Dynamically Reconfigurable Optical-Wireless Back-haul/Fronthaul with Cognitive Control Plane for Small Cells and Cloud-RANs

D5.1 Evaluation of SDN functionalities over heterogeneous RAN technologies at UTH testbed

This project has received funding from the European Union's Framework Programme Horizon 2020 for research, technological development and demonstration

Advanced 5G Network Infrastructure for the Future Internet

Project Start Date: July 1st, 2015 Duration: 36 months

H2020-ICT-2014-2 671551 June 30th 2017 – Final Version

Project co-funded by the European Commission

Under the H2020 programme

Dissemination Level: Public

Grant Agreement Number:	671551
Project Name:	Dynamically Reconfigurable Optical-Wireless Back-haul/Fronthaul with Cognitive Control Plane for Small Cells and Cloud-RANs
Project Acronym:	5G-XHaul
Document Number:	D5.1
Document Title:	Evaluation of SDN functionalities over heteroge-neous RAN technologies at UTH testbed
Version:	Final
Delivery Date:	June 30th 2017 (**July 15th 2017**)
Responsible:	**UTH**
Editor(s):	Kostas Choumas (**UTH**)
Authors:	Kostas Choumas, Dimitris Giatsios, Paris Flegkas (**UTH**), Daniel Camps-Mur (**I2CAT**)
Keywords:	NITOS, SDN, slicing, Traffic Engineering, mmWave
Status:	Final
Dissemination Level	**Public** / Confidential
Project URL:	http://www.5g-xhaul-project.eu/

Table of Contents

D5.1 EVALUATION OF SDN FUNCTIONALITIES OVER HETEROGENEOUS RAN TECHNOLOGIES AT UTH TESTBED.......... 1

LIST OF FIGURES.......... 5

LIST OF TABLES.......... 6

EXECUTIVE SUMMARY.......... 7

1 INTRODUCTION.......... 8

2 NITOS DESCRIPTION.......... 9

2.1 NITOS indoor (RF isolated) testbed.......... 9

2.2 OpenFlow Switches.......... 10

2.3 Wireless interfaces.......... 10

2.4 NITOS indoor testbed architecture.......... 11

2.5 NITOS software.......... 12

2.5.1 cOntrol and Management Framework - OMF.......... 12

2.5.2 NITOS Scheduler.......... 13

3 MMWAVE NODES IN NITOS.......... 14

3.1 Deployment setup.......... 14

3.2 Connectivity, validation and integration tests.......... 15

3.3 SDN control of mmWave nodes.......... 17

4 DEMOS AND EXPERIMENTS.......... 19

4.1 Slicing and tunnelling demo (UTH).......... 19

4.1.1 Setup and experiment description.......... 19

4.1.2 Experiment results.......... 21

4.2 TE Sub-6 demo.......... 23

4.2.1 Setup, test definition, metrics 23
4.2.2 Experiment results 24

5 JOINT SUB-6 AND MMWAVE EXPERIMENTS 27

5.1 Deployment setup, controllers, interfaces 27

5.2 Scenario description 28

5.3 Experiment results 29

6 SUMMARY AND CONCLUSIONS 30

7 REFERENCES 31

8 ACRONYMS 32

List of Figures

Figure 1: Icarus Node. 9
Figure 2: NITOS indoor testbed's server machine. 10
Figure 3: NITOS indoor testbed’s OpenFlow switch. 10
Figure 4: Wistron CM9. 11
Figure 5: Atheros 9380. 11
Figure 6: NITOS indoor testbed network architecture. 11
Figure 7: NITOS facility architecture. 12
Figure 8: OMF 6 Architecture. 13
Figure 9: NITOS Scheduler user interface. 13
Figure 10: BWT nodes topology in NITOS indoor testbed. 14
Figure 11: Deployment of BWT nodes at NITOS indoor testbed. 14
Figure 12: Network topology of mmWave nodes. 15
Figure 13: UDP throughput between BWT nodes, on one direction. 15
Figure 14: UDP throughput between BWT nodes, on the other direction. 16
Figure 15: TCP throughput between BWT nodes, on the one direction. 16
Figure 16: TCP throughput between BWT nodes, on the other direction. 16
Figure 17. Structure of SDN agent in mmWave device. 17
Figure 18. Setup for validation of the SDN agent. 17
Figure 19: P2P Throughput and link RSSI observed by the SDN controller when sweeping the RX gain in the transmitter. 18
Figure 20: (a) 5G-XHaul Transport Control Plane architecture, (b) Slice abstraction towards tenant. 19
Figure 21: The slicing mechanism. 20
Figure 22: Wireshark capture of a PBB packet. 20
Figure 23: Network topology for the slicing experiment. 21
Figure 24: VNFs of the same slice are connected. 22
Figure 25: VNFs from different slices are not able to ping each other. 23
Figure 26: LTE femtocell and UE in NITOS. 23
Figure 27: Network topology in the NITOS testbed. 24
Figure 28: UDP experiment 25
Figure 29: TCP Experiment. 26
Figure 30: Experimentation topology of joint sub6 and mmWave experiment. 27
Figure 31: Scenario of the joint sub6 and mmWave experiment. 28
Figure 32: Throughput results. 29

List of Tables

Table 1: Icarus nodes' specifications. 9
Table 2: NITOS indoor testbed's server specifications. 10

Executive Summary

This deliverable provides an evaluation of the SDN functionality over heterogeneous RAN technologies, such as Sub-6 and mmWave. The evaluation is carried out in the UTH testbed, named NITOS. The deliverable includes an analytic report of the SDN functionalities tested at NITOS, involving information of the setup used for their testing. It also analyses the results obtained.

1 Introduction

The testbed evaluation of the 5G-XHaul contributions is very crucial for the successful accomplishment of the project. This is the main focus of WP5. After the first two years of the project, NITOS is the main testbed that is used for the deployment and evaluation of the 5G-XHaul contributions, while Bristol is Open (BIO) will start being used from now on and will host the deployment and exhibition of the final demonstrator. This deliverable includes the experimentation done until now in the UTH's testbed that is named NITOS.

Organisation of the document

This deliverable is structured in the following sections. We start with a description of the NITOS testbed, included in Section 2. Section 3 describes how the BWT nodes are deployed at the NITOS testbed for enabling experimentation with the millimetre wave (mmWave) technology, as well as results of the experimentation done with this nodes for validating their normal operation. Section 4 includes the experiments done for the evaluation of the slicing and tunnelling mechanism we implemented with use of the OpenFlow technology (Section 4.1), as well as the OpenFlow based traffic engineering algorithm that we deployed on top of a 5G-XHaul WiFi mesh/ad-hoc network (Section 4.2). Section 5 presents a completed experiment that is built on top of all previous contributions, presented in Section 4, integrating the slicing/tunnelling mechanism and the OpenFlow control of the WiFi and mmWave enabled nodes. Finally, Section 6 provides a summary and the main conclusions of this deliverable.

2 NITOS description

NITOS [1] facility is open to the research community 24/7 and it is remotely accessible through the NITOS reservation tool. Parallel experimentation of different users is enabled, through the utilization of the NITOS scheduler software. The testbed is based on open-source software that allows the design and implementation of new algorithms, enabling new functionalities on the existing hardware. The control and management of the testbed is done using the cOntrol and Management Framework (OMF) open-source software. Users can perform their experiments by reserving slices (nodes, frequency spectrum) of the testbed through the NITOS scheduler that together with the OMF framework, support ease of use for experimentation and code development. NITOS supports evaluation of protocols and applications under real world settings.

