell Labs he led multiple generations of simulation tool development for commercial optical systems and more recently co-developed the Mininet-Optical emulator with Marco Ruffini and Bob Lantz.

Ivan Seskar is the Chief Technologist at WINLAB, Rutgers University responsible for experimental systems and prototyping projects. He is the program director for the COSMOS project, the PI for the NSF GENI Wireless project, and the PI for the NSF CloudLab deployment at Rutgers. He has also been the co-PI and project manager for all three phases of the NSF-supported ORBIT mid-scale testbed project at WINLAB.

Prof. Massimo Tornatore has almost two decades of research experience on performance evaluation, optimization and machine learning applications for optical network design and management. He has pioneered application of Machine Learning and Transfer Learning for estimating the Quality of Transmission parameters (as, e.g., Signal-to-Noise ratio and to perform failure management in optical communication systems.

Dr. Diarmuid Collins holds a Ph.D. degree in computer science from the Trinity College Dublin, Ireland, an MSc degree in Software Engineering and Database Systems from NUI Galway, and a BSc in Commercial Software Development from Waterford Institute of Technology. He also has over 15 years' experience working in as a software engineer in industry. Dr Collins is currently the lead engineer on the Iris software defined radio testbed and Pervasive Nation IoT testbeds. Since he

Co-funded by the Horizon 2020
Framework Programme of the European Union

joined the CONNECT Group in Trinity College as a research fellow in 2016. In this project Dr. Collins is working broadly on the areas of 5G networks and cloud integration, network control and resource management, Network Functional Virtualisation (NFV), automation, testbed integration, and interconnection.

Dr. Frank Slyne is Research Fellow with Connect research group in TCD. Dr. Slyne has a Ph.D in Computer Science. Dr. Slyne's research interests include converged optical-wireless (LTE) metro-access architectures, disaggregation of legacy switch functions, slicing and resource pooling through SDN Control and Network Function Virtualisation. Most recently, he has been researching the optimisation of QoS and performance in a multi-tenant TDM PON. This has involved simulation, virtualisation and prototyping of novel schedulers in the downstream and upstream PON, making use of Xilinx FPGA, Intel DPDK data plane toolkit, and software radio stacks. Dr Slyne is broadly working on SDN and control plane, optical networks, testbed integration, NFV, virtualisation, Mininet-Optical, automation, and 5G networks in this project.

Dr. Sebastian Troia

Sebastian Troia received the bachelor’s, master’s, and Ph.D. degrees in telecommunication engineering from the Politecnico di Milano, Italy, in 2013, 2016, and 2020, respectively. He is currently an Assistant Professor at the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano. In 2018, he joined SWAN Networks, a spin-off of Politecnico di Milano, developing network orchestration and advanced algorithms for SDN and SD-WAN-based networks. His current research interests include the field of SD-WAN orchestration and control-plane architectures, machine-learning for communication networks, SDN, NFV and optical network automation.

Dr. Alan Diaz

Alan A. Díaz-Montiel is a postdoctoral researcher at CONNECT in Trinity College Dublin under the supervision of Dr Marco Ruffini. He holds a Ph.D. from TCD and his research is currently focused on Software-Defined Networking (SDN)-based solutions on top of Optical Data Centre Networks that can provide reconfigurable, low-latency signaling at the Physical Layer. L3/L2 & L2/L1 communication protocols, and virtualization of network components are among his main interests. His research interests include Optical Switching Networks in the context of Software Defined Networking, 5G Communication Technologies and Converged Access-Metro Networks.

4 Experiments implementation and results

We provide here self-contained details on the overall results and for the entire period of the project duration. Section 4.1 provides details of Experiment 1, incorporating an overview of the Quality of Transmission (QoT) estimation of optical systems. Section 4.2 provides details of Experiment 2, which includes an overview of the OpenIreland-COSMOS testbed interconnection link and performance, the OpenAirInterface (OAI) 5G SA core and gNB, and results from the control experiment over the inter-testbed link.

Co-funded by the Horizon 2020
Framework Programme of the European Union

4.1 Experiment 1: Quality of Transmission (QoT) estimation of optical systems

Experiment 1 focuses on the Quality of Transmission (QoT) estimation of optical systems, whereby we train Machine Learning models and carry out experiments across the OpenIreland testbed. For this purpose, we have applied the ONIS framework, already introduced in Deliverable D2. ONIS aims to optimize optical network resources by integrating full operational Machine Learning (ML) algorithms into the OpenIreland testbed. Specifically, the Intelligence is based on deep Reinforcement Learning (RL), which is a subfield of ML that aims to capture the most important features of a dynamic environment by deploying a learning agent (powered by deep learning algorithms) that interacts with it to achieve a given goal.
The goal of this experiment is to run DRL software agents through the ONIS framework, capable of establishing new optical channels (i.e., wavelengths) inside the OpenIreland Testbed on the basis of the QoT of the channels already deployed in the network. Figure 1 shows the ONIS framework and the current software agents. They are based on Proximal Policy Optimization (PPO2) [6], Actor Critic (A2C) [7] and Deep Q Network (DQN) [8] algorithms.

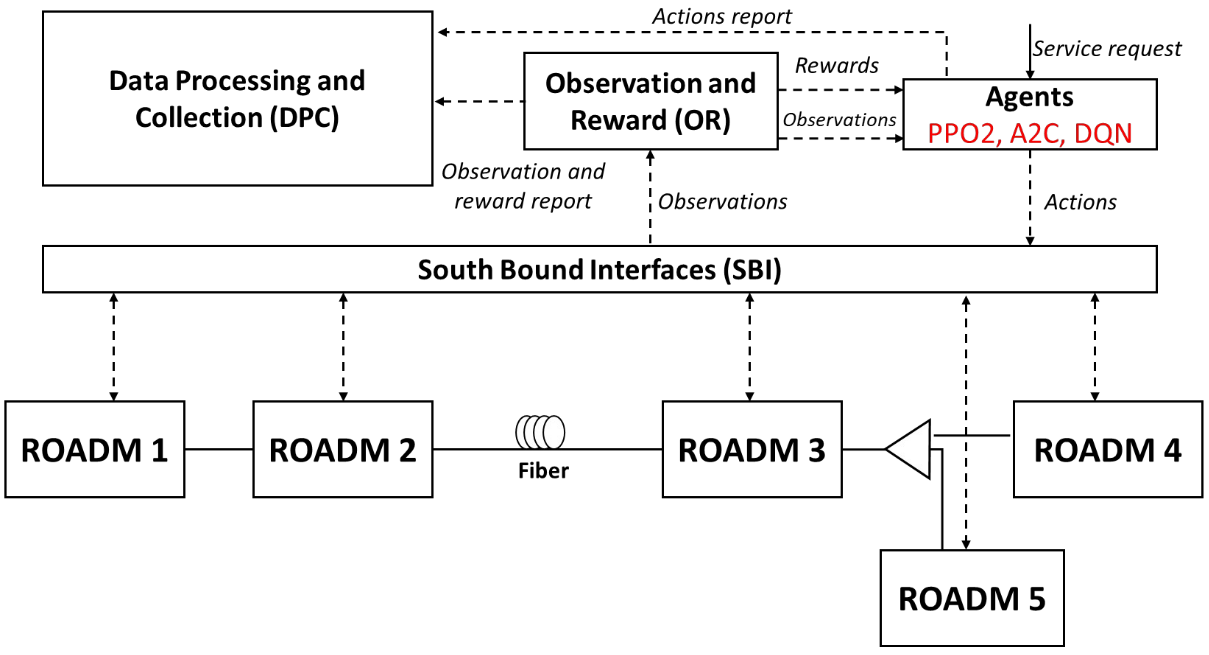


Figure 1 - ONIS framework with DRL software agents.

We are currently considering the OSNR degradation of each channel as a QoT parameter. Therefore, when installing a new channel, if this causes a degradation of the OSNR by -3 [dBm], then the channel is to be considered degraded, because the noise level has risen to the point of corrupting the received signal. The OSNR estimation is made by the Observation and Reward (OR) module with two methods: 1) OSA-based and 2) ROADM-based OSNR estimation. Both methods will be explored in the next section, but in a nutshell, the OSA method uses the optical spectrum analyser (OSA) to make an estimate of the signal and noise power values, and then derive the OSNR value. On the other hand, the second method uses the filters of the MUX and DeMUX inside the ROADM to filter the contributions of the power from the received electromagnetic field into a part measuring the noise and a part measuring the signal power.

Co-funded by the Horizon 2020 Framework Programme of the European Union

At each connection request, the agent decides to install a wavelength in the network. At the end of this installation, carried out through the SBI, the OR module estimates the OSNR of each optical channel to evaluate any degradation. If there has been no degradation, OR sends positive feedback to the agent, otherwise it sends negative feedback.
Given D(t) as the total number of degraded channels (max N) at a time (t). Then, D(t) is expressed in Equation 1 as follows:

$$D(t)=\sum_{i=1}^{N} d_i(t)\{d_i(t)=1\, if\ channel\, i\, is\, disrupted\}$$

Equation 1 - total number of degraded channels

The reward function is shown in Equation 2 as follows:

$$r(t)=1\, if\ D(t)=0\, r(t)=-2\cdot D(t)\, if\ D(t)>0$$

Equation 2 - Reward Function

In our experiments we are able to run 3 DRL agents capable of allocating 88 different wavelengths in the underlying optical network. Thanks to the automation functions developed in the SBI, software agents can learn directly in the field, without having to generate synthetic data from third-party software simulators.
Table 1 shows the performance of ONIS in terms of operation timings. In particular, we can see how this framework is able to run DRL algorithms in real time by carrying out an online training. An interesting result is highlighted by the estimation of the OSNR by the OSA. Clearly, it takes too long to estimate the OSNR and is therefore not usable. While the ROADM-based method is very fast and therefore suitable for the developed algorithms.

Table 1 - ONIS operation timings.

ONIS OPERATION TIMINGS		
	OSA-based OSNR computation	ROADM-based OSNR computation
Algorithm initialization	3.15 s	
Single step *channel opening + OSNR + reward*	1.31 s	
Episode (full spectrum filled)	**2400 s (40 mins)**	**182.2 s (3 mins)**
OSNR computation	25.84 s	1.18 s

Figure 2 shows the reward function trend after about 60 training episodes of the PPO2 agent. As can be seen from Equation 1, the number of degraded channels is inversely proportional to the reward function, consequently, if it increases, the number of degraded channels decreases, if instead it decreases then the number of degraded channels increases.

Co-funded by the Horizon 2020 Framework Programme of the European Union

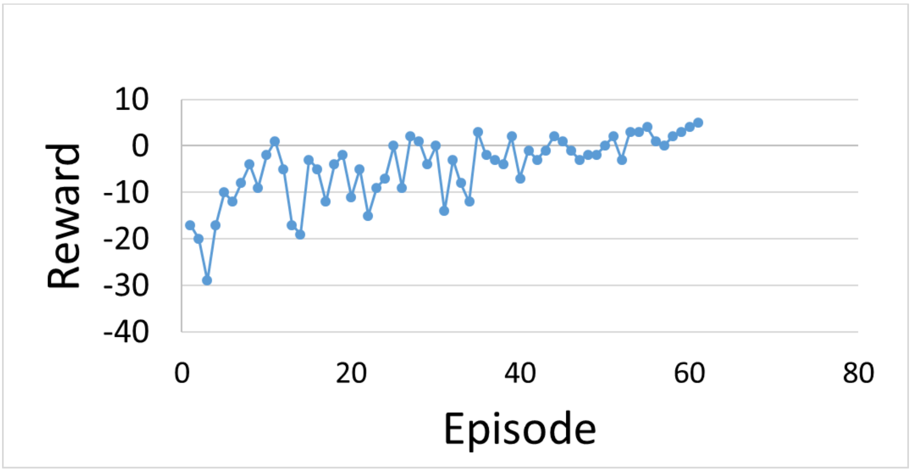
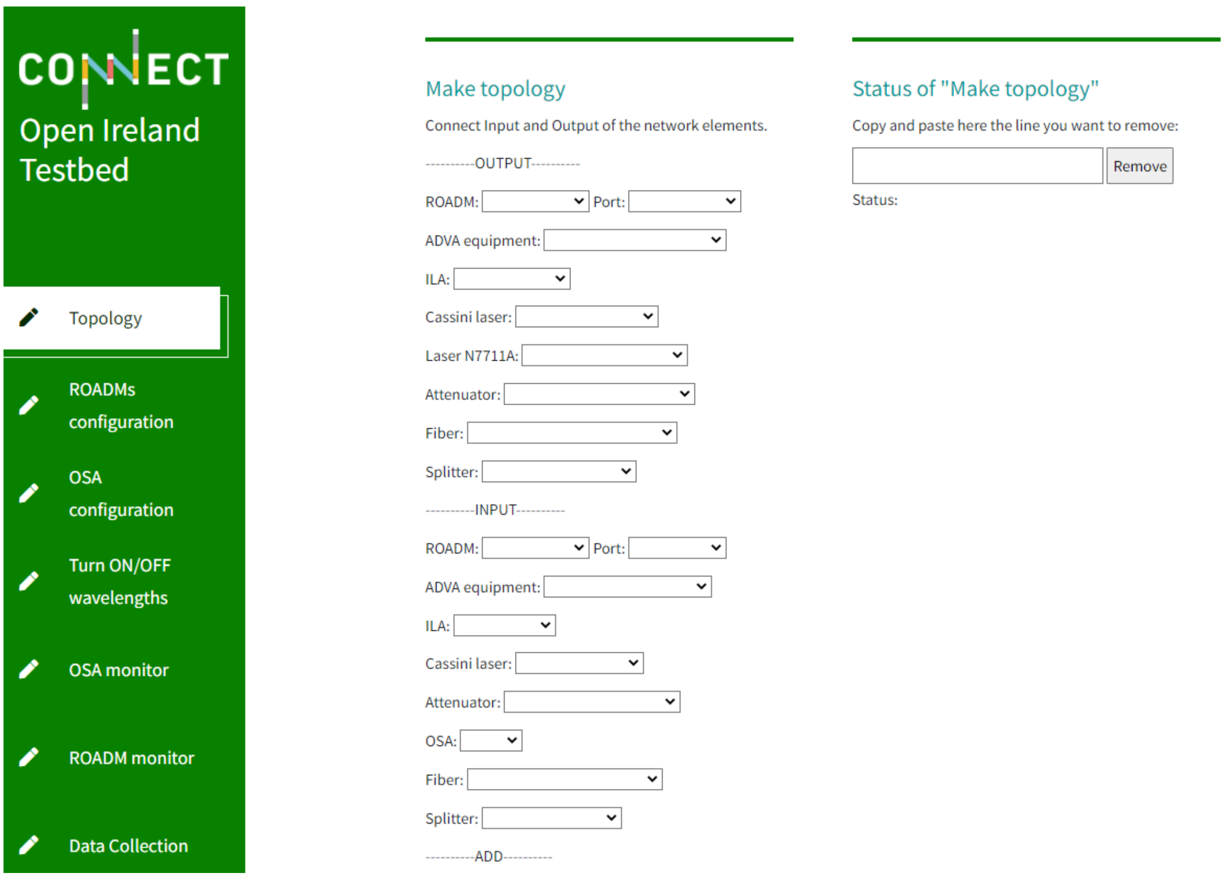


Figure 2 - Reward function for the PPO2 agent after 60 episodes.

4.1.1 Optical experimentation - OpenIreland Dashboard

The OpenIreland testbed consists of several complex optical network components, such as ROADMs and transponders. To simplify its management and enable the possibility of performing experiments remotely (for example from home), we have developed a dashboard with the aim of bringing together the individual entities that make up the testbed in a single graphical user interface. The dashboard in Figure 3 was developed in Python3 using the Flask library. It is part of the ONIS framework and uses the South Bound Interface (SBI) developed to automate the configurations of the optical components and thus perform multiple experiments simultaneously. The dashboard consists of 7 tabs: 1) Topology, 2) ROADMs configuration, 3) OSA configuration, 4) Turn ON/OFF wavelengths, 5) OSA monitor, 6) ROADM monitor and 7) Data Collection.

Co-funded by the Horizon 2020
Framework Programme of the European Union



Make topology

Connect Input and Output of the network elements.

---------OUTPUT---------

ROADM: Port:

ADVA equipment:

ILA:

Cassini laser:

Laser N7711A:

Attenuator:

Fiber:

Splitter:

---------INPUT---------

ROADM: Port:

ADVA equipment:

ILA:

Cassini laser:

Attenuator:

OSA:

Fiber:

Splitter:

---------ADD---------

Status of "Make topology"

Copy and paste here the line you want to remove:

Status:

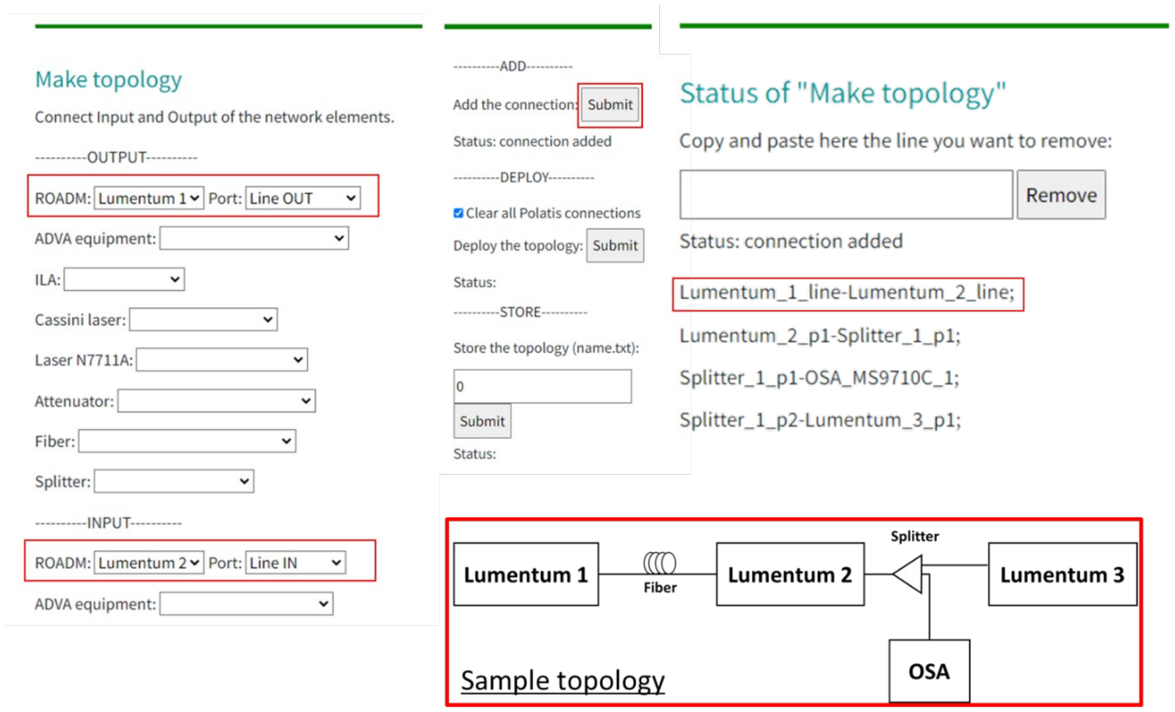
Figure 3 - OpenIreland Testbed Dashboard

4.1.2 Topology

The Topology tab allows the realization of the optical network topology inside the testbed. It uses a programmable optical fibre switch called "Polatis" where all the elements of the testbed are connected, namely: ROADMs, ILAs, OSA, Transponders, Attenuators, fibre reels, etc. Thanks to specific SBI functions, we can program the Polatis switch to create multiple network segments that make up the final topology. An example can be seen in Figure 4. First, we choose the network

Co-funded by the Horizon 2020 Framework Programme of the European Union

segment to add (ex. output of Lumentum 1 to input of Lumentum 2), then we deploy all the segments by clicking on the Deploy button.



Make topology

Connect Input and Output of the network elements.