NITOS facility is comprised of three wireless testbeds for experimentation with heterogeneous technologies. For the 5G-XHaul experimentation, one of these testbeds is utilized, namely the indoor RF isolated testbed, comprised of advanced powerful nodes, featuring WiFi, WiMAX and LTE support. A brief description of this testbed follows, as well as detailed descriptions of several key components they use, like the wireless interfaces and the OpenFlow switches.

2.1 NITOS indoor (RF isolated) testbed

The NITOS indoor testbed consists of 50 Icarus nodes and is deployed in an isolated environment at a University of Thessaly's campus building. Experimenters are able to run and evaluate power demanding processing algorithms and protocols in a large-scale testbed.

Icarus Node
Icarus have been developed by UTH, are equipped with 802.11a/b/g and 802.11a/b/g/n wireless interfaces and feature new generation Intel 4-core CPUs and new generation solid state drives. Figure 1 illustrates the Icarus node and more details about its specification can be found in Table 1.

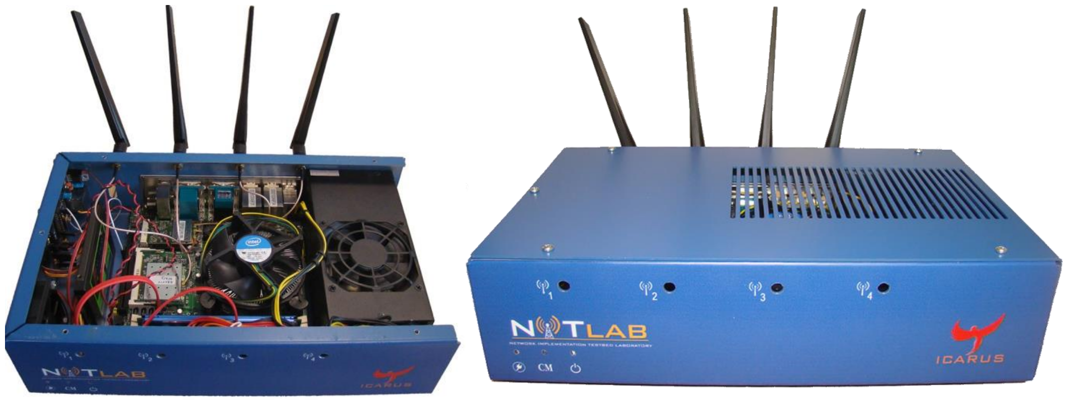


Figure 1: Icarus Node.

Table 1: Icarus nodes' specifications.

Component	Description
Motherboard	2 Gigabit Ethernet interfaces and 2 wireless interfaces
CPU	Intel Core i7-2600 Processor, 8M Cache at 3.40 GHz
RAM	4G HYPERX BLU DDR3
Wireless interfaces	Atheros 802/11 a/b/g & Atheros 802.11 a/b/g/n (MIMO)
Storage	Solid State Drive (SSD)
Power supply	350 Watt mini-ATX
Antennas	Multi-band 5dbi, operates both on 2.4 GHz & 5 GHz
Pigtails	High quality pigtails (UFL to RP-SMA)

Server machine

The NITOS indoor testbed's server machine, illustrated in Figure 2, is a HP ML350p G5, more details about its specification can be found in Table 2.

Figure 2: NITOS indoor testbed's server machine.

Table 2: NITOS indoor testbed's server specifications.

Component	Description
Processor	Intel Xeon E5-2609(4 core, 2,40 GHz)
Memory	8 GB (2 x 4 GB)
Hard Drive	2 x 500 GB SATA HDD 7200 rpm
Network Controller	1Gb 331i Ethernet Adapther 4 ports per Controller
Storage Controller	Smart Array P420i/ 5120 MB FBWC
Power Supply	460W power supply

2.2 OpenFlow Switches

NITOS indoor testbed operates two HP 3800 OpenFlow switches [2], depicted in Figure 3, which interconnect the Icarus nodes of the NITOS indoor testbed through a wired OpenFlow network. Each Icarus node has one of its Ethernet interfaces connected to the OpenFlow switch, which is also connected with the server machine, mentioned above. The OpenFlow controllers of these switches can be located at the server machine, where a variety of OpenFlow controller frameworks have been installed, such as POX, Trema, Ryu, etc.

Figure 3: NITOS indoor testbed's OpenFlow switch.

2.3 Wireless interfaces

NITOS uses several WiFi interfaces in order to provide many capabilities to NITOS users. Each WiFi interface has unique characteristics since it operates with different drivers and supports different features. To this end, UTH has acquired and equipped NITOS with the most practical and advantageous WiFi interfaces, capable to operate with open-source drivers. Bellow we list the three types that are supported for experimentation from NITOS users.

Wistron CM9 – Atheros AR5213A chip

CM9 [1], shown in Figure 4, is an IEEE802.11a/b/g 108Mbps WiFi mini-pci module in type IIIB. Built on Atheros® AR5213A chipset, CM9 is designed to IEEE802.11a/b/g standards, is compatible with all IEEE802.11b/g and IEEE802.11a WLAN and is ideally suited for integration in a wide range of OEM devices. CM-9 runs with the MadWifi [2] driver as well as with the MadWifi-old [3] driver which supports special features such as the support of the Click modular router.

Figure 4: Wistron CM9.

Figure 5: Atheros 9380.

Atheros AR9380

Atheros offers the industry's most innovative and complete portfolio of 802.11n wireless LAN chip solutions. This generation of Atheros' XSPAN 802.11n technology builds upon the company's XSPAN leadership, with enhanced performance, higher integration, smaller form factors and lower overall cost, to meet the needs of the rapidly growing 802.11n market.

AR9380 [4], shown in Figure 5, is the single-chip, dual-band (2.4 / 5 GHz), 3-stream 11n solution with PCIe interface. It packs the breakthrough Signal Sustain Technology 3 (SST3) technology that enhances the rate-over-range (RoR) performance. SST3 is a set of advanced technologies and features enabled by 802.11n including LDPC, TxBF and MLD. This interface runs the Atheros ath9k driver which is included in the open-source compat-wireless drivers [5]. It is installed in Grid nodes of NITOS testbed with 3 multi-bands antennas, in the mini-pcie slot of the Commell motherboards.

2.4 NITOS indoor testbed architecture

In this section, the network architecture of the NITOS indoor testbed is described, as illustrated in Figure 6. Two Gigabit (non-OpenFlow) Ethernet switches interconnect the nodes with NITOS Indoor server. The one is the Control switch that provides for control of experiment execution and measurement collection, and the other is the Chassis Manager (CM) switch that is dedicated in controlling the operational status of the nodes through their CM cards, which are attached on each node. The Experimental switch depicted in Figure 6 abstractly represents the two OpenFlow switches described before.