----------OUTPUT----------

ROADM: Lumentum 1 Port: Line OUT

ADVA equipment:

ILA:

Cassini laser:

Laser N7711A:

Attenuator:

Fiber:

Splitter:

----------INPUT----------

ROADM: Lumentum 2 Port: Line IN

ADVA equipment:

----------ADD----------

Add the connection: Submit

Status: connection added

----------DEPLOY----------

☑ Clear all Polatis connections

Deploy the topology: Submit

Status:

----------STORE----------

Store the topology (name.txt):

0

Submit

Status:

Status of "Make topology"

Copy and paste here the line you want to remove:

Remove

Status: connection added

Lumentum_1_line-Lumentum_2_line;

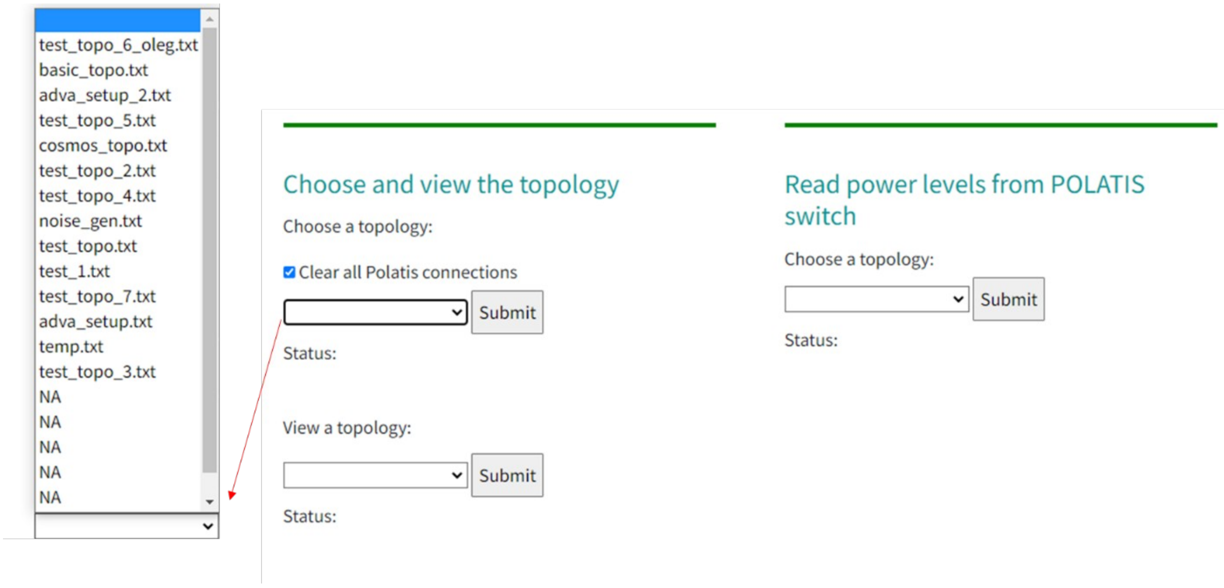
Lumentum_2_p1-Splitter_1_p1;

Splitter_1_p1-OSA_MS9710C_1;

Splitter_1_p2-Lumentum_3_p1;

Figure 4 - OpenIreland dashboard: topology tab (part 1).

Moreover, we can choose the topology to deploy from a stored catalogue and check the output power from each segment of the deployed topology, see Figure 5.



test_topo_6_oleg.txt
basic_topo.txt
adva_setup_2.txt
test_topo_5.txt
cosmos_topo.txt
test_topo_2.txt
test_topo_4.txt
noise_gen.txt
test_topo.txt
test_1.txt
test_topo_7.txt
adva_setup.txt
temp.txt
test_topo_3.txt
NA
NA
NA
NA
NA

Choose and view the topology

Choose a topology:

☑ Clear all Polatis connections

Submit

Status:

View a topology:

Submit

Status:

Read power levels from POLATIS switch

Choose a topology:

Submit

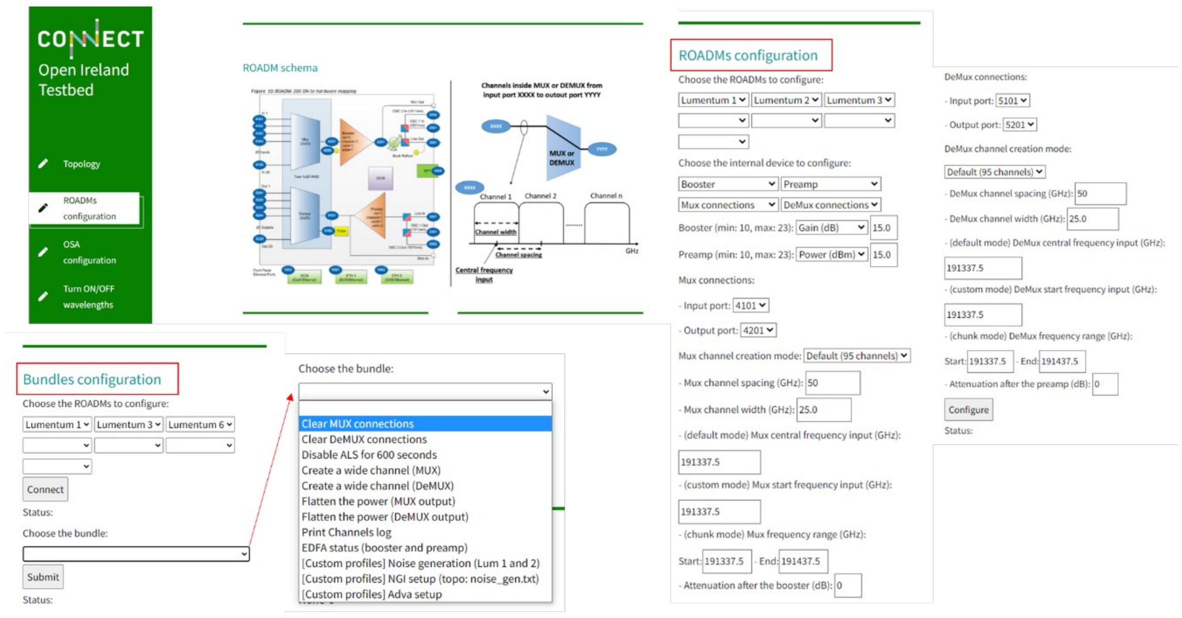
Status:

Figure 5 - OpenIreland dashboard: topology tab (part 2).

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.1.3 ROADM configuration

The ROADM configuration tab allows to perform a detailed configuration of each ROADM of the testbed. As shown in Figure 6, we can choose to deploy the same configuration for a set of ROADMs by configuring their MUX/DeMUX connections, boosters and preamps. Moreover, it is possible to deploy configuration bundles, which are set of functions that run all in one script. For instance, we can choose to clear all the MUX/DeMUX connections to a set of ROADMs, or we can code custom configuration profiles that can be deployed with a single click.



ROADMs configuration

Choose the ROADMs to configure:

Lumentum 1 | Lumentum 2 | Lumentum 3

Choose the internal device to configure:

Booster | Preamp

Mux connections | DeMux connections

Booster (min: 10, max: 23): Gain (dB) 15.0

Preamp (min: 10, max: 23): Power (dBm) 15.0

Mux connections:

- Input port: 4101
- Output port: 4201

Mux channel creation mode: Default (95 channels)

- Mux channel spacing (GHz): 50
- Mux channel width (GHz): 25.0
- (default mode) Mux central frequency input (GHz): 191337.5
- (custom mode) Mux start frequency input (GHz): 191337.5
- (chunk mode) Mux frequency range (GHz): Start: 191337.5 - End: 191437.5
- Attenuation after the booster (dB): 0

DeMux connections:

- Input port: 5101
- Output port: 5201

DeMux channel creation mode: Default (95 channels)

- DeMux channel spacing (GHz): 50
- DeMux channel width (GHz): 25.0
- (default mode) DeMux central frequency input (GHz): 191337.5
- (custom mode) DeMux start frequency input (GHz): 191337.5
- (chunk mode) DeMux frequency range (GHz): Start: 191337.5 - End: 191437.5
- Attenuation after the preamp (dB): 0

Configure

Status:

Bundles configuration

Choose the ROADMs to configure:

Lumentum 1 | Lumentum 3 | Lumentum 6

Connect

Status:

Choose the bundle:

Submit

Status:

Choose the bundle:

- Clear MUX connections
- Clear DeMUX connections
- Disable ALS for 600 seconds
- Create a wide channel (MUX)
- Create a wide channel (DeMUX)
- Flatten the power (MUX output)
- Flatten the power (DeMUX output)
- Print Channels log
- EDFA status (booster and preamp)
- [Custom profiles] Noise generation (Lum 1 and 2)
- [Custom profiles] NGI setup (topo: noise_gen.txt)
- [Custom profiles] Adva setup

Figure 6 - OpenIreland dashboard: ROADM configuration tab.

4.1.4 OSA configuration

The OSA configuration tab allows to configure OSAs (Instruction box in Figure 7) and collect data from them (query box in Fig. 7).

Co-funded by the Horizon 2020 Framework Programme of the European Union

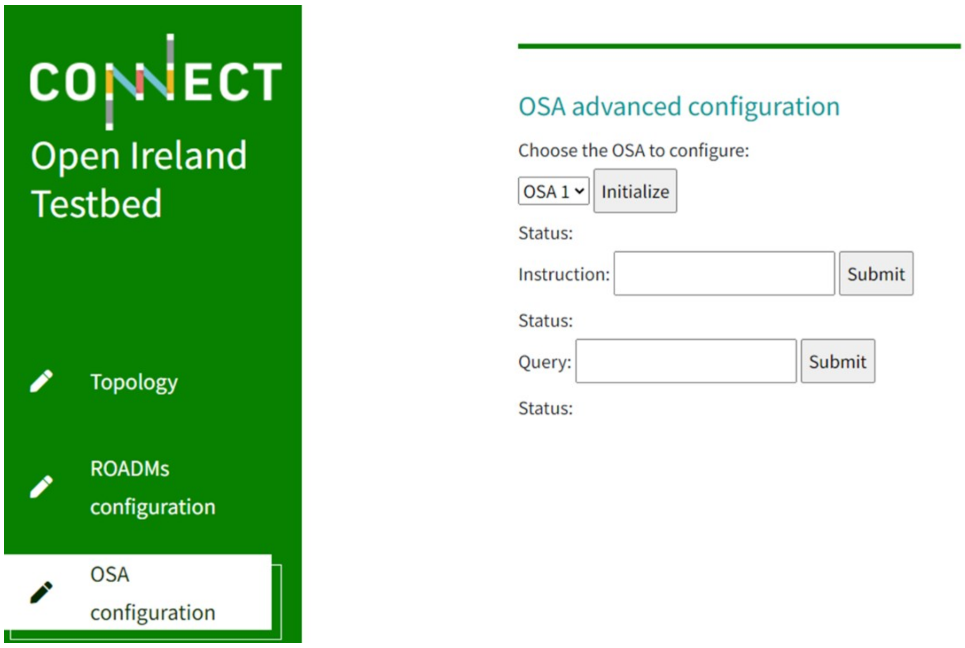
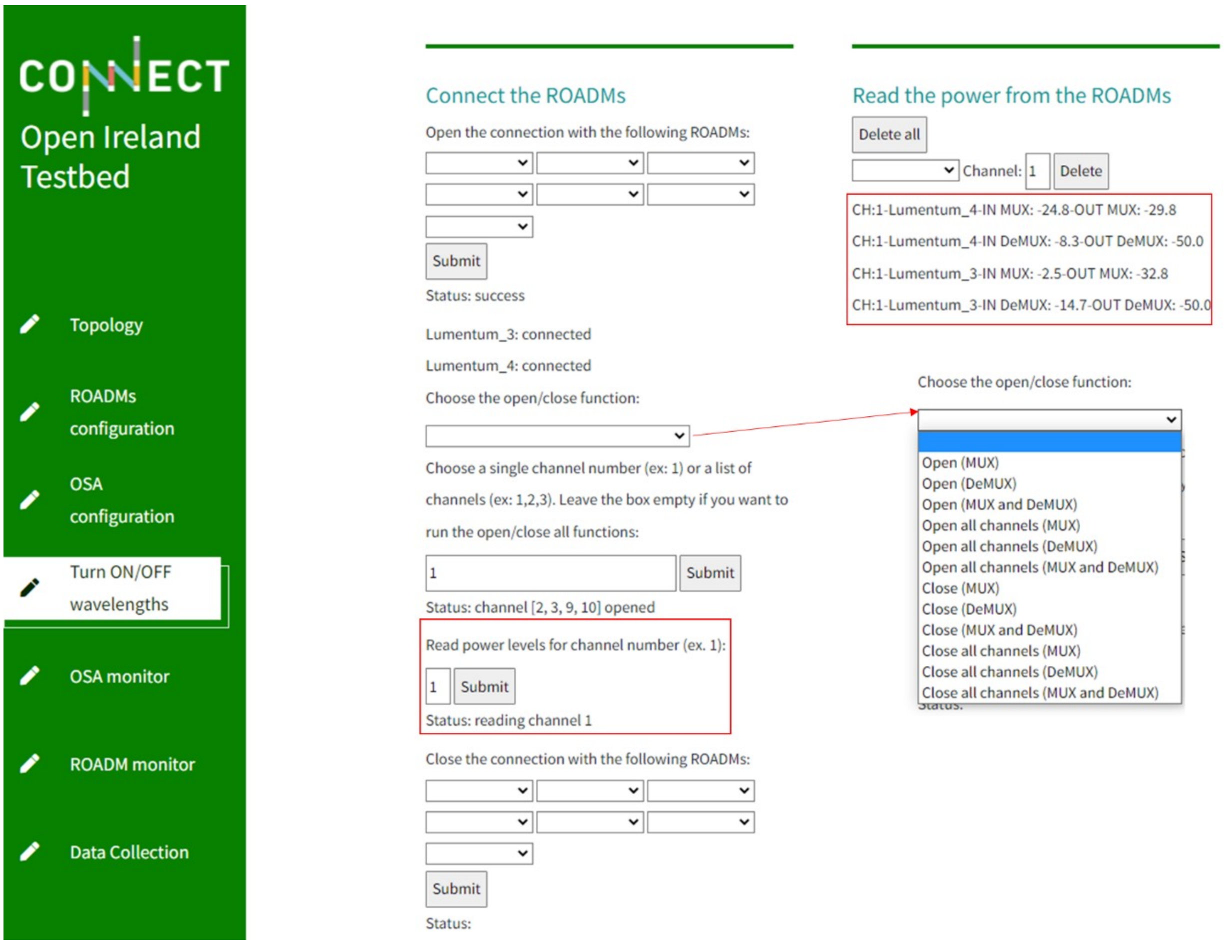


Figure 7: OpenIreland dashboard: OSA configuration tab.

4.1.5 Turn ON/OFF Wavelengths

This tab allows to connect to the ROADMs and handle the MUX and DeMUX channels. In particular, we can Open/Close MUX and DeMUX channels by specifying the ID number or just triggering the *all channels* options. Within the same tab, we can also read the power levels at the input/output of the MUX/DeMUX channels, see Figure 7.

Co-funded by the Horizon 2020 Framework Programme of the European Union


CONNECT
Open Ireland Testbed
Topology
ROADMs configuration
OSA configuration
Turn ON/OFF wavelengths
OSA monitor
ROADM monitor
Data Collection
Connect the ROADMs
Open the connection with the following ROADMs:
Submit
Choose the open/close function:
Choose a single channel number (ex: 1) or a list of channels (ex: 1,2,3). Leave the box empty if you want to run the open/close all functions:
Submit
Read power levels for channel number (ex. 1):
Submit
Close the connection with the following ROADMs:
Submit
Status:
Read the power from the ROADMs
Delete all
Channel:
Delete
Choose the open/close function:
Open (MUX)
Open (DeMUX)
Open (MUX and DeMUX)
Open all channels (MUX)
Open all channels (DeMUX)
Open all channels (MUX and DeMUX)
Close (MUX)
Close (DeMUX)
Close (MUX and DeMUX)
Close all channels (MUX)
Close all channels (DeMUX)
Close all channels (MUX and DeMUX)

Status: success

Lumentum_3: connected

Lumentum_4: connected

Status: channel [2, 3, 9, 10] opened

Status: reading channel 1

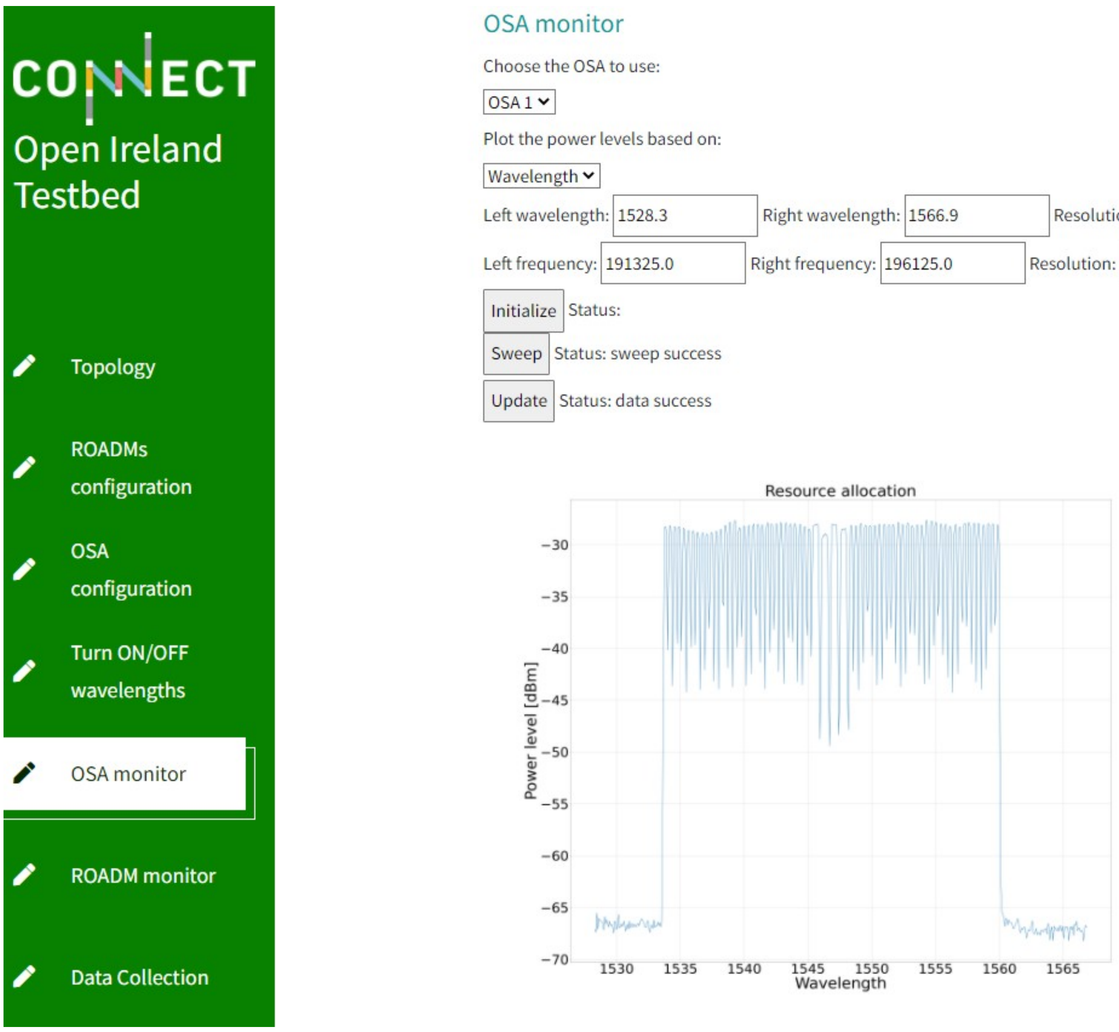
```
CH:1-Lumentum_4-IN MUX: -24.8-OUT MUX: -29.8
CH:1-Lumentum_4-IN DeMUX: -8.3-OUT DeMUX: -50.0
CH:1-Lumentum_3-IN MUX: -2.5-OUT MUX: -32.8
CH:1-Lumentum_3-IN DeMUX: -14.7-OUT DeMUX: -50.0
```

Figure 7 - OpenIreland dashboard: Turn ON/OFF Wavelengths tab.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.1.6 OSA monitor

The OSA monitor tab shown in Figure 8 is in charge of reporting the current OSA screen as a figure. This is one of the most important features of the dashboard since it allows to inspect what the OSA is actually receiving. We can trigger the *sweep* operation multiple times (i.e., refresh the OSA screen) and download the figure.



Choose the OSA to use: OSA 1

Plot the power levels based on: Wavelength

Left wavelength: 1528.3 Right wavelength: 1566.9 Resolution: 0.05

Left frequency: 191325.0 Right frequency: 196125.0 Resolution: 0.05

Status:

Status: sweep success

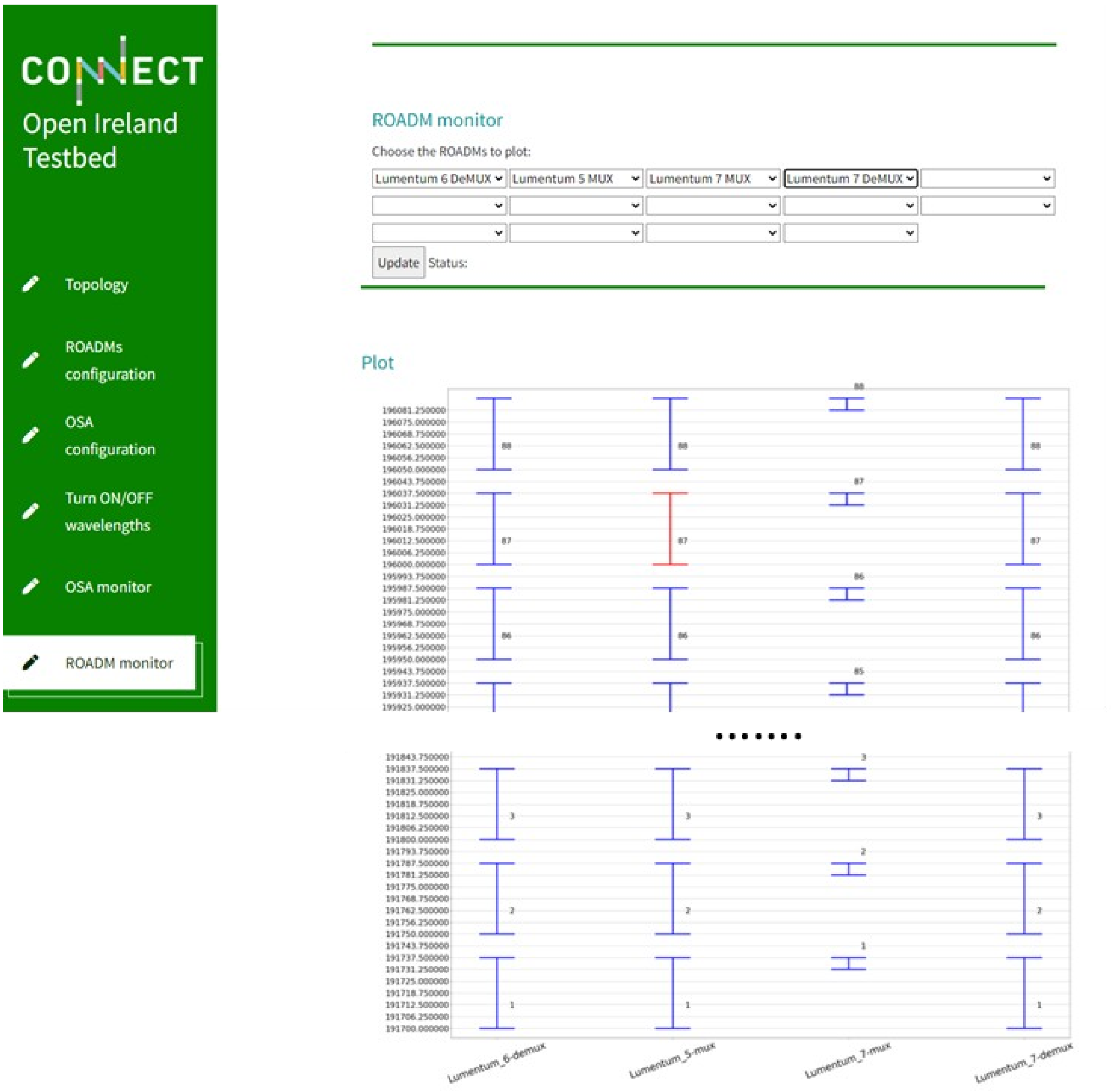
Status: data success

Figure 8 - OSA monitor

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.1.7 ROADM monitor

The ROADM monitor tab shows the channel gates in terms of frequency ranges. Once we select the ROAMDs MUX/DeMUX devices, we can check if the channels frequency ranges are consistent with the topology deployed and with the running experiment (see Figure 9). Moreover, we can see if the channel is closed (red bar) or opened (blue bar).



ROADM monitor

Choose the ROADMs to plot:

Lumentum 6 DeMUX | Lumentum 5 MUX | Lumentum 7 MUX | Lumentum 7 DeMUX

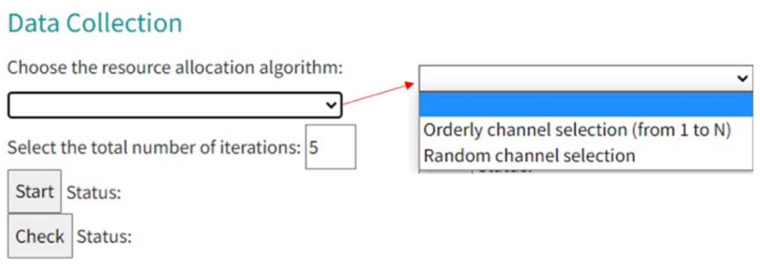
Status:

Figure 9 - Roadm Monitor

Co-funded by the Horizon 2020
Framework Programme of the European Union

4.1.8 Data Collection

The Data Collection tab is in charge of running resource allocation algorithms and collecting performance data from the testbed. The collected data can be used for training Machine-Learning algorithms but also for troubleshooting the testbed. We can freely code algorithms according to the type of resource allocation algorithm to run (e.g., random channel selection upon a connection request). In particular, each of these algorithms are coded to open/close channels of specific ROADMs (configured in advance with the dashboard) and collect performance parameters, for instance: OSNR values, noise power values, channel disruption rates, etc. Moreover, we can choose the total number of times the algorithm must be run and check the status of the data collection procedure, see Figure 10.


Data Collection

Choose the resource allocation algorithm:

Orderly channel selection (from 1 to N)
Random channel selection

Select the total number of iterations: 5

Figure 10 - OpenIreland dashboard: Data Collection tab.

4.1.9 Optical experimentation – Comb generator

Thanks to the dashboard presented in the previous section, we are able to test a large number of network scenarios. In this section, we focus on explaining how we manage to generate a set of signals from a broadband noise source (i.e., an EDFA). These signals can be used to simulate channel loading on the system, thus avoiding to use a large number of expensive coherent transceivers. Such comb generator can be used both for loading the system with channels and for measuring the OSNR (and its variation) on any of these channels. The comb is used in conjunction with a smaller number of optical coherent transceivers to test resource allocation algorithms based on online machine-learning, reinforcement learning, etc. In particular, the ONIS framework makes use of the comb generator to open/close channels for optimizing the optical resources. Figure 11 shows the practical setup of the ONSs generator.

Co-funded by the Horizon 2020
Framework Programme of the European Union

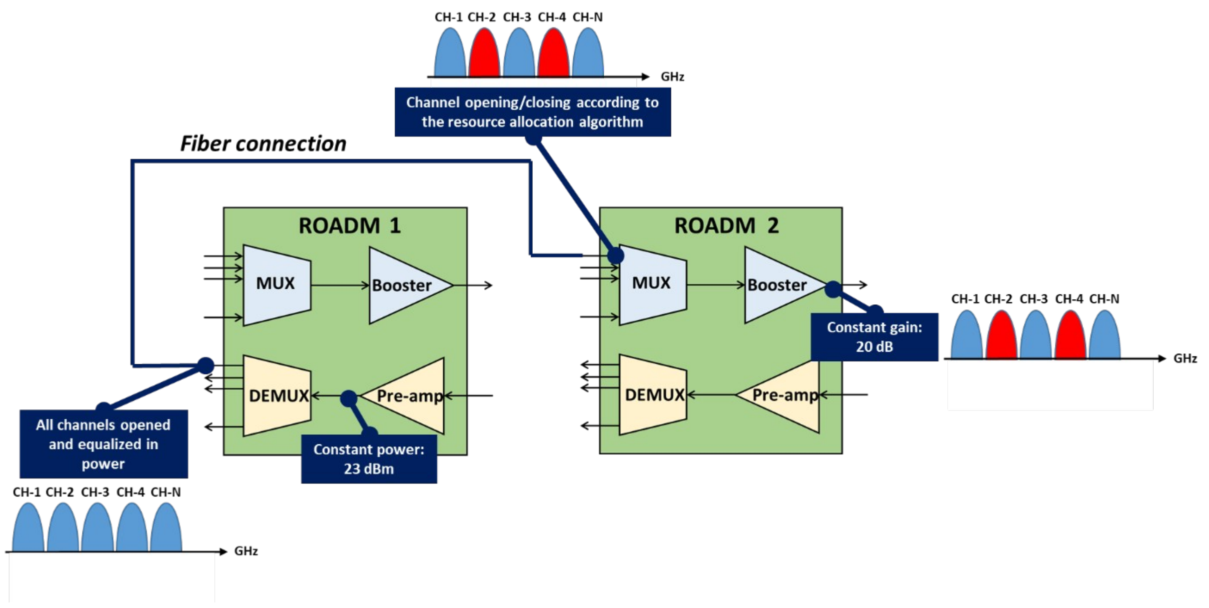


Figure 11 – Comb generator.

Table 2 shows the configuration details of the Comb generator. The preamp of the ROADM 1 is set to a high constant power (23 dBm) which will be the source of the broadband ASE noise power. After that, the DeMUX channels are used to shape the ASE noise into channels and to equalize the noise power across the entire C-band spectrum. Finally, the MUX acts as a channel activator, so the booster can further raise the power level of the channels.

Comb generator configuration	ROADM 1	ROADM 2
Booster	-	Constant gain 20 dB
Preamp	Constant power 23 dBm	-
Channel spacing (MUX and DeMUX)	50 GHz	50 GHz
Channel width (MUX and DeMUX)	37.5 GHz	37.5 GHz
Total number of channels	88	88
Starting frequency	191700 GHz	191700 GHz

Table 2: Configuration details for the Comb generator

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.1.10 Optical experimentation – OSNR estimation

The OSNR quantifies the ratio between the power of the signal and the power of the noise, which in optical amplified systems is mostly due to the EDFA ASE (while in non-amplified system is mostly due to the receiver noise and ultimately to the shot noise). The OSNR directly affects the system BER (which also depends on other factors such as the modulation format used). Since optical coherent transceivers typically use Forward Error Correction (FEC), the system will work up to a certain OSNR threshold, corresponding to a pre-FEC error rate of the order of $2*10^{-2}$. If the OSNR is lower than this FEC limit, then the FEC is not able to correct the bit errors and the optical link will fail. Such OSNR threshold is one of the key parameters that determines how far a wavelength can travel before regeneration.
OSNR can be mathematically determined once the pre-FEC BER is known, for a given modulation format. However, since our comb generator only simulates signals (being just spectrally shaped ASE noise), we have developed two methods to estimate the OSNR: **OSA-based** and **ROADM-based** OSNR estimation.

The **OSA-based OSNR estimation** makes use of the OSA to retrieve a screen shot of the spectrum at a given location and moment in time. Given this information, an algorithm can be designed to estimate the signal and noise power and thus the OSNR for each channel. Figure 12 shows the testbed setup for such OSNR estimation. Every time a new wavelength channel is added, we require a new OSA reading. This operation consists in carrying out a sweep across the c-band, and then reading the internal memory of the OSA where the data, displayed on the screen, are stored. The time to carry out a *sweep* operation depends on the resolution bandwidth set on the OSA and on the span (i.e., the amount of spectrum analysed). Once all the power values have been obtained, we run a function that makes an estimation of the OSNR by automatically distinguishing between the values of signal power and those of noise.

Co-funded by the Horizon 2020
Framework Programme of the European Union

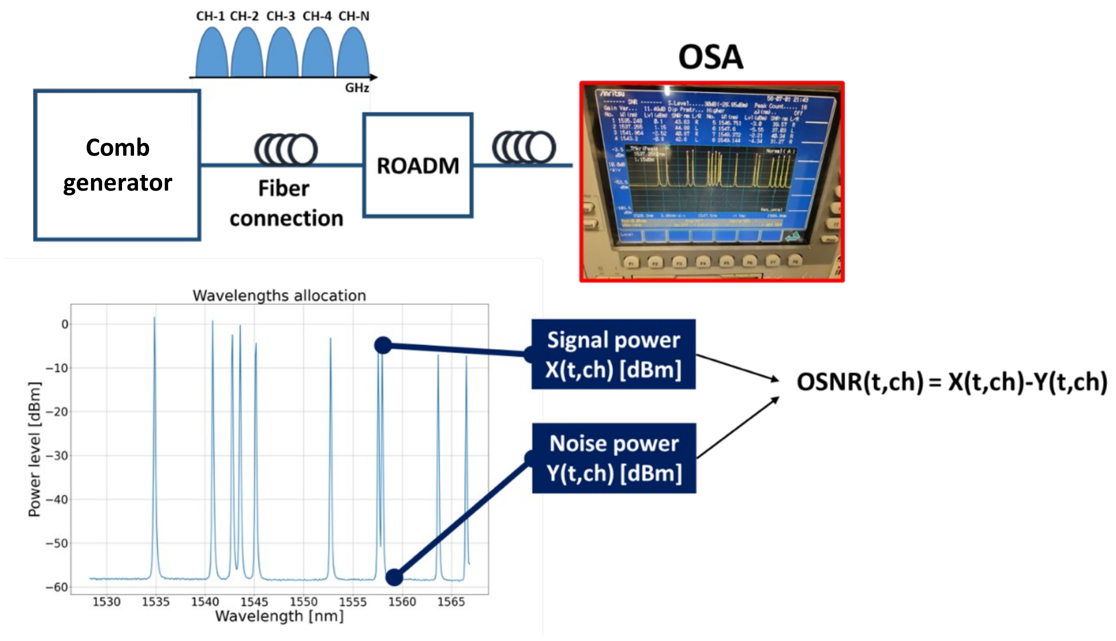


Figure 12 - OSA-based OSNR estimation setup.

The **ROADM-based OSNR estimation** makes use of the ROADM to collect both signal and noise power. Fig. 14 shows an example of the OSNR estimation. First, we generate the original signal from the comb generator. After that, the channels are transmitted along the optical network to finally reach a ROADM, which is responsible for obtaining the signal and noise power, and then calculating the OSNR. In particular, both powers are read by filtering the received signal on windows of contiguous and non-overlapping wavelengths. Figure 13 shows an example of how to configure the frequency ranges of the MUX and DeMUX in order to correctly filter the original and noise signals. The noise bandwidth is 1/N of that of the signal, therefore, the noise power value must be multiplied by a factor of N. In our experiment, the original signal bandwidth is 37.5 GHz, while the noise bandwidth is 6.25 GHz. It follows that N = 6.

Co-funded by the Horizon 2020 Framework Programme of the European Union

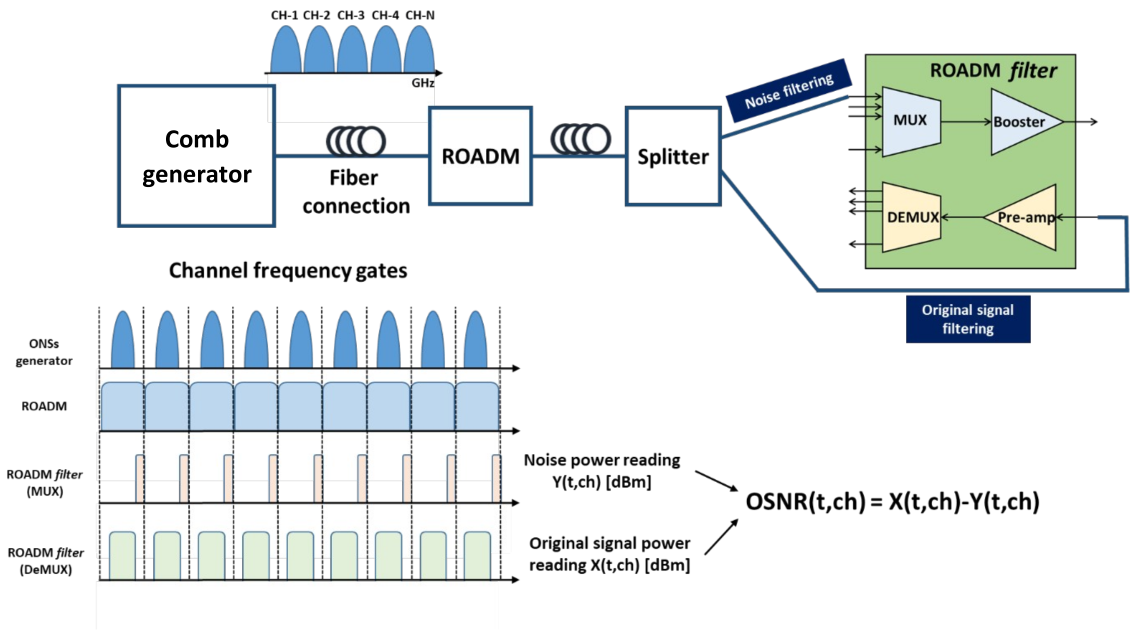


Figure 13 - ROADM-based OSNR estimation setup

We obtained –13 dBm as mean power of the signal across all the channels and –37.8 dBm as mean power of the noise. The mean value of the OSNR is 24.8 dBm.

Table 3 shows the performance of the testbed in terms of time computation. The OSA-based OSNR estimation depends on the resolution bandwidth and span. On the other hand, the ROADM-based OSNR estimation is constant since we query the MUX and DeMUX within an open NETCONF session.

TESTBED TIMINGS	
Single ROADM configuration (MUX, DeMUX, booster, preamp)	12.60 s
Single ROADM MUX/DeMUX reset (88 channels)	43.77 s
Single channel opening/closing	0.37 s
OSA power reading (all channels)	25.84 s
ROADM power reading (all channels into MUX and DeMUX)	1.18 s
OSNR computation (OSA-based)	**29.01 s**
OSNR computation (ROADM -based)	**2.28 s**

Table 3 - OpenIreland testbed timings

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.2 Experiment 2: Optical and wireless Testbed Convergence

Experiment 2 focuses on the convergence of optical and wireless networks, and was carried out across the OpenIreland and COSMOS testbeds supported by the dedicated 10G interconnection link. As outlined in D2, the goal of this experiment is to carry out a dynamic bandwidth allocation experiment similar to that implemented on a 4G system in the IEEE Communications Magazine 2019 paper: *FUTEBOL Control Framework: Enabling Experimentation in Convergent Optical, Wireless, and Cloud Infrastructures* [4]. This experiment aims to reproduce the core features of the Control Framework delivered in the FUTEBOL project and evaluate it through an experiment involving resource orchestration and automatic service scaling across the OpenIreland and COSMOS testbed domains. We have extended this experiment from 4G using srsRAN PHY layer connectivity, into 5G using open source OAI5G libraries with support for full stack connectivity.

The first part of this section provides a brief overview of the testbed interconnection link between COSMOS and OpenIreland testbeds, followed by set up details for the OpenAirInterface 5G SA core and gNB, and finally results from running a control experiment over the testbed interconnection link.

4.2.1 OpenIreland-COSMOS Testbed Interconnection

4.2.2 Interconnection: OpenIreland to COSMOS

The interconnection link between OpenIreland and COSMOS testbed was a complex procedure which took 12 months to organise and deploy as it required involvement from many networking entities distributed across the world. The main parties involved included:

- CONNECT Centre members from Trinity College,
- HEANet (Irelands Nation Education & Research Network),
- GÉANT (the pan-European data network for the research and education community), Internet2 (the United States not-for-profit computer networking consortium led by members from the research and education communities, industry, and government),
- Rutgers University,
- and finally terminating at the COSMOS testbed in Columbia University, Manhattan.

HEANet provide an S-TAG on their circuit between the CONNECT Centre at Trinity College and across the GEANT network (GEANT Dublin mx1.dub.ie and GEANT Amserdam mx1.ams.nl). The GEANT circuit has been provisioned as a GEANT plus circuit which is a best effort service, with no guarantee of bandwidth. Similar capabilities have been provisioned on the Internet2, Rutgers University, and COSMOS circuits. Combined, the expectation is that OpenIreland interconnection should be able to get 10Gbit/s in/out between and the OpenIreland and GEANT exit link, and the COSMOS, Rutgers University, and Internet2, with best effort server over third party networks, and across the transatlantic links. Figure 14 illustrates the networking links supporting the OpenIreland to COSMOS network interconnection and Inter-testbed Experimentation.