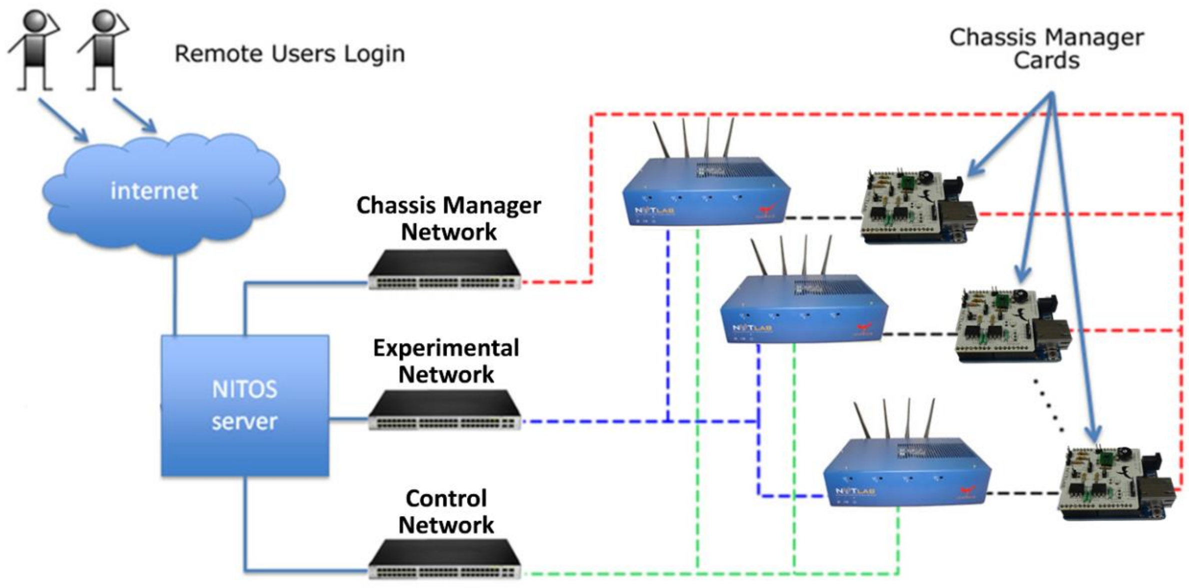


Figure 6: NITOS indoor testbed network architecture.

The overall architecture of the NITOS facility is shown in Figure 7.

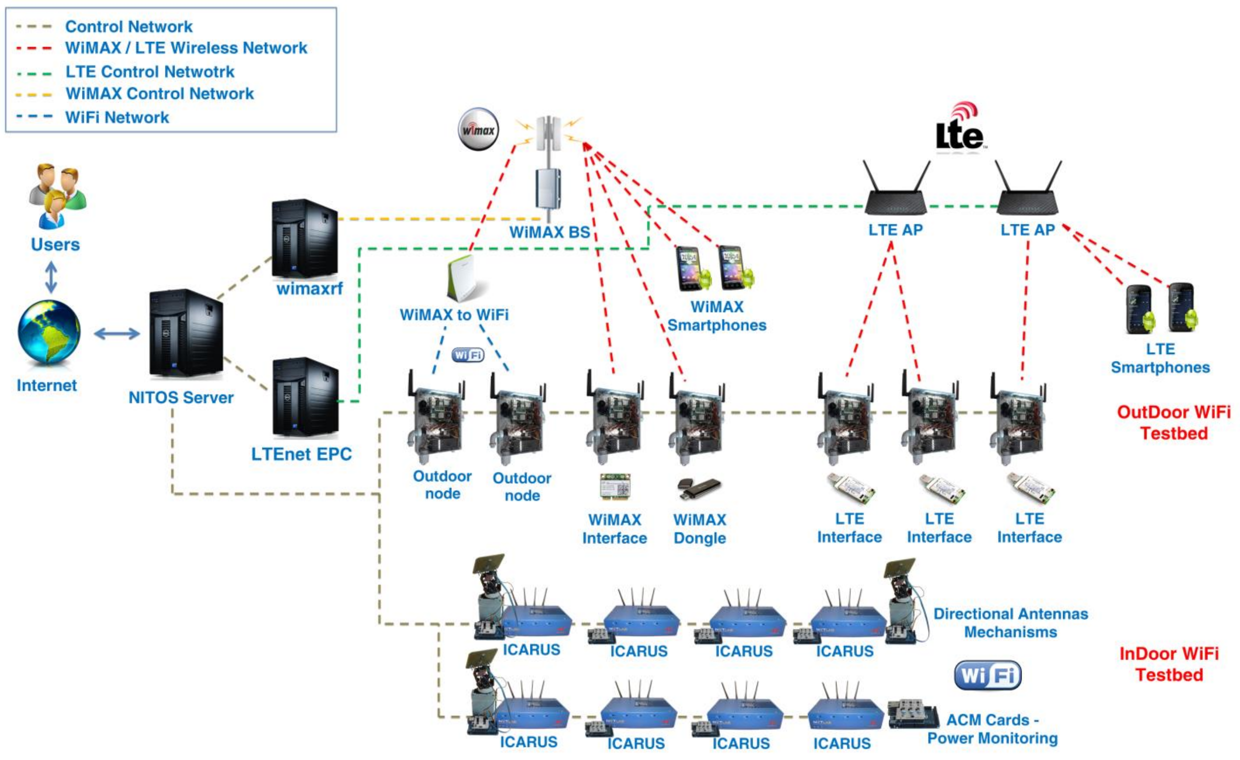


Figure 7: NITOS facility architecture.

2.5 NITOS software

The control and management of the testbed is done using the cOntrol and Management Framework (OMF) [3] open-source software. Users can perform their experiments by reserving slices (nodes, frequency spectrum) of the testbed through the NITOS Scheduler that together with OMF framework, support ease of use for experimentation and code development. In this section follows a detailed description of the basic software tools utilized by UTH in NITOS testbed.

2.5.1 cOntrol and Management Framework - OMF

The management of several heterogeneous resources is a significant issue for a testbed operator. The wireless testbeds of OpenLab are putting all their resources under a common management framework called OMF for effective management and control. OMF was initially developed in ORBIT by Winlab [6] and currently its development is being led by NICTA along with the contributions of other institutions like Winlab and UTH.

NICTA released a major update in OMF migrating from version 5.4 to version 6, which introduced radical changes in the architecture and philosophy of the framework. The main concept of the new architecture is that everything is being treated as a resource and for every resource there is a dedicated resource controller responsible for controlling it. OMF 6 moves towards to an architecture which incorporates loosely connected entities, that communicate with a "publish-subscribe" mechanism by exchanging messages that have been standardized.

In overall, OMF 6 aims to define the communication protocol between all the entities rather than their specific implementation. The messages of this communication protocol that are being exchanged are defined in the Federated Resource Control Protocol (FRCP) [4]. This novel protocol defines the syntax of the messages, but not the semantics that are subject to the different implementations concerning the various kinds of resources.