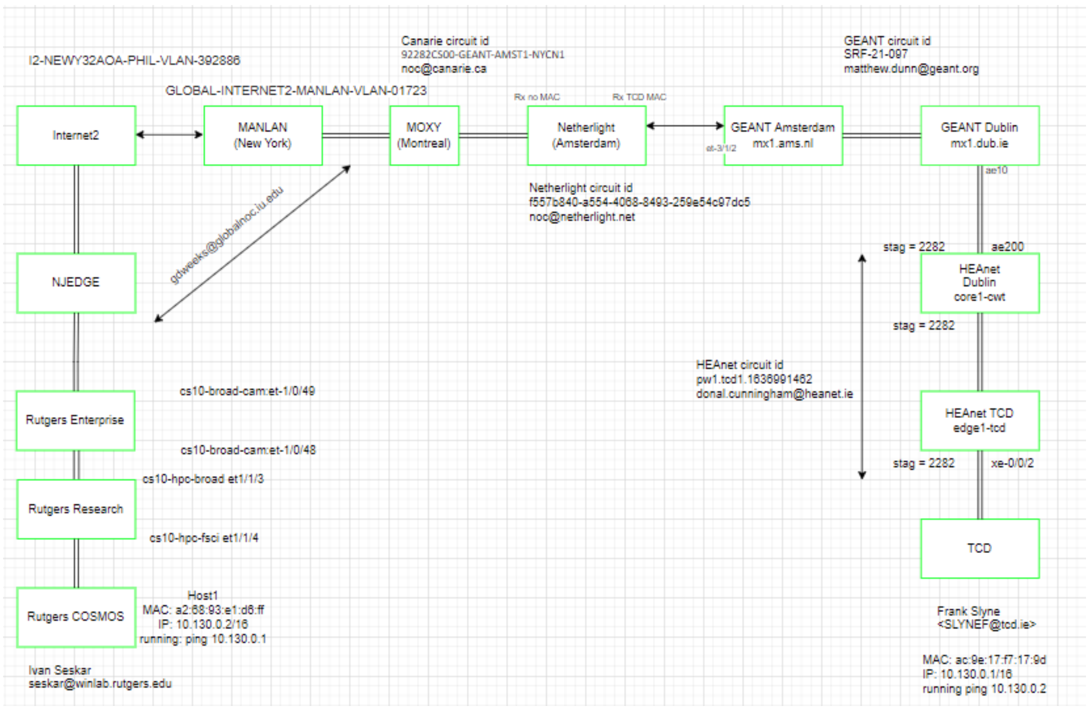
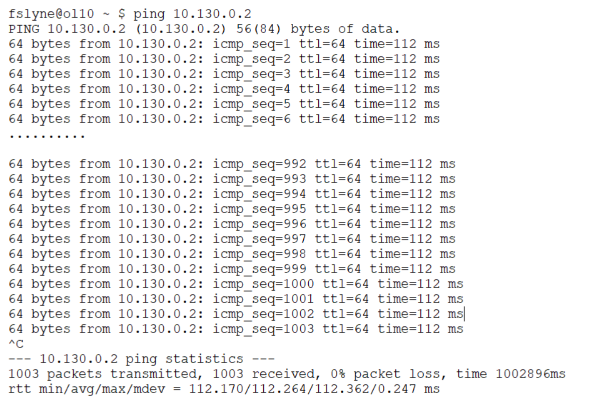


Figure 14 -Inter-testbed Experimentation - OpenIreland to COSMOS network interconnection

4.2.3 OpenIreland-COSMOS Testbed Interconnection: Initial Results

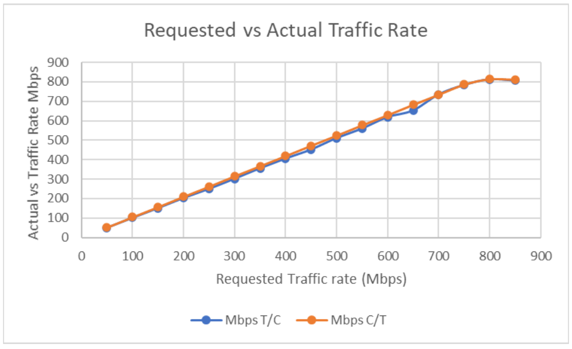
As shown in Figure 14 above, the testbed interconnection link terminates at a virtual machine in Rutgers-COSMOS, and at a bare metal server located in the OpenIreland testbed at Trinity College Dublin. Some of the results gathered from the OpenIreland-COSMOS testbed interconnection are show in Figure 15. The *Latency* observed on the transcontinental link using ping is 112ms, which is shown in Figure 15 (a). Figure 15(b) illustrates the *Requested vs Actual UDP Traffic* rate on the link, and performance received. Based on this graph, we were able to achieve just under 1Gb/s bidirectional link speed for UDP traffic across the inter-testbed link. The graphs in Figure 15 (c) exemplifies the UDP *Packet Loss v Traffic Rate*, which is showing 0% up to 750Mbps, at which point it starts to increase exponentially, hitting 8% packet loss when traffic reaches 850Mbps. This suggests we have hit the maximum traffic load possible on the link at that time. The *Jitter v Traffic Rate* shown in Figure 15(d), offers a view of the network congestion, timing drift and route changes which can take place across a transatlantic testbed interconnection link traversing many different networks. Figure 15(e) shows the results from an iperf test using the TCP protocol at 72.7Mbps, which is roughly 10 times lower than UDP traffic on over the same link. Depending on the experiment being performed, the results shown in Figure 15 might need to be factored into experimentation by future experimenters. However, for the purposes of our experiments in this project, the results are negligible and will not be considered further. Finally, Figure 15 (f) shows the parameters set to improve performance on the transatlantic link including IP packet memory allocation information, and default and maximum send/receive window size on the transatlantic link.

Co-funded by the Horizon 2020 Framework Programme of the European Union

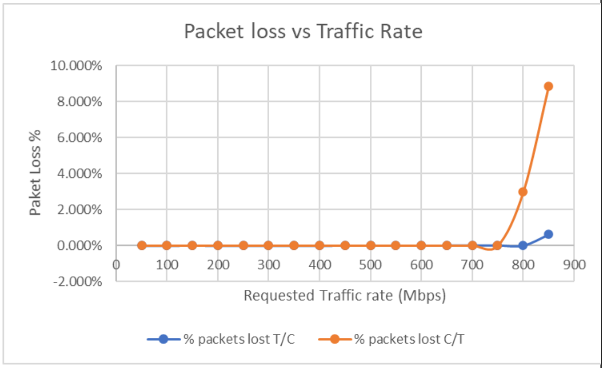


```
fslyne@ol10 ~ $ ping 10.130.0.2
PING 10.130.0.2 (10.130.0.2) 56(84) bytes of data.
64 bytes from 10.130.0.2: icmp_seq=1 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=2 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=3 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=4 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=5 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=6 ttl=64 time=112 ms
..........

64 bytes from 10.130.0.2: icmp_seq=992 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=993 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=994 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=995 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=996 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=997 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=998 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=999 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=1000 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=1001 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=1002 ttl=64 time=112 ms
64 bytes from 10.130.0.2: icmp_seq=1003 ttl=64 time=112 ms
^C
--- 10.130.0.2 ping statistics ---
1003 packets transmitted, 1003 received, 0% packet loss, time 1002896ms
rtt min/avg/max/mdev = 112.170/112.264/112.362/0.247 ms
```

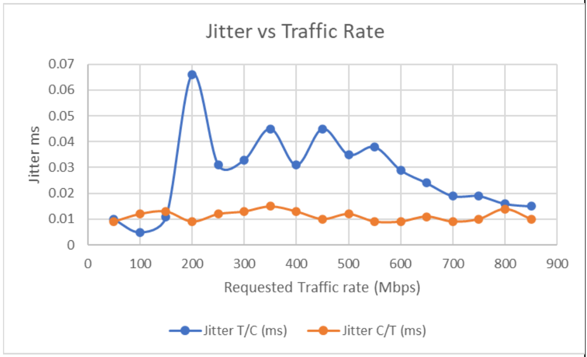
(a) Latency performance



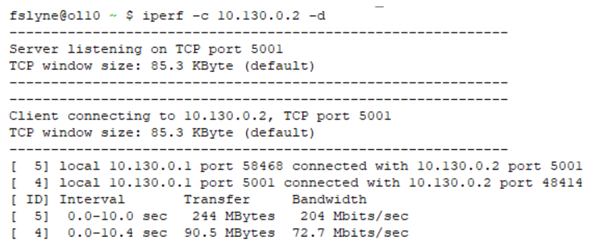
(b) UDP Performance



(c) Packet Loss vs Traffic Rate

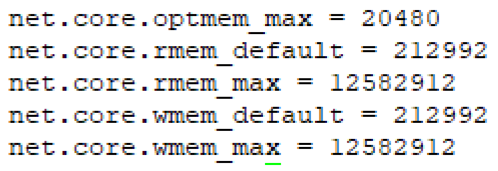


(d) Jitter vs Traffic Rate



```
fslyne@ol10 ~ $ iperf -c 10.130.0.2 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 10.130.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.130.0.1 port 58468 connected with 10.130.0.2 port 5001
[  4] local 10.130.0.1 port 5001 connected with 10.130.0.2 port 48414
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   244 MBytes   204 Mbits/sec
[  4]  0.0-10.4 sec  90.5 MBytes  72.7 Mbits/sec
```

(e) TCP Performance



```
net.core.optmem_max = 20480
net.core.rmem_default = 212992
net.core.rmem_max = 12582912
net.core.wmem_default = 212992
net.core.wmem_max = 12582912
```

(f) Defines the IP packet memory allocation, and default and maximum send/receive window size on the transatlantic link.

Figure 15 -Performance results across the testbed OpenIreland-COSMOS interconnection link

4.2.4 OpenIreland-COSMOS Control and Data plane Overview

The circuit built by HEANet included a S-TAG (Service VLAN tag) VLAN with VLAN-id 2282. Originally, we intended to use internal C-TAGS (customer tag), with double tagging mechanism based on the standard 802.1ad, that would allow us to create private networks for data and control planes between the COSMOS and TCD testbeds. Here a TPID of 0x88a8 is used for the outer S-Tag, followed by the C-Tag allowing us to distinguish data and control plane traffic at internal switches. Using different VLAN tag for each customer we can separate the traffic from different customers and also transparently transfer it throughout the service provider network. However, using C-TAGS would have made creating data and control planes inflexible since we would have had to rely on the intervention and support of HEAnet and the intermediary network providers to any changes.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Instead, the approach we took was to use our own encapsulation mechanisms, which are standard tunnelling techniques supported by the Linux kernel. A number of tunnelling techniques are supported by the Linux kernel such as Ip-in-ip, macvlan, vxlan, 802.1Q. The preferred choice would be IEEE 802.1Q tunneling (aka Q-in-Q), a standard technique used by Metro Ethernet providers as a layer 2 VPN for customers. In Q-in-Q tunnelling, the provider puts an 802.1Q tag on all the frames that it receives from a customer with a unique VLAN tag. This would enable TCD and COSMOS to use the outer S-tag with VLAN-id 2282, with two+ inner C-tags between OpenIreland and the COSMOS testbed. For example, C-tag vlan 1 would be used for the control plane, and C-tag vlan 2 for data plane traffic.

```
sudo tcpdump -i Ethernet24 -nn -e  vlan

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on Ethernet24, link-type EN10MB (Ethernet), capture size 262144 bytes

15:16:41.124227 ac:9e:17:f7:17:9d > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 64: vlan 2282, p 0, ethertype ARP, Request who-has 1.1.1.1 tell 10.10.10.52, length 46

15:16:42.148183 ac:9e:17:f7:17:9d > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 64: vlan 2282, p 0, ethertype ARP, Request who-has 1.1.1.1 tell 10.10.10.52, length 46

15:16:44.196197 ac:9e:17:f7:17:9d > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 64: vlan 2282, p 0, ethertype ARP, Request who-has 1.1.1.1 tell 10.10.10.52, length 46

15:16:45.220143 ac:9e:17:f7:17:9d > ff:ff:ff:ff:ff:ff, ethertype 802.1Q (0x8100), length 64: vlan 2282, p 0, ethertype ARP, Request who-has 1.1.1.1 tell 10.10.10.52, length 46
```

Figure 16 - Sample tcpdump Debug Output for continuous ping over vlan 2282

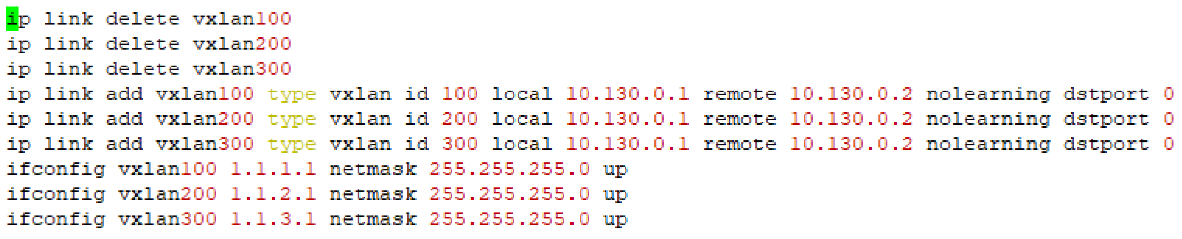
In practice, 802.1Q Q-in-Q tunnelling had some issues in between TCD and COSMOS. Traffic was been transmitted correctly in the direction TCD to COSMOS, but not in the opposite direction. Arp requests were being sent from TCD to COSMOS, but the corresponding Arp responses were being generated (and visible to tcpdump), but were not being received by TCD. See 802.1Q Arp request in Figure 16 above. Given the many hops through which the data must traverse, it was decided to use an alternate encapsulation and tunnelling technique, namely, vxlan. VXLAN is a VLAN-like encapsulation technique that encapsulates layer 2 Ethernet frames into UDP datagrams. Vxlan which uses UDP encapsulation of lower level ethernet data, worked immediately over the outer e-line (C-Tag) network.

Supported by Vxlan, we were able to create a number of distinct networks over the OpenIreland COSMOS testbed interconnection. These networks are ideal for separating control and data planes for network experiments. In the following, we show the IP address ranges of the control and data planes between the COSMOS and TCD servers on the testbed interconnection link.

- 1.1.1.0/24 - vxlan 1 - control plane
- 1.1.2.0/24 - vxlan 2 - data plane 1
- 1.1.3.0/24 - vxlan 3 - data plane 2
- 1.1.4.0/24 - vxlan 4 - storage, or MAAS etc for OpenStack

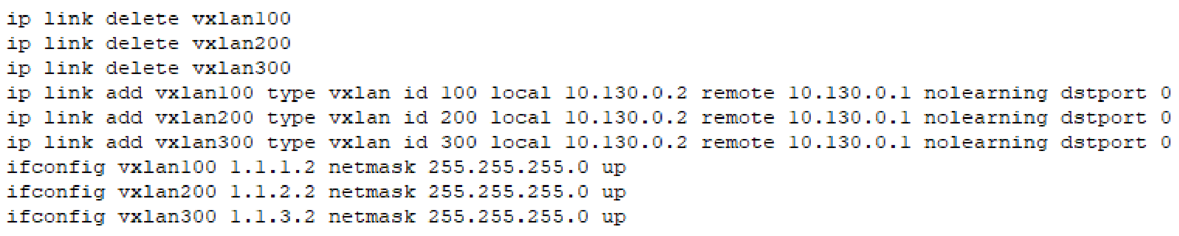
Co-funded by the Horizon 2020
Framework Programme of the European Union

In Figure 17 and Figure 18 below, we demonstrate how we create the separate data and control plane networks out of the vxlan's described above. Notice how, for both ends of the vxlan connectivity, we refer to the local and remote IP addresses of the outer C-tagged network. The *ip link add vxlan* creates new Linux network interfaces vxlan100, vxlan200 and vxlan300 to which the data and control plane network addresses are assigned.



```
ip link delete vxlan100
ip link delete vxlan200
ip link delete vxlan300
ip link add vxlan100 type vxlan id 100 local 10.130.0.1 remote 10.130.0.2 nolearning dstport 0
ip link add vxlan200 type vxlan id 200 local 10.130.0.1 remote 10.130.0.2 nolearning dstport 0
ip link add vxlan300 type vxlan id 300 local 10.130.0.1 remote 10.130.0.2 nolearning dstport 0
ifconfig vxlan100 1.1.1.1 netmask 255.255.255.0 up
ifconfig vxlan200 1.1.2.1 netmask 255.255.255.0 up
ifconfig vxlan300 1.1.3.1 netmask 255.255.255.0 up
```

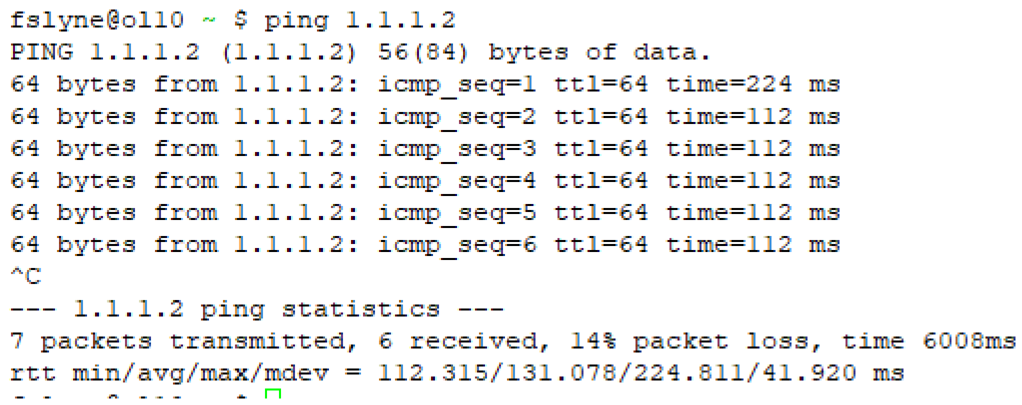
Figure 17 - Provisioning of data and control planes on TCD testbed



```
ip link delete vxlan100
ip link delete vxlan200
ip link delete vxlan300
ip link add vxlan100 type vxlan id 100 local 10.130.0.2 remote 10.130.0.1 nolearning dstport 0
ip link add vxlan200 type vxlan id 200 local 10.130.0.2 remote 10.130.0.1 nolearning dstport 0
ip link add vxlan300 type vxlan id 300 local 10.130.0.2 remote 10.130.0.1 nolearning dstport 0
ifconfig vxlan100 1.1.1.2 netmask 255.255.255.0 up
ifconfig vxlan200 1.1.2.2 netmask 255.255.255.0 up
ifconfig vxlan300 1.1.3.2 netmask 255.255.255.0 up
```

Figure 18 -Provisioning of data and control planes on COSMOS testbed

Note in Figure 19 and Figure 20, that the round trip latency of 112 ms. On the VXLAN network, the measured RTT between the new local and remote network interfaces is the same as that of the outer network (112 ms)



```
fslyne@ol10 ~ $ ping 1.1.1.2
PING 1.1.1.2 (1.1.1.2) 56(84) bytes of data.
64 bytes from 1.1.1.2: icmp_seq=1 ttl=64 time=224 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=112 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=64 time=112 ms
64 bytes from 1.1.1.2: icmp_seq=4 ttl=64 time=112 ms
64 bytes from 1.1.1.2: icmp_seq=5 ttl=64 time=112 ms
64 bytes from 1.1.1.2: icmp_seq=6 ttl=64 time=112 ms
^C
--- 1.1.1.2 ping statistics ---
7 packets transmitted, 6 received, 14% packet loss, time 6008ms
rtt min/avg/max/mdev = 112.315/131.078/224.811/41.920 ms
```

Figure 19 -Ping round trip delay between control plane interfaces.

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
diarmuid@repo-i:~$ ping 1.1.2.1
PING 1.1.2.1 (1.1.2.1) 56(84) bytes of data.
64 bytes from 1.1.2.1: icmp_seq=1 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=2 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=3 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=4 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=5 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=6 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=7 ttl=64 time=112 ms
64 bytes from 1.1.2.1: icmp_seq=8 ttl=64 time=112 ms
^C
--- 1.1.2.1 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7009ms
rtt min/avg/max/mdev = 112.267/112.313/112.442/0.049 ms
```

Figure 20 -Ping round trip delay between data plane interfaces.

Finally, once the experimentation is finished, we can easily remove the data and control plane networks, as show in Figure 21.

```
ip link delete vxlan100
ip link delete vxlan200
ip link delete vxlan300
```

Figure 21 - Removing of data and control planes on TCD and COSMOS testbed (same commands).

4.2.5 OpenAirInterface 5G Standalone (SA) – gNB and 5G Core

To support 5G experimentation we are currently using the OAI SA core, which is available for download at https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git. The OAI 5G SA components available in the core as docker containers include AMF, SMF, NRF, SPGW-U-TINY, AUSF, UDM, and UDR. OAI core functionality is being updated and extended regularly. Figure 22 shows the operating system dependencies required to support the OAI 5G SA core. Note, the recommended operating system installation is Ubuntu Mate (ubuntu-20.04.3-desktop-amd64.iso).

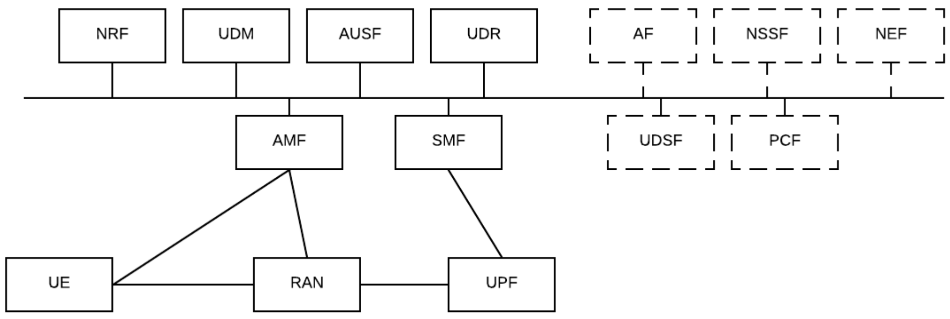


Figure 22 -OAI 5G Core Components including AMF, SMF, NRF, UPF, AUSF, UDM, and UDR currently deployed on the OpenIreland testbed

```
######################
#   Ubuntu 20.04.3   #
```

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
######################

# https://releases.ubuntu.com/20.04.2/ubuntu-20.04.3-desktop-amd64.iso

sudo apt update

sudo apt dist-upgrade

sudo apt autoremove

##############################

#   Install pre-requisites   #

##############################

# https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_PRE_REQUESITES.md



sudo apt install -y git

sudo apt install net-tools

sudo apt install -y putty

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  $(lsb_release -cs)  stable"

sudo apt update

sudo apt install -y docker

sudo apt install -y docker-ce



# add your username to the docker group, otherwise you will have to run in sudo mode.

sudo usermod -a -G docker $(whoami)

reboot



# https://docs.docker.com/compose/install/

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose



# Check pre-requistes

dpkg --list | grep docker

python3 --version



docker network create --driver=bridge --subnet=192.168.70.128/26 -o "com.docker.network.bridge.name"="demo-oai" demo-oai-public-net

sudo service docker restart
```

Figure 23 - OAI5G SA Core and gNB pre-requisites.

There are currently two different ways to compile and run the OAI5G SA core. These include:

1. Pull prebuilt Docker images, which takes several minutes.
2. Build the images locally, which can take up to an hour.

Steps to compile and run the OAI 5G SA core are outlined in Figure 24.

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
####################################################
#   Pull or Build the Docker Images            #
####################################################
#   AMF, SMF, NRF, SPGW-U-TINY, AUSF, UDM, UDR   #
####################################################
# https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/BUILD_IMAGES.md



# Git oai-cn5g-fed repository
git clone https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git
cd oai-cn5g-fed
git fetch --prune
git checkout master
git rebase origin/master
cd
cd oai-cn5g-fed
./scripts/syncComponents.sh --nrf-branch develop --amf-branch develop --smf-branch develop --spgwu-tiny-branch develop --ausf-branch develop --udm-branch develop --udr-branch develop --upf-vpp-branch develop --nssf-branch develop



####################################
#   Option 1/2: Pull the images   #
####################################
docker pull rdefosseoai/oai-amf:develop
docker pull rdefosseoai/oai-nrf:develop
docker pull rdefosseoai/oai-smf:develop
docker pull rdefosseoai/oai-udr:develop
docker pull rdefosseoai/oai-udm:develop
docker pull rdefosseoai/oai-ausf:develop
docker pull rdefosseoai/oai-upf-vpp:develop
docker pull rdefosseoai/oai-spgwu-tiny:develop
docker pull rdefosseoai/oai-nssf:develop
docker image tag rdefosseoai/oai-amf:develop oai-amf:develop
docker image tag rdefosseoai/oai-nrf:develop oai-nrf:develop
docker image tag rdefosseoai/oai-smf:develop oai-smf:develop
docker image tag rdefosseoai/oai-udr:develop oai-udr:develop
docker image tag rdefosseoai/oai-udm:develop oai-udm:develop
docker image tag rdefosseoai/oai-ausf:develop oai-ausf:develop
docker image tag rdefosseoai/oai-upf-vpp:develop oai-upf-vpp:develop
docker image tag rdefosseoai/oai-spgwu-tiny:develop oai-spgwu-tiny:develop
docker image tag rdefosseoai/oai-nssf:develop oai-nssf:develop
#####################################
#   Option 2/2: Build the images   #
#####################################
# AMF
docker build --target oai-amf --tag oai-amf:develop --file component/oai-amf/docker/Dockerfile.amf.ubuntu18 component/oai-amf
# SMF
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
docker build --target oai-smf --tag oai-smf:develop --file component/oai-smf/docker/Dockerfile.smf.ubuntu18 component/oai-smf

# NRF

docker build --target oai-nrf --tag oai-nrf:develop --file component/oai-nrf/docker/Dockerfile.nrf.ubuntu18 component/oai-nrf

# UDM

docker build --target oai-udm --tag oai-udm:develop --file component/oai-udm/docker/Dockerfile.udm.ubuntu18 component/oai-udm

# UDR

docker build --target oai-udr --tag oai-udr:develop --file component/oai-udr/docker/Dockerfile.udr.ubuntu18 component/oai-udr

# AUSF

docker build --target oai-ausf --tag oai-ausf:develop --file component/oai-ausf/docker/Dockerfile.ausf.ubuntu18 component/oai-ausf

# UPF-VPP

docker build --target oai-upf-vpp --tag oai-upf-vpp:develop --file component/oai-upf-vpp/docker/Dockerfile.upf-vpp.ubuntu18 component/oai-upf-vpp

# UPF

docker build --target oai-spgwu-tiny --tag oai-spgwu-tiny:develop --file component/oai-upf-equivalent/docker/Dockerfile.ubuntu18.04 component/oai-upf-equivalent

# NSSF

docker build --target oai-nssf --tag oai-nssf:develop --file component/oai-nssf/docker/Dockerfile.nssf.ubuntu18 component/oai-nssf

docker image prune --force

docker image ls

####################################

###### Start OAI 5G Core

####################################

cd

cd oai-cn5g-fed/docker-compose

python3 core-network.py --type start-basic --fqdn yes --scenario 1

####################################

###### Stop OAI 5G Core

####################################

cd

cd oai-cn5g-fed/docker-compose

python3 core-network.py --type stop-basic --fqdn yes --scenario 1
```

Figure 24 - OAI 5G SA Core installation

Steps to compile and run the gNB are outlined in Figure 25. To enable the OAI5G gNB to function properly, the operating system has a hard dependency on the Ettus Research UHD driver version v4.0.0.0 to support I/Q sampling. Additionally all Ubuntu dependencies are installed in this step. Note, the –*ninja* parameter during complication significantly reduces build time on recompiled builds (1-2minutes). This is very important for testing and adding code changes to the OAI5G environment, as building without the –*ninja* parameter can take 20-30 minutes.

```
################################################

#   Build OAI gNB with USRP and Ubuntu 20.04   #

################################################

# Build UHD from source

# https://files.ettus.com/manual/page_build_guide.html
```

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
cd

sudo apt-get install libboost-all-dev libusb-1.0-0-dev doxygen python3-docutils python3-mako python3-numpy python3-requests python3-ruamel.yaml python3-setuptools cmake build-essential



git clone https://github.com/EttusResearch/uhd.git

cd uhd

git checkout v4.0.0.0

cd host

mkdir build

cd build

cmake ../

make -j 4

make test # This step is optional

sudo make install

sudo ldconfig

sudo uhd_images_downloader



cd

git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git

cd openairinterface5g/

git checkout develop



# Install dependencies in Ubuntu 20.04

cd

cd openairinterface5g/

source oaienv

cd cmake_targets/

./install_external_packages.ubuntu20



# Build OAI 5G

cd

cd openairinterface5g/

source oaienv

cd cmake_targets/

./build_oai -w USRP --nrUE --gNB --build-lib all -c --ninja



# Known issue: if the build fails, just start again from # Install dependencies in Ubuntu 20.04
```

Figure 25 - Build OAI gNB with USRP and Ubuntu 20.04

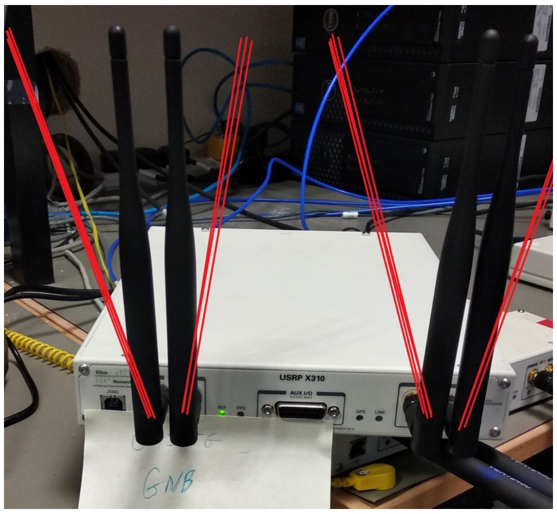
The instructions in Figure 26 are used to run the gNB. The system can be tuned for performance using parameters such as net.core.wmem_max and setting the cpufreq. These settings support more efficient parallel job performance necessary to get the best performance from the USRP devices.

```
for ((i=0;i<$(nproc);i++)); do sudo cpufreq-set -c $i -r -g performance; done #  Its possible to set the CPU governor to performance per core by issuing this command
```

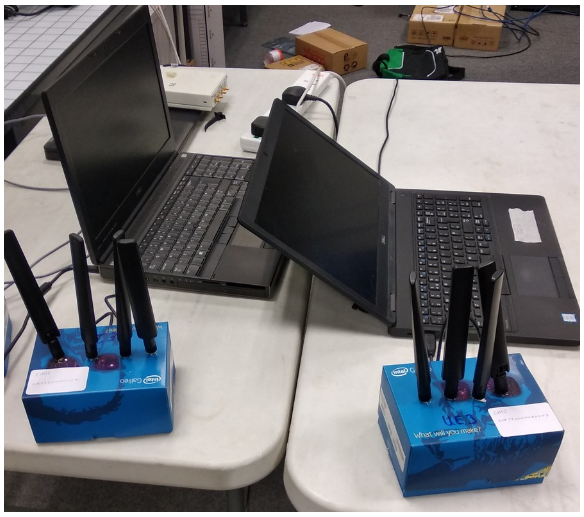
Co-funded by the Horizon 2020
Framework Programme of the European Union

```
sudo sysctl -w net.core.wmem_max=62500000 # Defines the default send window size

sudo sysctl -w net.core.rmem_max=62500000 # Defines the maximum receive window size

sudo sysctl -w net.core.wmem_default=62500000 # Defines the default send window size

sudo sysctl -w net.core.rmem_default=62500000 #  Defines the maximum receive window size

sudo ethtool -G ens8f3 tx 4096 rx 4096 #  Increasing the Ring Buffers on the NIC may help prevent flow control errors at higher rates



cd

cd openairinterface5g/

source oaienv

cd cmake_targets/ran_build/build

sudo ./nr-softmodem -O ~/gnb.sa.band78.fr1.106PRB.usrpb210.conf --sa --usrp-tx-thread-config 1 --thread-pool 0,2,4,6
```

Figure 26 - Steps to execute the gNB



(a)



(b)

Figure 27 - gNB(a) and Quectel RM500Q-GL 5G Modems (b) connected to the Dell Precision laptops

Quectel 5G UE

Unfortunately at present not many UEs are capable of attaching to a 5G gNB in Standalone mode. To simplify connectivity issues, we purchased three Quectel RM500Q-GL 5G Modems, two of which are shown above in Figure 27(b). 5G specification of the UE is available in Figure 28. The Quectel 5G UEs, which are effectively 5G dongle sticks, were plugged into a Dell Precision-M4700 and Precision-3520 laptops supporting the cards. The laptops allowed us to run ping and iperf tests over the wireless links between the gNB and UEs, once connectivity between the gNB and UE were established. Note, the available bandwidth is affected by the distance of the UE from the gNB. Currently the UEs are about 2 metres from the gNB. While not easy to maintain this exact distance in a busy lab environment, we tried to minimise the movement of the UEs throughout experimentation.

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
###############################################################################################################

Quectel RM500Q-GL 5G Modem M.2 module Quectel RM500Q+Testing USB adapter+5G Antenna Pigtail supply document

###############################################################################################################

5G RM500Q-GL

5G sub-6GHz module

52.0mm × 30.0mm × 2.3mm

8.4g

M.2 form factor

Max. downlink 2.5Gbps / 900Mbps uplink

Extended temperature range of -40°C to +85°C

5G:

https://www.quectel.com/wp-content/uploads/2021/03/Quectel_RM500Q-GL_5G_Specification_V1.4.pdf

5G NR 3GPP Release 15 NSA/SA operation, Sub-6 GHz

5G NR NSA n38/n41/n77/n78/n79

5G NR SA n1/n2/n3/n5/n7/n8/n12/n20/n25/n28/n38/n40/n41/n48*/n66/n71/n77/n78/n79

MIMO DL: 4 × 4 MIMO on n1/n2/n3/n7/n25/n38/n40/n41/n48*/n66/n77/n78/n79

UL: 2 × 2 MIMO on n41/n77/n78/n79
```

Figure 28 - Quectel RM500Q-GL 5G Modem specification

4.2.6 Overview of OAI 5G Challenges:

We encountered several challenges related to the delivery of this project with regards to the stability of the OAI 5G SA gNB, and OAI 5G SA core. Some of the challenging and time consuming problems experienced during this project are outlined briefly here.

Connectivity Issue: gNB crashes when running iperf tests to UE

When attaching the UE, we noticed the gNB would crash. Ping worked fine between the core and the UE, but running large volumes of traffic across the radio link using iperf caused the gNB to crash. After investigation, this issue appeared to be related to the MTU value on the UE. To rectify the problem, we changed the MTU parameter from 1500 bytes in the OAI-SMF container to 1358 bytes. Changes are shown in Figure 29. Note, this does not seem to happen on Windows machines supporting the UE, which defaults to 1500bytes. This issue seems to be specific to Ubuntu/Linux environments.

```
##Change the following 1358 parameter to 1500 bytes in the core SMF component and recompile

oai-cn5g-fed/component/oai-smf/src/smf_app/smf_config.hpp:    ue_mtu         = 1358;

oai-cn5g-fed/component/oai-smf/etc/smf.conf:    UE_MTU = 1358;

To

oai-cn5g-fed/component/oai-smf/src/smf_app/smf_config.hpp:    ue_mtu         = 1500;

oai-cn5g-fed/component/oai-smf/etc/smf.conf:    UE_MTU = 1500;

Then run the following to recompile the core:



# Recompile the SMF
```

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
cd

cd oai-cn5g-fed

docker build --target oai-smf --tag oai-smf:develop --file component/oai-smf/docker/Dockerfile
```

Figure 29 - Change OAI-SMF docker container MTU parameter from 1500 bytes to 1358 bytes

Connectivity Issue: LLLLs – UHD Drivers on servers have problem receiving and processing I/Q samples from X310 USRP.

We experienced several delays in the project due to an LLLLs issue with the X310 USRPs. The UE would not attach successfully to the gNB, and when it did, the bandwidth between the UE and 5G SA core was a very unreliable – typically measuring a Kbps bandwidth link. The problem was occurring 9 times out of every 10 attempts to run the gNB. We performed several remediations to solve the problem. The first, after taking advice from National Instruments, the USRP vendors, and an OpenAirInterface consultant company called ALLBESMART, was to utilise an Intel X710 10GbE Network SPF+ adapter [2]. We experienced an exponential improvement in the reliability of the link between server and USRP with minimal LLLs using this card. We achieved even further improvements when we utilised the recommended version of Ubuntu 20.04.3 MATE advised by the ALLBESMART consultants https://releases.ubuntu.com/20.04.2/ubuntu-20.04.3-desktop-amd64.iso. Prior to this success, we had tried using several other networking cards 10G/25G/etc, and many different Linux operating systems (Ubuntu Server (Generic), Ubuntu Server (low-latency), etc), different antenna's and USRPs, with very little if any improvement in overall performance. These improvements resulted in the UE receiving up to 90Mbps bandwidth on the 5G air link.

4.2.7 Overview of 5G NR Carrier Bandwidth Part

The Carrier Bandwidth Part is a new concept in 5G NR to support partition within a 5G new radio frequency band. Carrier bandwidth parts offer flexibility so that multiple different types of signal such as eMBB and mMCT can be sent and supported in a given carrier bandwidth. BWPs flexibility also offer power saving capabilities for the UE and gNB. According to 138.211 4.4.5 [3], the carrier bandwidth is the operating bandwidth, while the bandwidth parts (BWP) are a subset of the carrier bandwidth. A UE can be configured with a maximum of 4 bandwidth parts in the uplink(UL) and downlink(DL). However, only one uplink and one downlink bandwidth part can be active at any given time. Figure 30 shows the OAI 5G BWPs Dedicated Serving Cell Configuration for Downlink and Uplink currently defined in the X310 USRP gNB configuration file, and used during this project.

```
# Dedicated Serving Cell Configuration

servingCellConfigDedicated = ({

  # BWP-Downlink

    # BWP 1 Configuration

      dl_bwp-Id_1 = 1;

      dl_bwp1_locationAndBandwidth = 28875; // RBstart=0, L=106 (40 MHz BW)

      # subcarrierSpacing

      # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120

      dl_bwp1_subcarrierSpacing = 1;
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
 # BWP 2 Configuration
  dl_bwp-Id_2 = 2;
  dl_bwp2_locationAndBandwidth = 13750; // RBstart=0, L=51 (20 MHz BW)
  # subcarrierSpacing
  # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
  dl_bwp2_subcarrierSpacing = 1;

 # BWP 3 Configuration
  dl_bwp-Id_3 = 3;
  dl_bwp3_locationAndBandwidth = 6325; // RBstart=0, L=24 (10 MHz BW)
  # subcarrierSpacing
  # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
  dl_bwp3_subcarrierSpacing = 1;

 firstActiveDownlinkBWP-Id = 3;  #BWP-Id
 defaultDownlinkBWP-Id     = 3;  #BWP-Id

 # bwp-InactivityTimer              ENUMERATED {ms2, ms3, ms4, ms5, ms6, ms8, ms10, ms20, ms30,
 #                                  ms40,ms50, ms60, ms80,ms100, ms200,ms300, ms500,
 #                                  ms750, ms1280, ms1920, ms2560, spare10, spare9, spare8,
 #                                  spare7, spare6, spare5, spare4, spare3, spare2, spare1 }

# UplinkConfig
 # BWP-Uplink
  # BWP 1 Configuration
   ul_bwp-Id_1 = 1;
   ul_bwp1_locationAndBandwidth = 28875; // RBstart=0, L=106 (40 MHz BW)
   # subcarrierSpacing
   # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
   ul_bwp1_subcarrierSpacing = 1;

  # BWP 2 Configuration
   ul_bwp-Id_2 = 2;
   ul_bwp2_locationAndBandwidth = 13750; // RBstart=0, L=51 (20 MHz BW)
   # subcarrierSpacing
   # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120
   ul_bwp2_subcarrierSpacing = 1;

  # BWP 3 Configuration
   ul_bwp-Id_3 = 3;
   ul_bwp3_locationAndBandwidth = 6325; // RBstart=0, L=24 (10 MHz BW)
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
        # subcarrierSpacing

        # 0=kHz15, 1=kHz30, 2=kHz60, 3=kHz120

        ul_bwp3_subcarrierSpacing = 1;



      firstActiveUplinkBWP-Id = 3;  #BWP-Id

  }

);
```

Figure 30 - OAI 5G BWPs Dedicated Serving Cell Configuration for Downlink and Uplink USRP X310 Configuration