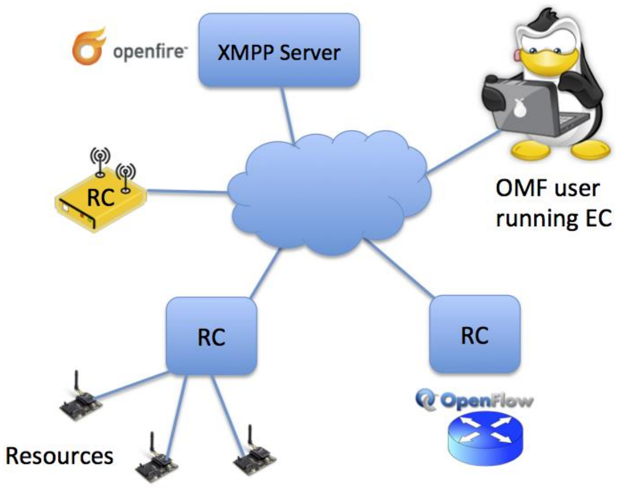


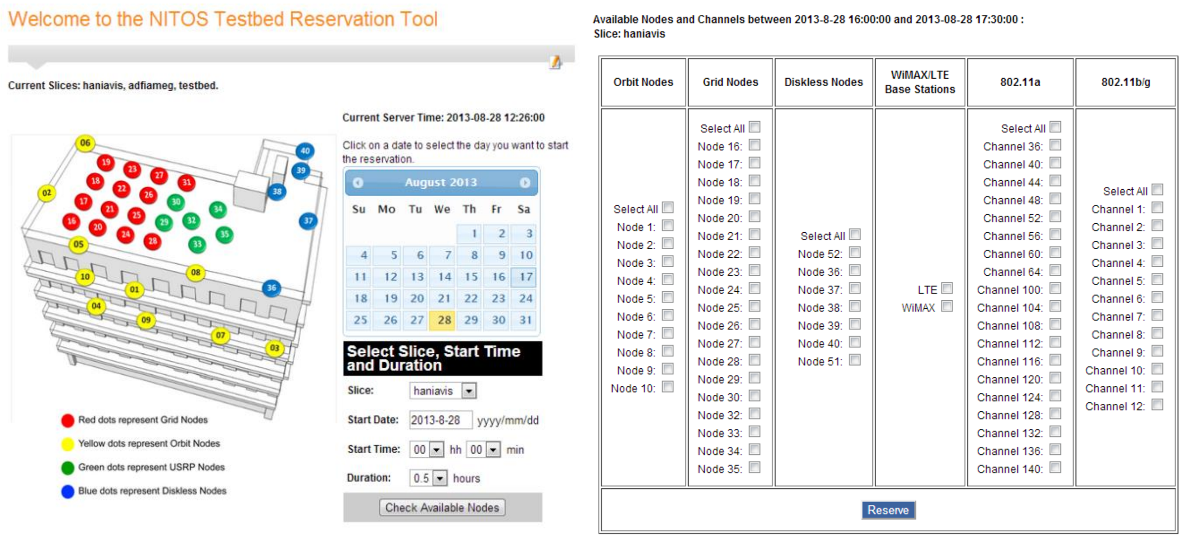
Figure 8: OMF 6 Architecture.

The architectural components of OMF 6 can be seen in Figure 8, where several RCs are deployed and an experimenter using an EC communicates with the support of Openfire XMPP server. The true power of OMF's new version comes from the capability to easily introduce new resources that are currently not supported by the existing resource controllers. If someone brings a new resource to the community, then he is free to develop an RC responsible for controlling the new resource and share it with the community. This way, OMF can be maintainable and being extended by its users, according to their needs and their different use cases.

2.5.2 NITOS Scheduler

Another tool responsible for managing the testbed's resources is NITOS Scheduler [5] developed by UTH. It is developed in the spirit of serving as many users as possible without any complicated procedures and relies its functionality on the OMF architecture. NITOS resources, namely nodes and channels, are associated with the corresponding slice during the reserved time slots, in order to enable the user of the slice to execute an experiment. UTH has enabled spectrum slicing support in NITOS, meaning that various users may use the testbed at the same time, without interfering with each other, since each one of them is using different spectrum.

As depicted in Figure 9, the wireless nodes and the spectrum channels that a user is going to use are declared during the reservation process and the scheduler does not allow for a user to choose any other resource during the execution of his/her experiment.



Welcome to the NITOS Testbed Reservation Tool

Current Slices: haniavis, adfiameg, testbed.

● Red dots represent Grid Nodes
● Yellow dots represent Orbit Nodes
● Green dots represent USRP Nodes
● Blue dots represent Diskless Nodes

Current Server Time: 2013-08-28 12:26:00

Click on a date to select the day you want to start the reservation.

Select Slice, Start Time and Duration

Slice:	haniavis
Start Date:	2013-8-28 yyyy/mm/dd
Start Time:	00 hh 00 min
Duration:	0.5 hours

Available Nodes and Channels between 2013-8-28 16:00:00 and 2013-08-28 17:30:00 :
Slice: haniavis

Orbit Nodes	Grid Nodes	Diskless Nodes	WiMAX/LTE Base Stations	802.11a	802.11b/g
Select All ☐	Select All ☐	Select All ☐	LTE ☐	Select All ☐	Select All ☐
Node 1: ☐	Node 16: ☐	Node 52: ☐	WiMAX ☐	Channel 36: ☐	Channel 1: ☐
Node 2: ☐	Node 17: ☐	Node 36: ☐		Channel 40: ☐	Channel 2: ☐
Node 3: ☐	Node 18: ☐	Node 37: ☐		Channel 44: ☐	Channel 3: ☐
Node 4: ☐	Node 19: ☐	Node 38: ☐		Channel 48: ☐	Channel 4: ☐
Node 5: ☐	Node 20: ☐	Node 39: ☐		Channel 52: ☐	Channel 5: ☐
Node 6: ☐	Node 21: ☐	Node 40: ☐		Channel 56: ☐	Channel 6: ☐
Node 7: ☐	Node 22: ☐	Node 51: ☐		Channel 60: ☐	Channel 7: ☐
Node 8: ☐	Node 23: ☐			Channel 64: ☐	Channel 8: ☐
Node 9: ☐	Node 24: ☐			Channel 100: ☐	Channel 9: ☐
Node 10: ☐	Node 25: ☐			Channel 104: ☐	Channel 10: ☐
	Node 26: ☐			Channel 108: ☐	Channel 11: ☐
	Node 27: ☐			Channel 112: ☐	Channel 12: ☐
	Node 28: ☐			Channel 116: ☐	
	Node 29: ☐			Channel 120: ☐	
	Node 30: ☐			Channel 124: ☐	
	Node 32: ☐			Channel 128: ☐	
	Node 33: ☐			Channel 132: ☐	
	Node 34: ☐			Channel 136: ☐	
	Node 35: ☐			Channel 140: ☐	

Figure 9: NITOS Scheduler user interface.

3 mmWave nodes in NITOS

3.1 Deployment setup

Within the framework of 5G-XHaul, six mmWave nodes from BWT have been deployed at the NITOS indoor testbed. The nodes have been installed at the same level than the NITOS nodes, in the topology depicted in Figure 10.

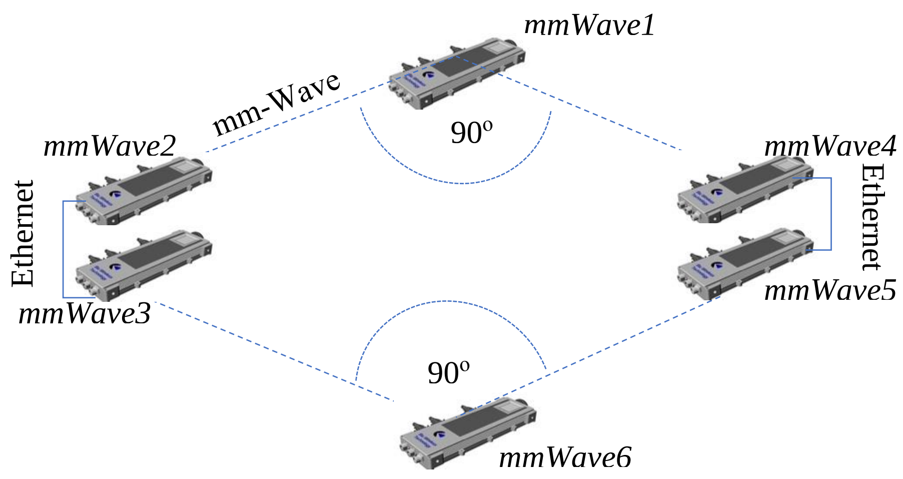


Figure 10: BWT nodes topology in NITOS indoor testbed.

The following Figure 11 shows some snapshots taken during the deployment of the BWT nodes at the NITOS indoor testbed.

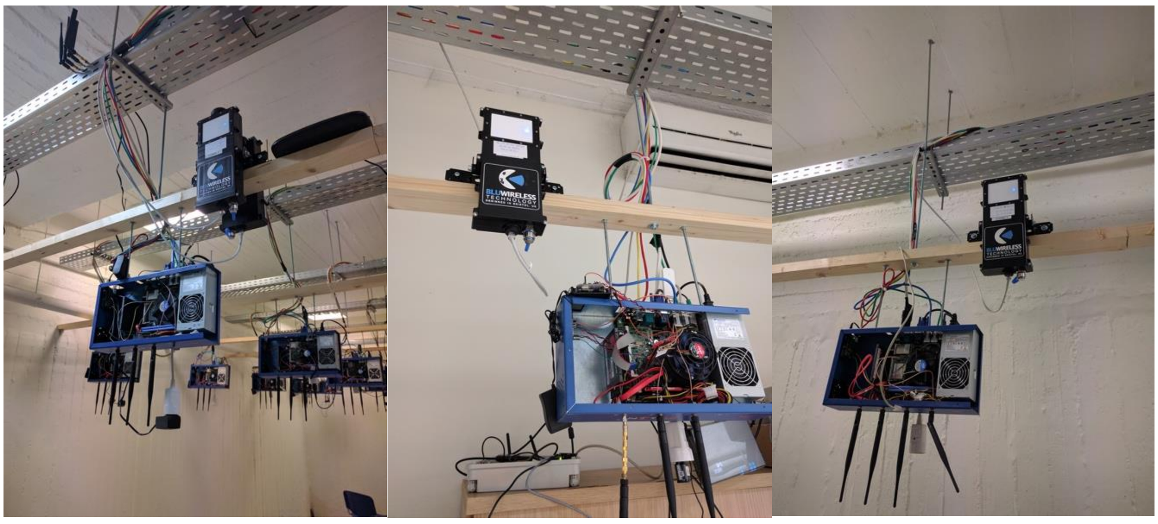


Figure 11: Deployment of BWT nodes at NITOS indoor testbed.

Each BWT node has a Power over Ethernet (PoE) interface that is connected to an HPE 1920 PoE switch, that is also connected to the HP 3800 OpenFlow switch (described before in section 2.2), as it is depicted in Figure 12. The PoE switch has been sliced with use of the VLAN technology into 6 separated broadcast domains, so that BWT nodes cannot see each other through the PoE switch. If two BWT nodes ping each other through their Ethernet interfaces, the ping packets will go through the controlled OpenFlow switch. The reason for this deployment is that BWT nodes may have bridged their PoE and wireless interfaces, which could easily produce a switching loop, if the wireless interfaces of the two nodes are configured to see each other and their Ethernet interfaces were into the same broadcast domain through the PoE switch. On the other hand, the SDN control offered by the OpenFlow switch and the BWT virtual bridges enables the loop-free management of the network traffic. Each BWT node has its own static IP address and hostname, mmWave1 to mmWave6.

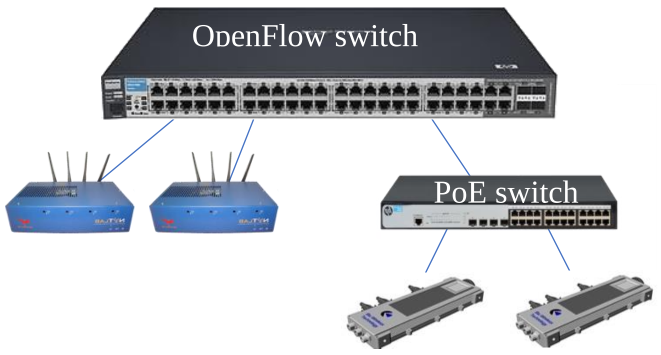


Figure 12: Network topology of mmWave nodes.

3.2 Connectivity, validation and integration tests

In this section, the experimentation on the wireless connectivity of the BWT nodes is given, presenting their support for high throughput links. Figure 13 and Figure 14 show a few indicative results of our experimentation, such as the UDP throughput between each pair of BWT nodes that are able to see each other. As it is depicted in Figure 10, the pairs of BWT nodes that are visible to each other, are mmWave1-mmWave2, mmWave1-mmWave4, mmWave3-mmWave6 and mmWave5-mmWave6. For example, in Figure 13, ‘mm1(9)-mm2(8)’ shows the maximum UDP throughput from mmWave1 to mmWave2, after using all possible MCS values (from 1 to 9) for each BWT node and keeping the ones with the highest result, which are 9 and 8 respectively in this case. In the same way, Figure 14 depicts the maximum UDP throughput from mmWave2 to mmWave1, labeled as ‘mm2(8)-mm1(9)’, since the utilized MCS is 8 and 9 for the two nodes respectively. At this point, we should mention, that the highest throughput offered by the mmWave nodes is almost 500-600Mbps, with the packets’ Aggregate MAC Service Data Unit (AMSDU) equal to 32 and Maximum Transmission Unit (MTU) equal to 1500Bytes, which is the minimum MTU and the only supported on the ‘bridge’ mode. This mode of operation happens when the wireless interface is bridged with the Ethernet interface and the traffic generator/receiver is out of the BWT node. Finally, the measurements are collected with the *iperf* software tool.

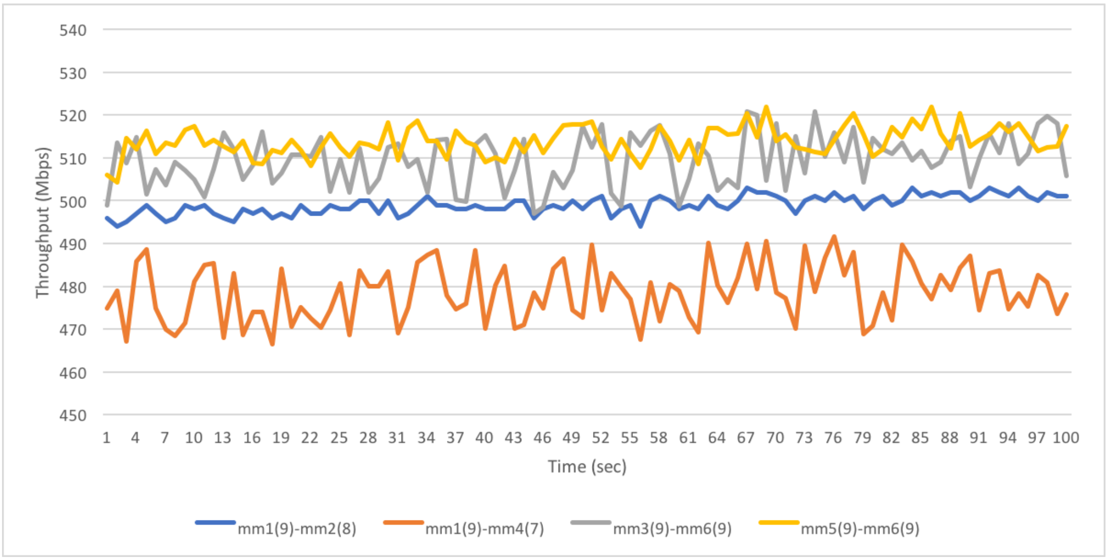


Figure 13: UDP throughput between BWT nodes, on one direction.