The OAI 5G gNB developers have implemented and released a carrier bandwidth parts branch called *Dedicated-BWPs-dynamic-switching*. Figure 31 shows the instructions to compile and build this branch. As part of this NGI Atlantic deliverable, we modified this branch to support dynamic bandwidth changing which is supported by ZeroMQ library. ZeroMQ is an asynchronous messaging library used in distributed or concurrent applications, which can run without a dedicated message broker. Code changes applied to the *Dedicated-BWPs-dynamic-switching* branch to support ZMQ are show in Appendix 1. The ZMQ client, which supports real-time configuration of the OAI5G gNB BWPs, is show in Appendix 2.

```
cd

git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git

cd openairinterface5g/

git checkout Dedicated-BWPs-dynamic-switching

#https://gitlab.eurecom.fr/oai/openairinterface5g/-/commits/Dedicated-BWPs-dynamic-switching

#to compile – only gNB

cd

cd openairinterface5g/

source oaienv

cd cmake_targets/

./build_oai -w USRP --gNB --ninja
```

Figure 31 - Download and compile the OAI 5G Dedicated-BWPs-dynamic-switching

4.2.8 Initial Results: BWPs changes between OAI5G SA Core and UE

We ran iperf server on the UE, and iperf client from the OAI 5G SA core to capture bandwidth results over the 5G link using BPWs 1,2, and 3. The iperf server on the UE needs to be started first. The command to start the iperf server on the Quectel UE, and client on the OAI5G core and start the iperf data transfer is shown in Figure 32.

```
##Start iperf Server

##Quectel UE node

iristestbed@iristestbed-Precision-3520:~$ iperf -s -u -i 1 -p 5002 -B 12.1.1.53

------------------------------------------------------------

Server listening on UDP port 5002
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
UDP buffer size:  208 KByte (default)

------------------------------------------------------------



###Start iperf Client

###OAI5G SA Core

docker exec -it oai-ext-dn iperf -u -t 86400 -i 1 -fk -B 192.168.70.135 -p 5002 -b 90M -c 12.1.1.2
```

Figure 32 - iperf client command executed against the oai-ext-dn docker container

Figure 33 shows some sample iperf results when changing the BWPs parameter between BWPs 3 using 10MHz bandwidth. We received roughly 16 Mbits/sec of data over the downlink connection. BWPs 2 configured with 20MHz of bandwidth averaged 26.8 Mbits/sec of data on the downlink connection. Finally, BWPs 1 supporting 40MHz bandwidth achieved 51.8 Mbits/sec of data. Note, the client attempts to transfer 90 megabytes of data between the client (shown in Figure 32 above) and server, and this is throttled by the gNB based on the available radio link bandwidth.

```
iristestbed@iristestbed-Precision-3520:~$ iperf -s -u -i 1 -p 5002 -B 12.1.1.53

------------------------------------------------------------

Server listening on UDP port 5002

UDP buffer size:  208 KByte (default)

[  1] local 12.1.1.53 port 5002 connected with 192.168.70.135 port 50552

[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams

<BWP 1 Configuration: 40 MHz Bandwidth >

[  1] 71.0000-72.0000 sec  6.17 MBytes  51.8 Mbits/sec   0.639 ms 12411/16815 (74%)

[  1] 72.0000-73.0000 sec  6.77 MBytes  56.8 Mbits/sec   0.276 ms 4200/9027 (47%)

[  1] 93.0000-94.0000 sec  6.76 MBytes  56.7 Mbits/sec   0.197 ms 2968/7793 (38%)

[  1] 94.0000-95.0000 sec  6.62 MBytes  55.5 Mbits/sec   0.303 ms 2995/7714 (39%)

...

<BWP 2 Configuration: 20 MHz Bandwidth >

[  1] 99.0000-100.0000 sec  3.19 MBytes  26.8 Mbits/sec   0.706 ms 5823/8099 (72%)

[  1] 100.0000-101.0000 sec  3.27 MBytes  27.4 Mbits/sec   1.047 ms 5729/8059 (71%)

[  1] 101.0000-102.0000 sec  3.19 MBytes  26.7 Mbits/sec   0.323 ms 5884/8158 (72%)

[  1] 102.0000-103.0000 sec  3.08 MBytes  25.8 Mbits/sec   0.602 ms 5622/7817 (72%)

[  1] 103.0000-104.0000 sec  3.27 MBytes  27.5 Mbits/sec   0.430 ms 5726/8062 (71%)

[  1] 104.0000-105.0000 sec  3.38 MBytes  28.3 Mbits/sec   0.336 ms 6145/8554 (72%)

..

<BWP 3 Configuration: 10 MHz Bandwidth >

[  1] 127.0000-128.0000 sec  1.91 MBytes  16.0 Mbits/sec   0.968 ms 2343/3703 (63%)

[  1] 128.0000-129.0000 sec  1.74 MBytes  14.6 Mbits/sec   2.288 ms 2934/4177 (70%)

[  1] 129.0000-130.0000 sec  1.53 MBytes  12.9 Mbits/sec   0.870 ms 2339/3432 (68%)

[  1] 130.0000-131.0000 sec  1.69 MBytes  14.2 Mbits/sec   1.038 ms 3401/4607 (74%)

[  1] 131.0000-132.0000 sec  1.71 MBytes  14.4 Mbits/sec   0.796 ms 6012/7234 (83%)

....
```

Figure 33 - Sample Results from Quectel UE iperf tests

Co-funded by the Horizon 2020
Framework Programme of the European Union

Figure 34 illustrates the Dynamic BWP change occurring in real-time on the OAI 5G UE interface. Note, the reliability of the radio link can vary due to various issues including time of day, UE distance from gNB, gNB server load, etc.

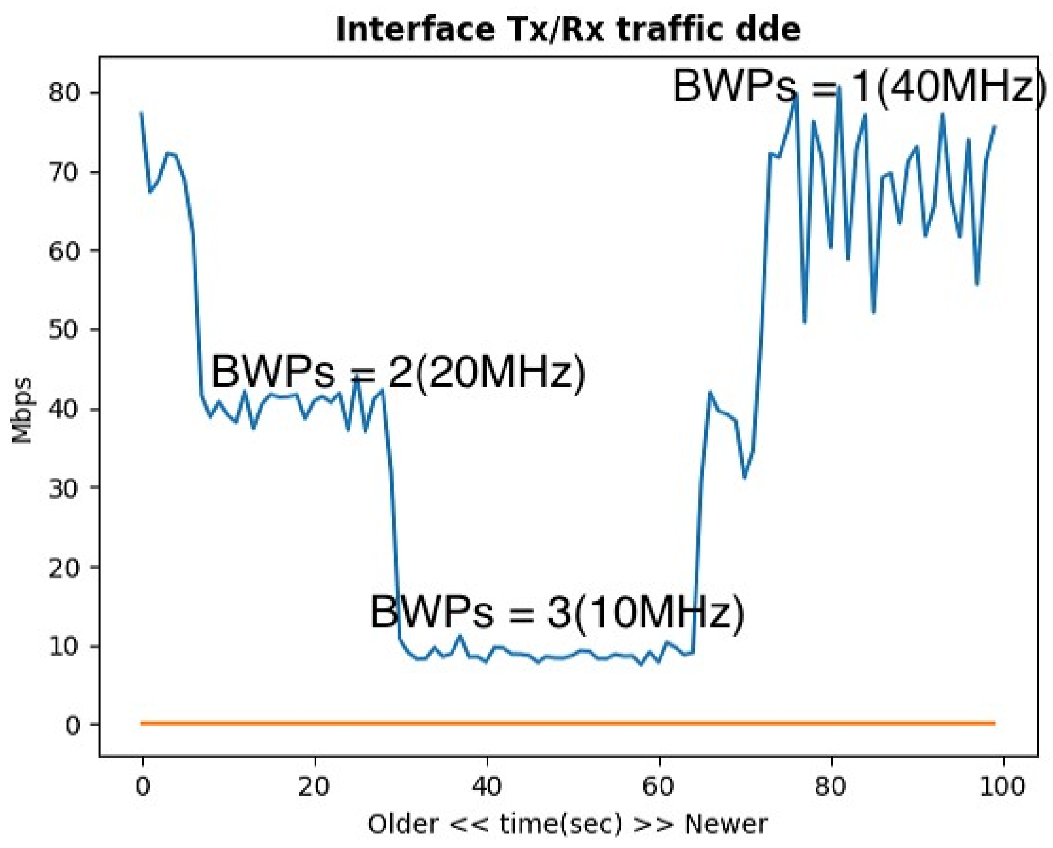


Figure 34 -Shows the Dynamic BWP change occurring in real-time on the OAI 5G UE interface.

4.2.9 Wireless-optical testbed convergence

In order to implement variable bandwidth in 5G, we utilised the 5G/NR - Carrier Bandwidth Part outlined above, implemented by the OAI 5G community. As discussed, the OpenIreland-COSMOS testbed interconnectivity utilises VXLANs for control and data. In this experiment TCD represents the edge computing node, while COSMOS acts as the central controlling cloud node, and traffic source. TCD Provides 5G based C-RAN using bare metal servers and X310 USRPs with full stack 5G implementation supported by OpenAirInterface 5G. The BWPs is orchestrated by a custom QoS controller running in the COSMOS testbed. Note, we also utilise the OAI 5G SA Core, which requires some configuration to support pushing data from COSMOS testbed to the Quectel UE. Required container modifications are show in Figure 35 and Figure 36, which need to be run before experimentation.

```
docker exec -it oai-spgwu /bin/bash

2apt-get update
```

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
3apt install iputils-ping tcpdump iptables net-tools vim -y

4iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE

5sysctl net.ipv4.conf.all.forwarding=1

6iptables -P FORWARD ACCEPT
```

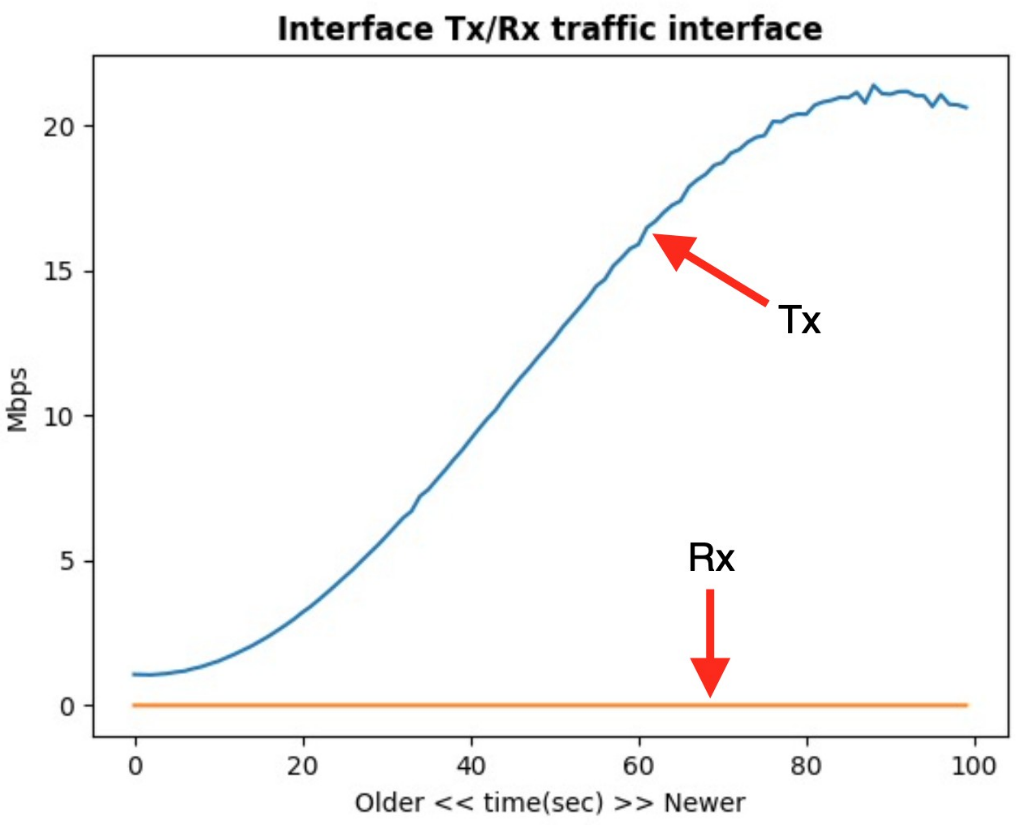
Figure 35 - Enable packet forwarding on the OAI 5G SA Core container (oai-spgwu)

```
sudo route add -net 10.130.0.0/24 wwan0
```

Figure 36 - Forward traffic destined to the COSMOS testbed out OAI UE wwan0 interface

4.2.10 Experiment Results

Figure 37 shows the output of a *traffic generator* application running on the COSMOS testbed. The *blue* line represents Tx traffic created by the traffic generator showing varying traffic, while the *orange* line represents Rx traffic received. An intelligent QoS controller function monitors the output from the traffic generator script, and dynamically reserves capacity in the optical transmission networks, and spectrum in the wireless OAI 5G edge nodes running in the OpenIreland testbed based on the diverging traffic patterns. A subset of the Trinity College Dublin proprietary intelligent controller is show in Figure 38. In our case the bandwidth parts on the UE are dynamically changed when the QoS program sends commands to the gNB ZMQ agent. The controller outputs the date/time, QoS setting, the averaged Tx bandwidth observed, observed bandwidth trigger parameter, and the average Rx traffic. The intelligent controller output is shown in Figure 39 as it sends BWPs instructions based on diverging traffic patterns observed to the OpenAirInterface gNB in the OpenIreland testbed.

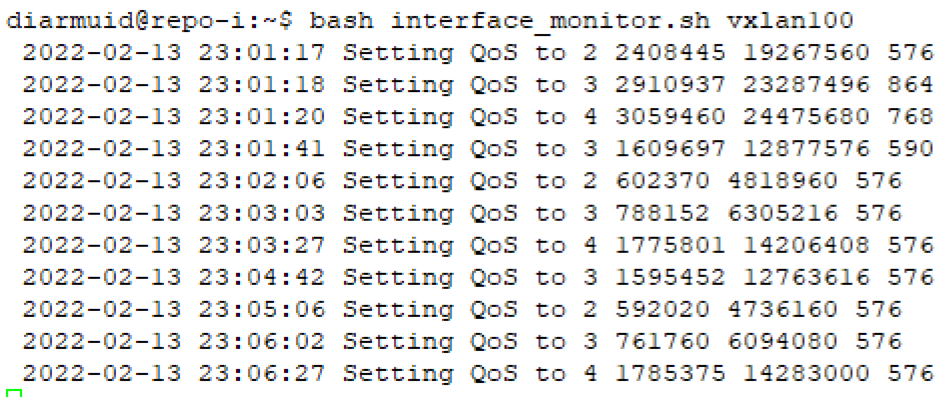


Co-funded by the Horizon 2020 Framework Programme of the European Union

Figure 37 - Dynamic BWP change occurring in real-time on the OAI 5G UE interface

```
if [[ ( "$QOS_LEVEL" == 1 ) && ( "$btx_avg" -gt 17000000 ) ]] ; then

    QOS_LEVEL=2

    echo "" `date +'%Y-%m-%d %H:%M:%S'` Setting QoS to $QOS_LEVEL $tx_avg $btx_avg $rx_avg

    # Run ABNO bash script

    python3 oi_zmq_client.py ${UE} ${QOS_LEVEL} > /dev/null

elif [[ ( "$QOS_LEVEL" == 2 ) && ( "$btx_avg" -gt 35000000 ) ]] ; then

    QOS_LEVEL=3

    echo "" `date +'%Y-%m-%d %H:%M:%S'` Setting QoS to $QOS_LEVEL $tx_avg $btx_avg $rx_avg

    # Run ABNO bash script for QoS 3

    python3  oi_zmq_client.py ${UE} ${QOS_LEVEL} > /dev/null

 elif [[ ( "$QOS_LEVEL" == 3 ) && ( "$btx_avg" -gt 70000000 ) ]] ; then

    QOS_LEVEL=4

    echo "" `date +'%Y-%m-%d %H:%M:%S'` Setting QoS to $QOS_LEVEL $tx_avg $btx_avg $rx_avg

    # Run ABNO bash script for QoS 4

    python3 oi_zmq_client.py ${UE} ${QOS_LEVEL} > /dev/null
```

Figure 38 - The BWPs monitor and controller running on the COSMOS testbed

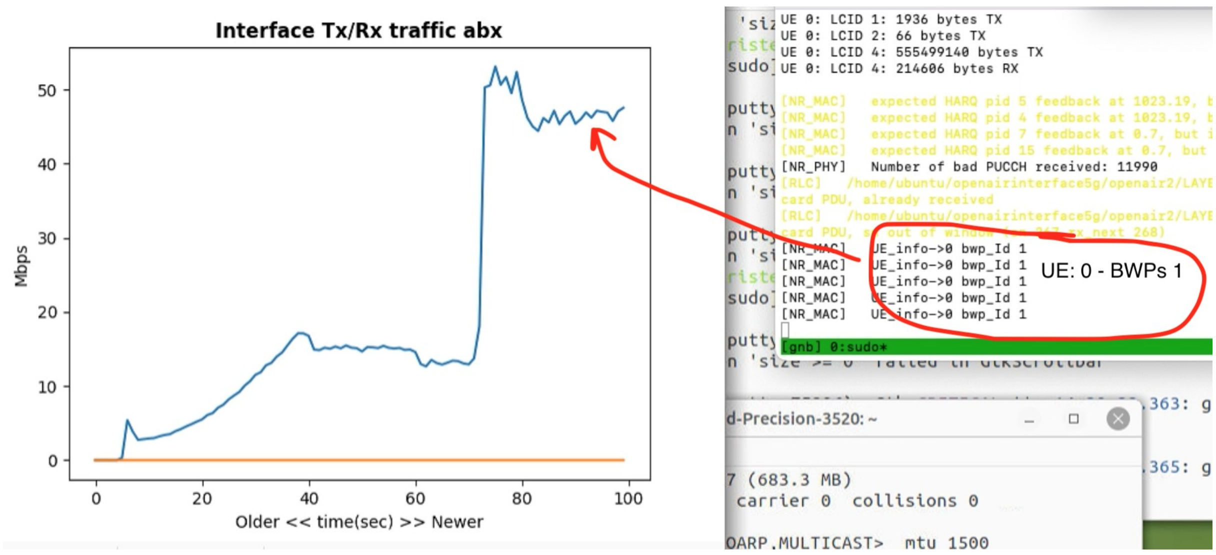


```
diarmuid@repo-i:~$ bash interface_monitor.sh vxlan100
 2022-02-13 23:01:17 Setting QoS to 2 2408445 19267560 576
 2022-02-13 23:01:18 Setting QoS to 3 2910937 23287496 864
 2022-02-13 23:01:20 Setting QoS to 4 3059460 24475680 768
 2022-02-13 23:01:41 Setting QoS to 3 1609697 12877576 590
 2022-02-13 23:02:06 Setting QoS to 2 602370 4818960 576
 2022-02-13 23:03:03 Setting QoS to 3 788152 6305216 576
 2022-02-13 23:03:27 Setting QoS to 4 1775801 14206408 576
 2022-02-13 23:04:42 Setting QoS to 3 1595452 12763616 576
 2022-02-13 23:05:06 Setting QoS to 2 592020 4736160 576
 2022-02-13 23:06:02 Setting QoS to 3 761760 6094080 576
 2022-02-13 23:06:27 Setting QoS to 4 1785375 14283000 576
```

Figure 39 - Illustration showing the QoS in the UE to BWPs 1, 2, 3.

Figure 40, Figure 41, and Figure 42 show the BWPs change experienced by the Quectel UE. We have deployed a simple traffic monitoring tool that monitors the traffic on the OAI 5G network UE interface. Figure 40 shows the UE 0 changing to BWPs 1, which represents QoS = 3 in the intelligent QoS controller application. Data speeds averaged 45Mbps.

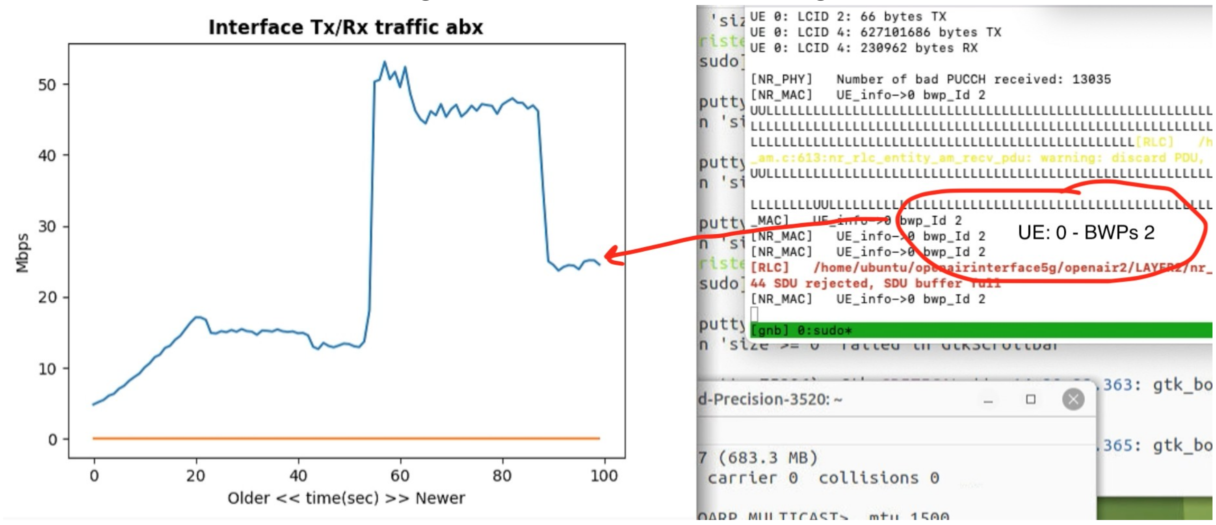
Co-funded by the Horizon 2020
Framework Programme of the European Union