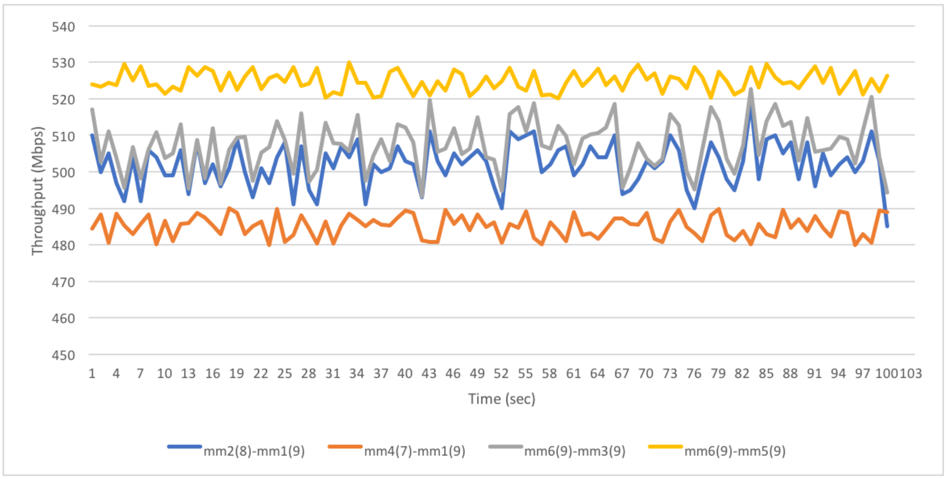


Figure 14: UDP throughput between BWT nodes, on the other direction.

Figure 15 and Figure 16 show the TCP throughput measurements collected in both ends with use again of the *iperf* software tool.

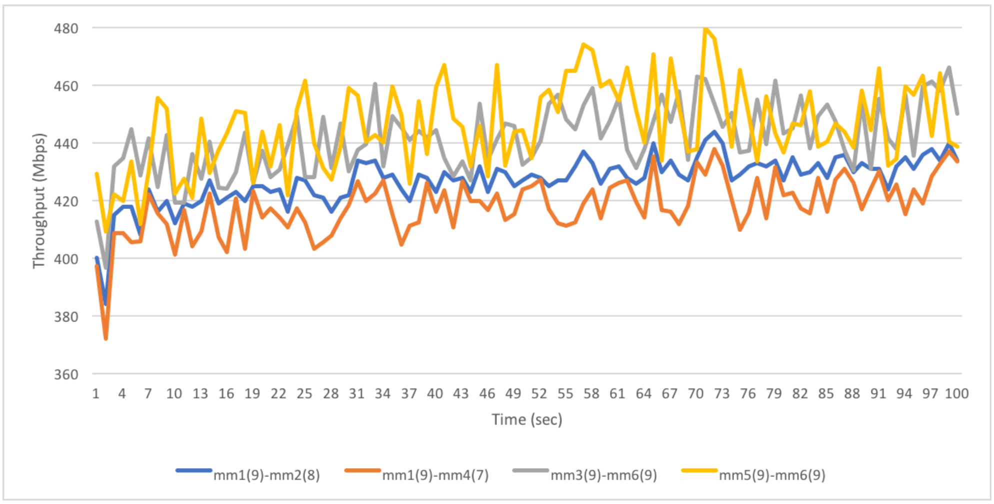


Figure 15: TCP throughput between BWT nodes, on the one direction.

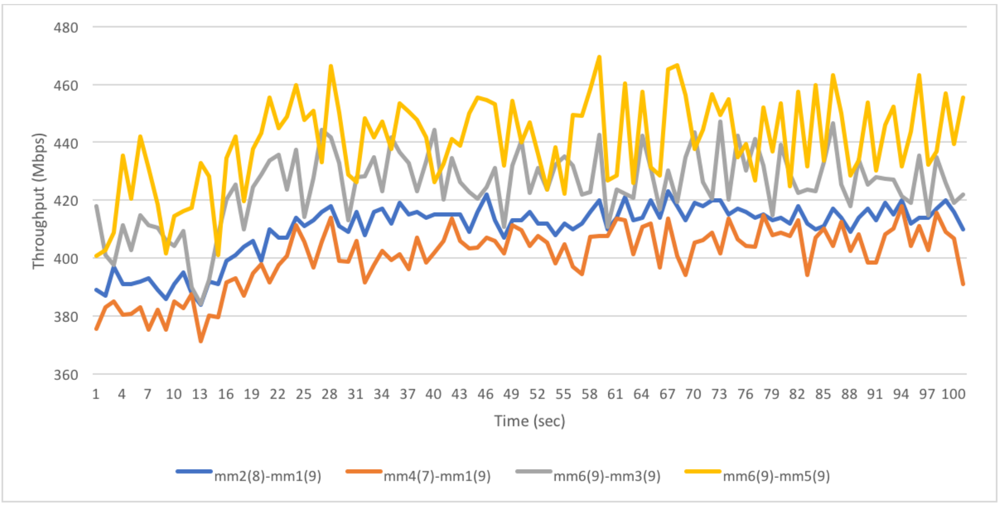


Figure 16: TCP throughput between BWT nodes, on the other direction.

3.3 SDN control of mmWave nodes

The mmWave nodes from BWT have been provided with an SDN agent developed within the framework of 5G-XHaul. The SDN agent is based on the *openvswitch* [6] software switch, which has been extended to report mmWave related metrics, such as over-the-air throughput, retransmissions, and signal strength (RSSI). To report this information the OpenFlow (OF) port statistics data structure has been extended, which has an impact both in the SDN agent and in the OF dissector plugin in the SDN controller. The port statistics extensions featured by the SDN agent are the same ones being used in the 5G-XHaul Sub6 GHz nodes, reported in deliverable D4.11 [7], which enables the control of mmWave and Sub6 devices using a single SDN controller. The SDN controller used for experimenting with the mmWave device is based on OpenDayLight [8].

Figure 17 depicts a generic setup where the SDN agent runs in the NPU (Network Processing Unit), which connects to several Lighting radios (mmWave modems). The mmWave radios offer an Ethernet abstraction towards the SDN agent. However, to ease the deployment in NITOS an integrated device, illustrated in Figure 18, has been used that contains an NPU and a single mmWave radio.

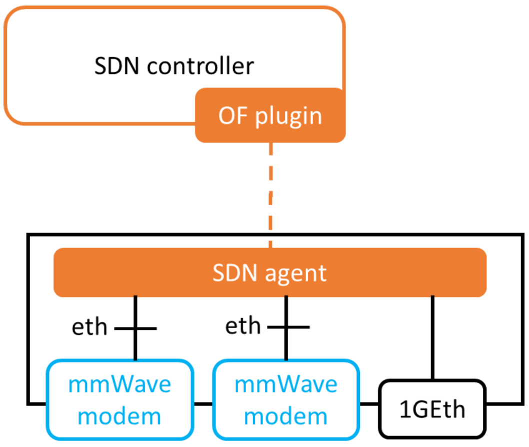


Figure 17. Structure of SDN agent in mmWave device.