```
UE 0: LCID 1: 1936 bytes TX
UE 0: LCID 2: 66 bytes TX
UE 0: LCID 4: 555499140 bytes TX
UE 0: LCID 4: 214606 bytes RX

[NR_MAC]   expected HARQ pid 5 feedback at 1023.19, b
[NR_MAC]   expected HARQ pid 4 feedback at 1023.19, b
[NR_MAC]   expected HARQ pid 7 feedback at 0.7, but i
[NR_MAC]   expected HARQ pid 15 feedback at 0.7, but
[NR_PHY]   Number of bad PUCCH received: 11990
[RLC]   /home/ubuntu/openairinterface5g/openair2/LAYE
card PDU, already received
[RLC]   /home/ubuntu/openairinterface5g/openair2/LAYE
card PDU, sn out of window (sn 267 rx_next 268)
[NR_MAC]   UE_info->0 bwp_Id 1
[NR_MAC]   UE_info->0 bwp_Id 1
[NR_MAC]   UE_info->0 bwp_Id 1
[NR_MAC]   UE_info->0 bwp_Id 1
[NR_MAC]   UE_info->0 bwp_Id 1

[gnb] 0:sudo*
```

Figure 40 - UE 0 - BWPs 1 (QoS = 3) with averaging 45Mbps

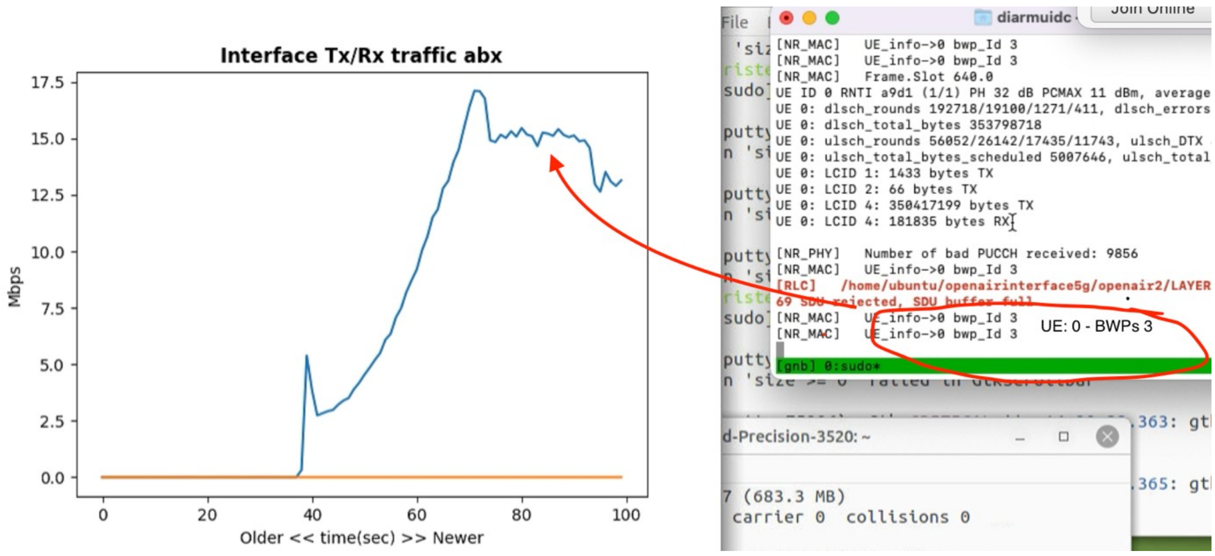
Figure 41 shows UE 0 changing to BWPs 2, which is QoS = 2 in the intelligent QoS controller, averaging about 27 Mbps. Finally Figure 42 shows UE 0 changing to BWPs 3, which is QoS = 3 in the intelligent controller, averaging about 15Mbps. We have uploaded a video to YouTube [4] showing the traffic to the UE, and activity on the gNB as the QoS controller, which is running on the COSMOS testbed, makes network wide changes based on the UDP traffic being sent across the network.



```
UE 0: LCID 2: 66 bytes TX
UE 0: LCID 4: 627101686 bytes TX
UE 0: LCID 4: 230962 bytes RX

[NR_PHY]   Number of bad PUCCH received: 13035
[NR_MAC]   UE_info->0 bwp_Id 2
UULLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL[RLC]   /h
_am.c:613:nr_rlc_entity_am_recv_pdu: warning: discard PDU,
UULLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

LLLLLLLLUULLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
_MAC]   UE_info->0 bwp_Id 2
[NR_MAC]   UE_info->0 bwp_Id 2
[NR_MAC]   UE_info->0 bwp_Id 2
[RLC]   /home/ubuntu/openairinterface5g/openair2/LAYER2/nr_
44 SDU rejected, SDU buffer full
[NR_MAC]   UE_info->0 bwp_Id 2

[gnb] 0:sudo*
```

Figure 41 - UE 0 - BWPs 2 (QoS = 2) averaging 27 Mbps

Co-funded by the Horizon 2020
Framework Programme of the European Union



```
[NR_MAC]   UE_info->0 bwp_Id 3
[NR_MAC]   UE_info->0 bwp_Id 3
[NR_MAC]   Frame.Slot 640.0
UE ID 0 RNTI a9d1 (1/1) PH 32 dB PCMAX 11 dBm, average
UE 0: dlsch_rounds 192718/19100/1271/411, dlsch_errors
UE 0: dlsch_total_bytes 353798718
UE 0: ulsch_rounds 56052/26142/17435/11743, ulsch_DTX
UE 0: ulsch_total_bytes_scheduled 5007646, ulsch_total
UE 0: LCID 1: 1433 bytes TX
UE 0: LCID 2: 66 bytes TX
UE 0: LCID 4: 350417199 bytes TX
UE 0: LCID 4: 181835 bytes RX

[NR_PHY]   Number of bad PUCCH received: 9856
[NR_MAC]   UE_info->0 bwp_Id 3
[RLC]   /home/ubuntu/openairinterface5g/openair2/LAYER
69 SDU rejected, SDU buffer full
[NR_MAC]   UE_info->0 bwp_Id 3
[NR_MAC]   UE_info->0 bwp_Id 3
[gnb] 0:sudo*
```

```
7 (683.3 MB)
 carrier 0  collisions 0
```

Figure 42 - UE 0 – BWPs 2 (QoS = 3) averaging 15Mbps

5 Discussion and Analysis of Results

5.1 Experiments

Here we comment on how we fared on the Experiments we proposed in Deliverable 1. For Experiment 1, we proposed Quality of Transmission (QoT) estimation of optical systems where we would train machine learning models and carry out transfer learning experiments across the OpenIreland and COSMOS testbeds. With respect to the key headings of what we set out to do in Deliverable 1, we were able to achieve the following results:

- Generation of optical signals by using the constant power generated by the preamp of the ROADMs
- Estimation of the OSNR by filtering the received signal with the filters embedded inside the ROADMs (MUX and DeMUX)
- Development of a dashboard for configuration, automation and data collection

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Development of the ML framework called ONIS (Optical Network Intelligent System) where we can launch state-of-the-art Reinforcement-Learning agents for network optimization

If we look at the performance in terms of time computation, we can: deploy a new wavelength + estimate the OSNR for all the wavelengths already deployed in the network within 2.5/3 seconds; and the maximum power level of the generated signal that we can obtain is about -15/-13 dBm.

We demonstrated that there was effective collaboration between TCD and Polimi in the training of machine learning models to assign WDM channels in the transmission spectrum of a fibre. This was achieved through QoT estimation regime based on the collection of many test data points.

In Deliverable 2, we reported on a number of activities that were executed originally on COSMOS but then transposed to OpenIreland. Firstly, the Mininet Optical experiment demonstrated at OFC was run on OpenIreland [5]. Secondly, we implemented the COSMOS ROADM ring on the OpenIreland testbed. We had intended to do a more in-depth study of the use of Transfer learning executed on OpenIreland and then on COSMOS, but this was limited by the delays in the provisioning of the circuit between the OpenIreland and COSMOS. On the other hand, we were able to delivery other innovations in the 5G domain, implementing a stable 5G standalone stack, and a polished graphical user interface to the Machine Learning experiments on OpenIreland.

In Experiment 2 we proposed to investigate the convergence of optical and wireless networks by carrying out a variable bandwidth experiment (originally developed in [1] for a 4G system), between the COSMOS and TCD testbeds, which are linked using a dedicated 10G link. The aim of this experiment, as mentioned in Deliverable 1, was to achieve dynamic capacity allocation by reserving optical fibre and wireless spectrum resources, through a hybrid optical and wireless control plane, to satisfy QoS of high end applications. In our experiment we were able to deploy a new 5G standalone system, based on OAI5G at the OpenIreland testbed, and we deployed traffic generation and measurement system in the COSMOS testbed. Variable rate traffic was generated at COSMOS and transmitted over the transatlantic, inter-continental 10G link provisioned as part of the project. The provisioned bandwidth for the front-haul traffic which was received at the UEs in OpenIreland was orchestrated by the QoS controller application according to the traffic that was generated at COSMOS. In this experiment however, the dynamic allocation of optical capacity was not carried out in COSMOS, due to short term unavailability of part of the optical resources when the experiment was carried out. However, such dynamic optical allocation was demonstrated in experiment 1, through dynamic channels setup across a ROADM network.

5.2 Results

Here we comment on how we fared against the plan provided in Deliverable 1.

1. We stated we aimed to understand how well ML algorithms can approximate real device and systems behaviour, taking into consideration issues of unpredictable variability, which are typically neglected in simulation.
   In this we believe, we achieved the objectives through experiment 1. There we implemented from scratch an element management system for controlling the optical testbed. The devices

we controlled included Lumentum Roadms, Polatis optical switch, Juniper EDFAs and Cassini transponders. We used open source tools and software throughout, such as Flask, Python, Netconf, Linux. On top of this, we implemented a GUI framework for allowing third parties to create and execute experiments. Polimi used the element management system and the GUI to fabricate the ONIS machine learning system as described in the second deliverable D2. Polimi was then able to run an important experiment on the testbeds that demonstrated how channels should be allocated in a spectrum range, to optimise the performance of all the channels. This is an extremely complex exercise to run, and almost impossible to do manually. The project achieved this result through the application of machine learning with knowledge gained over the running of thousands of cycles.

2. We stated we aimed to showcase large-scale orchestration across continents, over multiple technology domains, such as wireless, optical and cloud. In this we believe, we achieved the objectives through experiment 2. We demonstrated the creation of resources such as computation, memory and storage in virtualisation frameworks at both COSMOS and TCD testbeds. We deployed a 5G standalone system, based on OAI5G at the TCD OpenIreland testbed, and we deployed a traffic generation and measurement system at COSMOS. Variable rate traffic was generated at COSMOS and transmitted over the transatlantic, inter-continental 10G link provisioned as part of the project. The provisioned bandwidth for the front-haul traffic which was received at the UEs in TCD was varied according to the traffic that was generated at COSMOS. As the traffic increased, we provisioned more bandwidth using the 5G Bandwidth Parts scheme, and vice versa, when less bandwidth was needed, less bandwidth parts were provisioned. A fundamental aspect is that there was no drop in connection when bandwidth was changed, with traffic increasing smoothly.

3. We stated we aimed to demonstrate the execution of experiments between testbeds. In this we believe, we again we achieved the objectives also through experiment 2, as described above.

6 Management of Risks and Contingencies

- ***People availability***. The dominant issue was the ability of project partners to attend labs physically to set up and conduct experiments during the Covid-19 pandemic lock down. For much of the period 2020 and 2021, either the labs were closed, or access was restricted to one person at a time. This impacted on the efficiency of how the experiments could be executed. Fortunately, Polimi was able to visit the TCD labs in September 2021 to assist in the set up and configure the equipment and test scenarios. After which, Polimi and other project partners were able to work remotely on the OpenIreland testbed with the high level of flexibility.
- ***Platform equipment and links***. As can be seen from the underlying infrastructure needed for the connectivity between the COSMOS and OpenIreland testbeds, there are many hops and connections needed to provide the link. There was an inordinate and unforeseen amount of delay in getting the circuit provisioned. The link was eventually provisioned in Jan 2022, several months late. To mitigate this issue, we proceeded with activities in anticipation that

Co-funded by the Horizon 2020 Framework Programme of the European Union

the circuit would be installed. We used an alterative path over the internet to access the COSMOS testbed. While it provided some connectivity to demonstrate functionality, it was far from performing as a dedicated circuit, using the metrics of latency, jitter and bandwidth. Using this approach, we were able to complete the bulk of the experiments in advance of the link being available with the final part of the experiments completed when the circuit was installed.

- ***Unforeseen circumstances***. The Covid-19 pandemic was a once in a generation event. While it existed at the beginning of the project, what was unknown was how long it would last and the impact it would bring. We believe that the remote access to the testbed allowed experiments to continue smoothly, with minimal on-site support required.
- ***Processes, procurement delays***. Given that the NGI Atlantic project was run during a pandemic, we believe that there were no extraordinary delays due to global freight and logistics issues.

7 Present and Foreseen TRL

In Deliverable 1, we benchmarked the maturity of the solutions using the Technology Readiness Level (TRL) framework. Overall, our research followed standard approaches of articulating both the question and goal of the research, then collecting data to resolve the problem. We went through a number of iterations to refine the problem and solution. As stated in our first deliverable, because we were going to use disaggregated componentised systems in end to end lab experiments, our starting point was going to be TRL4. The elements we believe improved with maturity

1. Framework for managing and programming components in the testbed.
2. User interface for configuring experiments, running experiments and retrieving data
3. Refining OAI5G in standalone mode for high bandwidth transmission
4. Adapting OAI5g for control of bandwidth
5. Implementing a robust inter-testbed connectivity

8 Exploitation, Dissemination and Communication Status

Here is a consolidated list of exploitation, dissemination and communications actions, with an emphasis on the exploitation plans after the end of the project.

8.1 Digital Around the World 2021 – Next Generation Internet

Plenary Title:	Digital Around the World 2021 - Next Generation Internet
Dates:	23:00, Wednesday 20 Oct 2021 (1 hour)
Presenter:	Professor Marco Ruffini (Trinity College Dublin)
Location:	Webinar
	This session will explore results of the Next Generation Internet Initiative (NGI),

Co-funded by the Horizon 2020 Framework Programme of the European Union

	an initiative spearheaded by the European Commission in order to shape the development and evolution of the Internet into an "Internet of Humans". NGI has the goal to develop an Internet that responds to people's fundamental needs, including trust, security, and inclusion, while reflecting the values and the norms all citizens enjoy in Europe. Of course, the work being carried out in NGI is global and the session today in Digitalaroundtheworld will have a particular focus on the support of the NGIatlantic.eu project to EU – US projects to experiment with NGI results with key stakeholders from Europe and the United States of America. The following projects will be presented during the session: Opportunities from NGIatlantic.eu, Responsibility to protect population through peer governance and trusted community (P2PR2P), CloudBank EU NGI, and Integrating OpenIreland and NSF COSMOS testbeds for delivering a cross-Atlantic Open Networking Solution.
Evidence:	https://sites.grenadine.co/sites/iot/en/digital-around-the-world-2021/schedule/8105/NextGenerationInternet

8.2 Towards Digital Twinning – Hosted by the Rutgers University (winlab)

Plenary Title:	Towards Digital Twinning: A Networking Journey from Closed Systems Towards Open Programmable Concepts, Architecture and Infrastructure
Dates:	November 17, 2021 1:00 PM (EST)
Presenter:	Professor Marco Ruffini (Trinity College Dublin)
Location:	Webinar
	This talk discussed how networking concepts have evolved in the past 10 years from distributed, closed and independent systems to become centralized, open and convergent. It showed how programmability has been a key factor across all such changes and focus on the next evolution, which brings the use of AI techniques to a new level, with the concept and operations of Digital Twins. It will finally touch on a key topic for this development, which is the access to open research infrastructure. The presentation will provide examples of this transition, showcasing research work from the presenter and his research group including from the NGI Atlantic project.
Evidence:	http://www.winlab.rutgers.edu/events/Search

Co-funded by the Horizon 2020 Framework Programme of the European Union

8.3 IEEE Future Networks - Enabling 5G and Beyond: Future Networks Testbed Requirements, Challenges and Opportunities Testbed Workshop

Plenary Title:	Future Networks Testbed Requirements, Challenges and Opportunities Testbed Workshop
Dates:	7th-8th February 2022 09:00-9:30am (EST)
Presenter:	Professor Marco Ruffini (Trinity College Dublin)
Location:	Webinar
Evidence:	https://futurenetworks.ieee.org/conferences/fn-testbed-workshop
Video:	Event Recording - Day 1: https://us06web.zoom.us/rec/share/8lnWAgoz7lQ2lQY7p2DtaLr3uFPor9-D4r82IWyZeyoEDk0dS06vL_tCHiUp1hhk.3I3GyaEq3PeME2S1

8.4 Publications

While we have not yet published this results in a conference or journal paper, work is ongoing for two publication on the results from the two experiments described above. In addition, early results were presented at invited talks at the Photonics in Switching (https://www.optica.org/en-us/meetings/topical_meetings/photonics_in_switching_and_computing/) and Asian conference on Photonics (ACP - http://www.acpconf.com), both in Q4 of 2021.

8.5 Testbed Interconnection

The dedicated testbed interconnection link was established in January 2022. The goal is to keep this link active for at least 2 years after completion of the project.

8.6 OpenIreland GitHub repository Contributions

During the project, we have contributed to open-source repositories in various ways including code and testing such as OpenAirInterface (gNB, NSA, and SA Core), Mininet-Optical (pending public release), and an infrastructure setup and control framework for the COSMOS testbed.

Repository Title:	**Public Automated NGI Atlantic experiment developed using Ansible, which executes an automated experiment across the NGI Atlantic testbeds of OpenIreland and COSMOS.**

Co-funded by the Horizon 2020 Framework Programme of the European Union

URL:	https://github.com/dcollin5/OpenIreland-NGI-Atlantic.git

Repository Title:	**Mininet-Optical (Not released yet)**
URL:	https://github.com/Mininet-Optical/mininet-optical

Repository Title:	**Dedicated-BWPs-dynamic-switching**
URL:	https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/Dedicated-BWPs-dynamic-switching

Repository Title:	**OpenAirInterFace5G**
URL:	https://gitlab.eurecom.fr/oai/openairinterface5g

Repository Title:	**OpenAirInterFace5G oai-cn5g-fed (OAI5G SA Core)**
URL:	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.git

8.7 Impacts

The following is a consolidated and self-contained discussion of the achieved impacts of the project, in relation to the overall impacts of the NGI initiative and for **the entire duration of the project**, namely:

<u>Impact 1: Enhanced EU – US cooperation in Next Generation Internet, including policy cooperation.</u>

The main impact of this collaboration was to facilitate the use of two Open Networking testbeds to communities in the US and Europe. There are economies of scales when it comes to testbed experimentation due to the steep learning curve in remote access of external infrastructure. To support barrier reduction, the extension of the Mininet-Optical emulation toolkit to support the OpenIreland testbed will enable experimenters to become familiar with optical infrastructure available in OpenIreland and COSMOS testbeds prior to real experimentation. This is outlined further in Deliverable 2 Section 5.2 and Deliverable 2 Section 6.1. It is our intension to release the Mininet-Optical framework to the general public under a MIT Licence in the near future. This work has involved significant cooperation between EU and US counterparts. Additionally, by designing a common access and control scheme for the two testbeds, we have simplified experimenters' access to a trans-Atlantic testbed facility. Part of this functionality was developed at the mid-point of the project, as outlined in Deliverable 2 - Section 5.1, using the jFed Experimenter toolkit, and an organized set of Ansible scripts available from a public GitHub repository. These scripts support configuration of the Microstack-OpenStack environment across bare metal COSMOS infrastructure supporting virtualisation and higher-layer configuration. This is supported by the OpenIreland-COSMOS 10G inter-testbed connection, which enables experimenters to create experiments on OpenIreland, and execute them on the COSMOS facility. Note, the set-up of the dedicated testbed interconnection link between OpenIreland and COSMOS testbeds was achieved in January 2022. The goal is to keep this link active for at least 2 years after completion of the project. Finally, we have also demonstrated that there has been very effective collaboration between TCD and Polimi in the training of machine learning models to assign WDM channels in the transmission spectrum of a fibre. This has also resulted in production of a dashboard supporting optical experimentation that simplifies management, configuration, automation and data collection enabling the possibility of performing experiments remotely. The dashboard consists of 7 tabs: 1) Topology, 2) ROADMs configuration, 3) OSA configuration, 4) Turn ON/OFF wavelengths, 5) OSA monitor, 6) ROADM monitor and 7) Data Collection.

Impact 2: Reinforced collaboration and increased synergies between the Next Generation Internet and the Tomorrow's Internet programmes.

The priority area addressed in this project is "Open Internet architecture renovation", since one of the main features of both testbeds is the use of open source and open interface architectures, which enable experimentation on disaggregated architectures and control systems. Indeed, the testbeds address network disaggregation across several domains: the wireless domain, with the use of Software Defined Radio using OpenAirInterface 5G and srsRAN, and consideration for the OpenRAN specifications. The optical domain, with the use of whitebox optical components (i.e., SDN-enabled ROADMs, amplifiers and transponders) and Open Disaggregated Transport Network (ODTN) software specifications (from the ONF as well as from the Telecom Infra Project - TIP). The motivation for using these open technologies is to support experimentation and publication. Consequently, we are also committed to contributing tools developed as open source repositories, as exemplified by our contribution to opensource initiatives such as OpenAirInterface, and Mininet-Optical. Moreover, simple experiment setup and reproducibility across OpenIreland and testbeds (such as COSMOS) can be offered by utilising frameworks such as Canonical Microstack and using the Jfed experimenter toolkit. A description of some of the efforts in this space are available in Deliverable 2 - Section 5.1.

Impact 3: Developing interoperable solutions and joint demonstrators, contributions to standards.

Co-funded by the Horizon 2020 Framework Programme of the European Union

The fact that researchers will be able to work across the two platforms will boost the EU-US research activities considerably. We have extended the Mininet-Optical emulation toolkit to support the OpenIreland testbed facility by integrating support for NETCONF SDN management protocol, a prerequisite to using the OpenIreland testbed. A description and analysis of this work is available in Deliverable 2 - Section 5.2 and Section 6.1. This is an important step towards training researchers to use both testbed environments. Having researchers that are capable of operating across both testbeds will naturally increase the inclusion of US academics into European collaborative proposals. In addition, some EU countries (including Ireland) have already frameworks for cooperation with the US. We believe this joint testbed will increase the participation of network researchers into such bilateral partnerships.

The proposal also fits with the following NGI enabling technologies: 5G (or beyond), AI, ML and resilience. One of the important aspects for the proposed collaboration with COSMOS is to develop strategies for signal quality monitoring, so that data can be collected dynamically and fed to machine learning algorithms for the estimation of optical quality of transmission (QoT). QoT is needed for efficient and reliable channel provisioning and the automation methods considered here are essential to achieve autonomous network operation—moving away from the mostly manual methods used today. Another central aspect is the ability to carry out research on 5G systems, with cross-optimisation of optical and wireless capacity (i.e., using Passive Optical Networks, metro transmission systems and wireless 5G nodes). We have made several strides towards this goal at the end of the project supported by the work of Prof. Tornatore from the Polytechnic University of Milan and his team. Data such as ASE Noise, NLI Noise, Power dBm, OSNR/g-OSNR, has been collected by the TCD team and processed by Prof. Tornatore's team, to support the extension and validation of the QoT estimation algorithm and a transfer learning algorithm developed during the Metro-Haul EU project. Analysis of this work has been presented in Deliverable 2 - Section 5.3 and Section 6.2, and Deliverable 3 – Section - Experiment 1: Quality of Transmission (QoT) estimation of optical systems. Furthermore the development of the optical dashboard outlined in Deliverable 3 Section 4.1 has also resulted in production of a dashboard supporting optical experimentation that simplifies management, configuration, automation and data collection enabling the possibility of performing experiments remotely.

Impact 4: An EU - US ecosystem of top researchers, hi-tech start-ups / SMEs and Internet-related communities collaborating on the evolution of the Internet

Including a third-party institution (Polytechnic University of Milan) in this proposal has helped design the system in a way that is easily accessible also by other third parties. In addition, considering that the Professors involved in the experiments (Prof Ruffini, Kilper, Seskar and Tornatore) are of high profile and well known within their community, publications that will follow from such experiments will attract several other experimenters to the two testbeds, and Mininet-Optical emulation environment. At the end of the project three dissemination opportunities have already been produced showcasing some of the experimentation that can be performed across OpenIreland and COSMOS testbeds using Mininet-Optical emulation software. Our goal is at least 2 additional publications based on the results gathered in the final months of the project.

9 Conclusions and Future Work

The extension of the Mininet-Optical emulation toolkit to support the OpenIreland testbed will enable experimenters to become familiar with optical infrastructure available in OpenIreland and COSMOS testbeds prior to real experimentation. This will act as an important first step in training future researchers and experimenters from the US and Europe towards optimisation of optical metro equipment and support the development of research hypothesis to be tested. The additional work carried out in this project lays the foundation for advanced research towards cross-optimisation of optical and wireless capacity using Passive Optical Networks (PON), metro transmission systems, and wireless 5G nodes. Towards this objective, the optical GUI developed will simplify the management, configuration, and data collection from metro network optical equipment in the OpenIreland testbed, and enable the possibility of performing experiments remotely. One of the most beneficial parts of this framework is that data can be collected dynamically and fed to machine learning algorithms for the estimation of optical quality of transmission (QoT), which is needed for efficient and reliable channel provisioning and the automation methods considered essential to achieve autonomous network operation. This work can be easily extended into Passive Optical Networks using the same principles. ML algorithms developed can also be further broadened into the wireless domain supporting policies and strategies for improving wireless signal quality and optimal data delivery using opensource software frameworks such as OpenAirInterface 5G based on O-RAN principles. The fact that researchers will be able to work seamlessly across the OpenIreland and COSMOS testbeds using these tools will boost the EU-US research activities and strengthen research outputs considerably. It is envisaged these initiatives, testbeds, and frameworks will enhance European and US researchers towards better international collaboration.

10 References

1. Both, Cristiano, et al. "Futebol control framework: Enabling experimentation in convergent optical, wireless, and cloud infrastructures." IEEE Communications Magazine 57.10 (2019): 56-62.
2. Intel X710 10 GbE Network Adapter Family Product Guide, https://lenovopress.com/tips1229-intel-x710-10gbe
3. ETSI TS 138 211 V15.3.0 (2018-10) 5G; NR; Physical channels and modulation (3GPP TS 38.211 version 15.3.0 Release 15), https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.03.00_60/ts_138211v150300p.pdf
4. NGI Atlantic BWPs v4 video Final, https://www.youtube.com/watch?v=3gjTQSGCtck
5. Lantz, Bob, et al. "SDN-controlled Dynamic Front-haul Provisioning, Emulated on Hardware and Virtual COSMOS Optical x-Haul Testbeds." Optical Fiber Communication Conference. Optical Society of America, 2021.
6. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

Co-funded by the Horizon 2020 Framework Programme of the European Union

7. Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. PMLR, 2016.
8. Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

11 Glossary

5G	**Fifth Generation (mobile/cellular networks)**
AI	Artificial Intelligence
BWP	Bandwidth Part
COSMOS	Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deployment
C-TAG	Customer Tag (vlan terminology)
DRL	Deep Reinforcement Learning
FUTEBOL	Federated Union of Telecommunications Research Facilities for an EU-Brazil Open Laboratory
LTE	Long Term Evolution (Radio)
ML	Machine Learning
Mux	Multiplexor
Demux	Demultiplexor
OAI	Open Air Interface
OAI5G	Open Air Inteface 5G
ODTN	Open Disaggregated Transport Network
ONIS	Optical Network Intelligence System
ONOS	Open Network Operating System
ONS	Optical Noise Signals
O-RAN	Open Radio Access Network
OSA	Optical Spectrum Analyser
OSNR	Optical Signal Noise Ratio

Co-funded by the Horizon 2020 Framework Programme of the European Union

MTU	Mean Traffic Unit
NSF	National Science Foundation
PAWR	Platforms for Advanced Wireless Research
Polimi	Politecno di Milano
PON	Passive Optical Network
PPO	Proximal Policy Optimisation
QoT	Quality of Transmission
ROADM	Reconfigurable Optical Add Drop Multiplexor
S-TAG	Service Tag (vlan terminology)
SA	Stand Alone
SBI	South Bound Interface
SDN	Software Defined Networks
SDR	Software Defined Radio
TCD	Trinity College Dublin
TRL	Technology Readiness Level
UE	User equipment
Vlan	Virtual local area network

Co-funded by the Horizon 2020
Framework Programme of the European Union

12 Appendix 1. OAI5G Dedicated-BWPs-dynamic-switching branch with changes supporting ZMQ integration