To validate the integration of the SDN agent in the mmWave devices we perform an experiment where we set up a point-to-point link between two mmWave devices hosted by BWT in their office in Bristol, and we connect them to an SDN controller hosted by i2CAT in Barcelona. A VPN connection is set up between the devices and the SDN controller. Figure 18 illustrates the setup of the mmWave nodes in BWT's office, where the orange boxes are the NPUs running the SDN agents, and the black boxes the Lighting modules (mmWave modems).

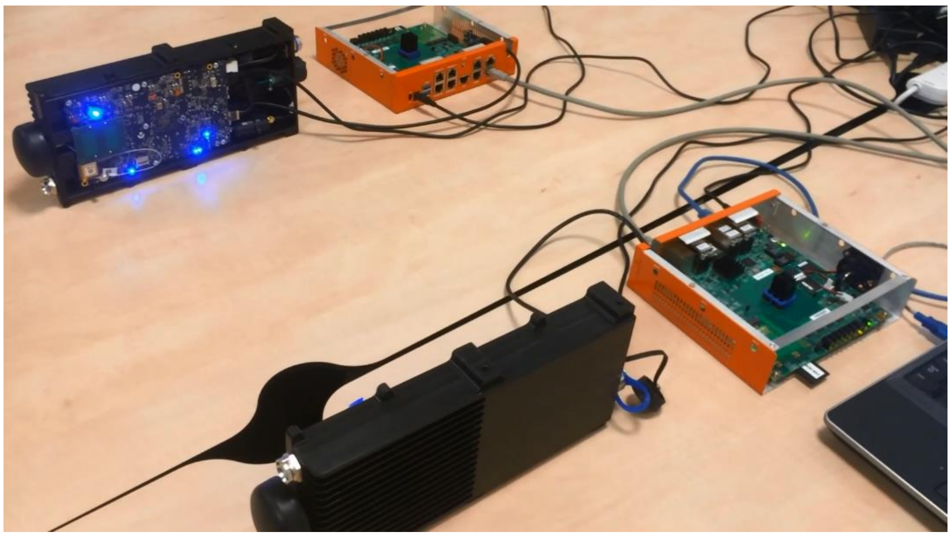


Figure 18. Setup for validation of the SDN agent.

To validate that the SDN agent can indeed measure radio parameters and report them to the SDN controller, we perform an experiment where we establish an *iperf* transmission between the two nodes that saturates the channel, whilst varying the RX gain at the transmitter following a squared signal. The effect of sweeping the RX gain is to vary the signal level at the receiver, which results in MCS adaptation and in a throughput variation. Figure 19 depicts the measurements presented at a dashboard in the SDN controller featuring the instantaneous point-to-point throughput (above) and RSSI at the receiver (below). The RSSI varies between 5 and 15 dBs, and perfectly follows the square shape that results from sweeping the RX gain. The throughput (above) also reacts to the sweep in RX gain, resulting in a signal resembling a filtered squared shape, due to the memory introduced by the link adaptation mechanism. Throughput varies between 400 Mbps and 600 Mbps.

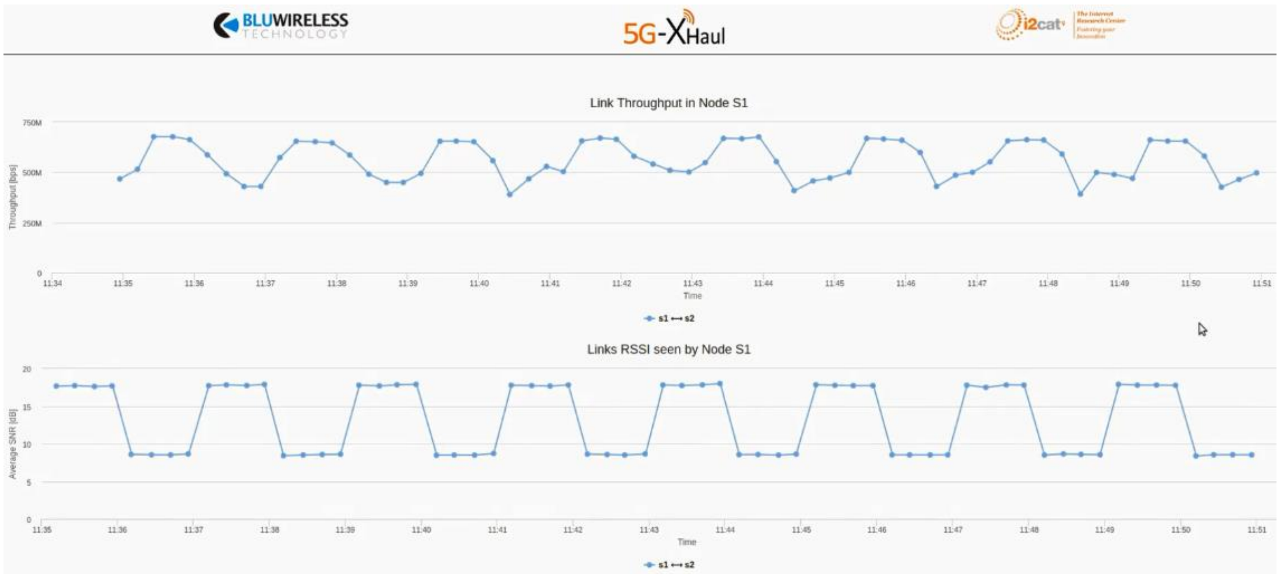


Figure 19: P2P Throughput and link RSSI observed by the SDN controller when sweeping the RX gain in the transmitter.

4 Demos and experiments

In this section, we describe the setup and results of a series of experiments we conducted at the NITOS indoor testbed, in order to verify the correct operation of the control plane mechanisms developed in the project and to evaluate them using suitable metrics.

As a reminder, in the figure below we depict the 5G-XHaul control plane architecture. More information on this architecture can be found in deliverables D3.1 [9] and D3.2 [10].

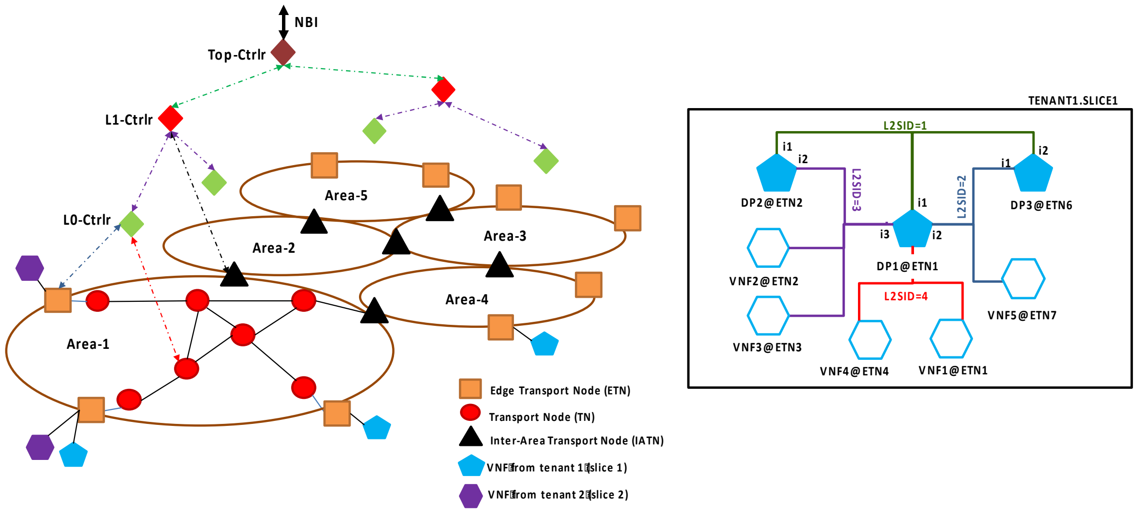


Figure 20: (a) 5G-XHaul Transport Control Plane architecture, (b) Slice abstraction towards tenant.