```
ubuntu@ubuntu-PowerEdge-R740xd:~/openairinterface5g$ git diff

+++ b/cmake_targets/CMakeLists.txt

 target_link_libraries (lte-softmodem ${LIBXML2_LIBRARIES})

  target_link_libraries (lte-softmodem pthread m ${CONFIG_LIB} rt crypt ${CRYPTO_LIBRARIES} ${OPENSSL_LIBRARIES} sctp ${PROTOBUF_LIB}  ${CMAKE_DL_LIBS} ${LIBYAML_LIBRARIES})

@@ -2995,7 +2995,7 @@ target_link_libraries (nr-softmodem

   ITTI ${FLPT_MSG_LIB} ${ASYNC_IF_LIB} ${FLEXRAN_AGENT_LIB} LFDS7 ${MSC_LIB} ${RAL_LIB} ${NAS_UE_LIB} RRC_LIB NR_RRC_LIB

   NGAP_LIB NGAP_GNB S1AP_LIB S1AP_ENB L2_LTE_NR L2_NR MAC_NR_COMMON NFAPI_COMMON_LIB NFAPI_LIB NFAPI_VNF_LIB NFAPI_PNF_LIB NFAPI_USER_LIB

   X2AP_LIB X2AP_ENB F1AP_LIB F1AP M2AP_LIB M2AP_ENB M3AP_LIB M3AP_ENB ${PROTO_AGENT_LIB} ${FSPT_MSG_LIB}

-  -Wl,--end-group z dl)

+  -Wl,--end-group z dl zmq)



+++ b/openair2/LAYER2/NR_MAC_gNB/gNB_scheduler_dlsch.c

@@ -779,6 +779,11 @@ void pf_dl(module_id_t module_id,

       continue;

     }



+//    if ((106 - (n_rb_sched + min_rbSize)) >= sched_ctrl->sched_pdcch.BWPSize) {

+//          LOG_W(NR_MAC,"Cannot allocate DL resources for UE %d with BWPsize %d (allocated: %d)\n", UE_id, sched_ctrl->sched_pdcch.BWPSize,106 - (n_rb_sched + min_rbSize));

+//      continue;

+//    }

+

     /* Find a free CCE */

     const int cid = sched_ctrl->coreset->controlResourceSetId;

     const uint16_t Y = get_Y(cid%3, slot, UE_info->rnti[UE_id]);

@@ -1012,6 +1017,58 @@ void nr_bwp_switch(module_id_t module_id,

   }

 }



+// TODO: Periodic switching algorithm - to be developed a better switching algorithm.

+//  At the moment, and for demo purposes, we are going through all the bands. The UE stays a few seconds in each one.

+void schedule_nr_bwp_zmq_switch(module_id_t module_id, frame_t frame, sub_frame_t slot) {

+

+      if (RC.nrmac[module_id]->UE_info.num_UEs == 0) {

+            return;

+      }

+

+      // TODO: To develop a better switching algorithm.

+      //  At the moment, and for demo purposes, we are going through all the bands. The UE stays a few seconds in each one.

+      if (NR_zmq_t.zmq_update == true) {
```

Co-funded by the Horizon 2020 Framework Programme of the European Union

```
+
+           NR_UE_info_t *UE_info = &RC.nrmac[module_id]->UE_info;
+           const NR_list_t *UE_list = &UE_info->list;
+
+           for (int UE_id = UE_list->head; UE_id >= 0; UE_id = UE_list->next[UE_id]) {
+
+
+                   if (UE_id == NR_zmq_t.zmq_user_id) {
+                           LOG_I(NR_MAC, "1 - UE_id == NR_zmq_t.zmq_user_id");
+                           if (&UE_info->active[UE_id] == false)
+                                   return;
+
+                           NR_UE_sched_ctrl_t *sched_ctrl = &UE_info->UE_sched_ctrl[UE_id];
+                           int *bwp_switch_timer =
+                                           &sched_ctrl->bwp_switch_info.bwp_switch_timer;
+                           if (UE_info->Msg4_ACKed[UE_id] != true || sched_ctrl->bwp_switch_info.bwp_switch_state == BWP_SWITCH_RUNNING)
+                                   return;
+
+                           int bwp_id = UE_info->UE_sched_ctrl[UE_id].active_bwp->bwp_Id;
+                           LOG_I(NR_MAC,
+                                           "++++++++++ UE_info->Change to ZMQ BWP from bwp_id %d  to NR_zmq_t.zmq_bwp_id%d\n",
+                                           bwp_id, NR_zmq_t.zmq_bwp_id);
+
+                           if (sched_ctrl->bwp_switch_info.bwp_switch_state
+                                           == BWP_SWITCH_INACTIVE) {
+                                   if (NR_zmq_t.zmq_bwp_id != bwp_id) {
+                                           bwp_id = NR_zmq_t.zmq_bwp_id;
+
+                                           sched_ctrl->bwp_switch_info.current_bwp =
+                                                           UE_info->UE_sched_ctrl[UE_id].active_bwp->bwp_Id;
+                                           sched_ctrl->bwp_switch_info.next_bwp = bwp_id;
+                                           sched_ctrl->bwp_switch_info.bwp_switch_state = BWP_SWITCH_TO_START;
+                                           LOG_I(NR_MAC, "++++++++++ UE_info-switching in progress ++++++++");
+                                   }
+                                   NR_zmq_t.zmq_update = false;
+                           }
+                   }
+           }
+     }
+}
+


// TODO: Implementation of a algorithm to perform:
//  - the BWP switch trigger: sched_ctrl->bwp_switch_info.bwp_switch_state = BWP_SWITCH_TO_START
//  - the BWP selection:      sched_ctrl->bwp_switch_info.next_bwp = bwp_id
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
-  schedule_nr_bwp_periodic_switch(module_id, frame, slot, 30000); // Temporary periodic algorithm for demo purposes

+  //schedule_nr_bwp_periodic_switch(module_id, frame, slot, 30000); // Temporary periodic algorithm for demo purposes

+  schedule_nr_bwp_zmq_switch(module_id, frame, slot);



+++ b/openair2/LAYER2/NR_MAC_gNB/gNB_scheduler_ulsch.c

@@ -1289,6 +1289,11 @@ void pf_ul(module_id_t module_id,



     NR_UE_sched_ctrl_t *sched_ctrl = &UE_info->UE_sched_ctrl[UE_id];



+//    if ((106 - (n_rb_sched + min_rbSize)) >= sched_ctrl->sched_pdcch.BWPSize) {

+//          LOG_W(NR_MAC,"Cannot allocate UL resources for UE %d with BWPsize %d (allocated: %d)\n", UE_id, sched_ctrl->sched_pdcch.BWPSize,106 - (n_rb_sched + min_rbSize));

+//      continue;

+//    }

+

+++ b/openair2/LAYER2/NR_MAC_gNB/main.c

@@ -42,6 +42,15 @@

 #include "common/ran_context.h"

 #include "executables/softmodem-common.h"



+/* NGI Atlantic*/

+#include "zmq.h"

+#include "stdio.h"

+#include "unistd.h"

+#include "string.h"

+#include "assert.h"

+#include "stdbool.h"

+#include "mac_proto.h"

+

 extern RAN_CONTEXT_t RC;




@@ -67,6 +76,37 @@ void *nrmac_stats_thread(void *arg) {

   return NULL;

 }



+void *zmq_thread(void *arg) {

+

+    //  Socket to talk to clients

+    void *context = zmq_ctx_new ();

+    void *responder = zmq_socket (context, ZMQ_REP);

+    int rc = zmq_bind (responder, "tcp://*:5555");
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
+    assert (rc == 0);
+    //NR_zmq_t stru_zmq;
+
+    while (oai_exit == 0) {
+        char buffer [10];
+        char zmq_user_id, zmq_bwp_id;
+        bool zmq_update=false;
+        zmq_recv (responder, buffer, 10, 0);
+        printf ("%s\n",buffer);
+        NR_zmq_t.zmq_user_id = atoi(&buffer[0]);
+        NR_zmq_t.zmq_bwp_id = atoi(&buffer[2]);
+        printf ("user_id %d\n",NR_zmq_t.zmq_user_id);
+        printf ("bwp_id %d\n",NR_zmq_t.zmq_bwp_id);
+        NR_zmq_t.zmq_update = true;
+
+//        NR_zmq_t.zmq_update = zmq_user_id;
+//        NR_zmq_t.zmq_user_id = zmq_user_id;
+//        NR_zmq_t.zmq_bwp_id = zmq_bwp_id;
+        sleep (1);          //  Do some 'work'
+        zmq_send (responder, buffer, 5, 0);
+    }
+    return NULL;
+}
+
+

   const NR_UE_sched_ctrl_t *sched_ctrl = &UE_info->UE_sched_ctrl[UE_id];
+
+    /***********dc***************/
+    const NR_BWP_Downlink_t *bwp = sched_ctrl->active_bwp;
+    const int bwp_id = bwp ? bwp->bwp_Id : 0;
+    LOG_I(NR_MAC,"UE_info->%d bwp_Id %d \n",UE_id, bwp_id);
+    /***********dc***************/
+
      pthread_create(&RC.nrmac[i]->stats_thread,NULL,nrmac_stats_thread,(void*)RC.nrmac[i]);


+      //thread supporting BWPs changes
+      pthread_create(&RC.nrmac[i]->zmq_thread,NULL,zmq_thread,(void*)RC.nrmac[i]);
+
+++ b/openair2/LAYER2/NR_MAC_gNB/nr_mac_gNB.h
@@ -91,6 +91,12 @@ typedef struct {
   int len;
 } NR_list_t;
```

Co-funded by the Horizon 2020
Framework Programme of the European Union

```
+struct NR_zmq {
+  bool zmq_update;
+  int zmq_user_id;
+  int zmq_bwp_id;
+} NR_zmq_t;
+
@@ -728,6 +734,8 @@ typedef struct gNB_MAC_INST_s {
   /// Pointer to IF module instance for PHY
   NR_IF_Module_t                *if_inst;
   pthread_t                     stats_thread;
+  //zmq integration
+  pthread_t                     zmq_thread;
```

13 Appendix 2: OpenIreland ZMQ client (io_zmq_client.py)

```
import zmq
import sys


context = zmq.Context()


#  Socket to talk to server
print("Connecting to hello world server…")
socket = context.socket(zmq.REQ)
socket.connect("tcp://localhost:5555")
ue_id = sys.argv[1]
bwp_id = sys.argv[2]


#  Do 10 requests, waiting each time for a response
for request in range(1):
    bwps_str = ue_id+','+bwp_id
    print("Sending request %s … ue_id %s " % (request, bwps_str))
#    socket.send(b"%s",bwps_str)
    socket.send_string(bwps_str)
    #  Get the reply.
    message = socket.recv()
    print("Received reply %s [ %s ]" % (request, message))
```