The experiments were implemented in the context of a single control plane area, where tenants deploy slices by placing their Virtual Network Functions (VNFs) in Edge Transport Nodes (ETNs) of the area. First experiment is presented in Section 4.1, which validates the normal operation of the developed ETNs and their slicing capabilities, as they were presented in deliverable D3.2. At this first experiment, the tunnels connecting the ETNs are based on Ethernet connections and the wireless technologies are not involved. The second experiment, presented in Section 4.2, showcases how Traffic Engineering could be feasible over a transport network that leverages on Sub-6. Finally, next Section 5 integrates everything in a single experiment that exploits all aforementioned contributions and technologies (slicing, tunnelling, Sub6) as well as mmWave.

4.1 Slicing and tunnelling demo (UTH)

The first experiment we conducted was targeted at verifying the correct operation of network slicing at the edge of the transport network and encapsulation of tenant traffic into transport specific tunnels. In this experiment, we did not use any type of wireless technology in the physical links of the transport network, as our focus was on testing the OpenFlow-controlled datapaths at the edge, the encapsulation and decapsulation procedures, and whether traffic isolation between different tenant slices is achieved using the 5G-XHaul architecture.

4.1.1 Setup and experiment description

The indoor testbed of NITOS was exploited for experimentation and evaluation of a slicing mechanism using the OpenFlow technology. Slicing is implemented by inserting a virtual L2 segment identifier in the VLAN header field of each packet, where each identifier is unique in the transport network and is not reused within or across slices. The tunneling mechanism leverages on the 802.1ah protocol (Provider Backbone Bridging – PBB or MAC-in-MAC), that enables the deployment of tunnels connecting individual networks of tenants, built on top of a common shared network. The inner Ethernet or MAC header corresponds to the original tenant frame augmented with the VLAN-ID, while the outer MAC header is used for bridging and path indication.

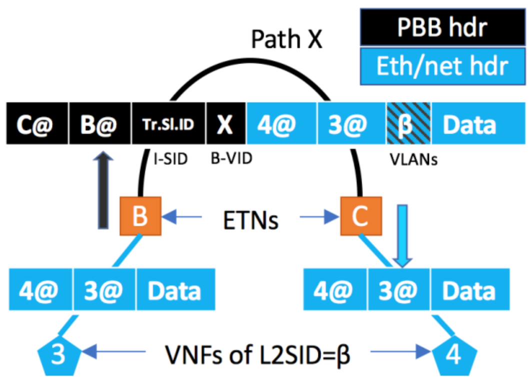


Figure 21: The slicing mechanism.

An example is depicted in Figure 21, where the two nodes B and C are edges of a shared transport network (Edge Transport Nodes – ETNs) and host two Virtual Network Functions (VNFs), 3 and 4, that are provided by a tenant. The tenant wants these two VNFs to be connected to the same virtual L2 segment, in an isolated environment with QoS guarantees. The packets going from one VNF to the other have as source and destination addresses the tenant-defined MAC addresses (3@ and 4@). The ETN datapath places the identifier of the virtual L2 segment (β in this case) in the VLAN-ID header field. When the packets are forwarded from one ETN to the other, they are encapsulated with an outer MAC (PBB) header that includes the source and destination addresses of the ETNs, an identifier for the path that will be used for their connection, and a slice identifier for potential slice-specific traffic engineering policies to be applicable. The following Figure 22 shows one of the ARP request packets that is created during our experimentation and is encapsulated into a PBB header. Obviously, this is an ARP Request that is produced by the VNF with MAC address 00:00:00:00:00:11 (the C-Src address in the 802.1ah header), which belongs to the L2SID 20 (the VLAN-ID is 0x16) and is hosted by the ETN with MAC address 00:03:1d:0d:40:a1 (the Src address in the Ethernet header).

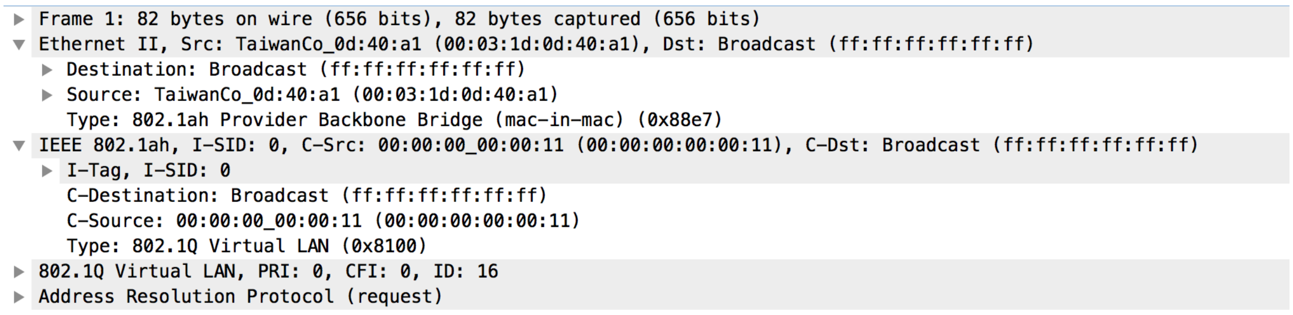

```
▶ Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
▼ Ethernet II, Src: TaiwanCo_0d:40:a1 (00:03:1d:0d:40:a1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: TaiwanCo_0d:40:a1 (00:03:1d:0d:40:a1)
    Type: 802.1ah Provider Backbone Bridge (mac-in-mac) (0x88e7)
▼ IEEE 802.1ah, I-SID: 0, C-Src: 00:00:00_00:00:11 (00:00:00:00:00:11), C-Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ I-Tag, I-SID: 0
    C-Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    C-Source: 00:00:00_00:00:11 (00:00:00:00:00:11)
    Type: 802.1Q Virtual LAN (0x8100)
▶ 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 16
▶ Address Resolution Protocol (request)
```

Figure 22: Wireshark capture of a PBB packet.

The NITOS OpenFlow network was exploited for the implementation of an area of the transport network, which includes 4 NITOS nodes behaving as ETNs. Each ETN uses the Mininet [12] environment to create virtual hosts (representing VNFs), which are connected with an OpenFlow virtual switch to a physical interface attached to the physical transport network. Figure 23 illustrates the transport network deployed at NITOS, the nodes used for the ETNs deployment, the VNFs hosted at each ETN and the tenant slice they belong to. At this experiment, the physical transport network consists only of the HP 3800 OpenFlow switch connecting the ETNs, since the evaluation of the wireless technologies (Sub6 and mmWave) for transport networking services is done in other experiments. The OpenFlow switch is programmed to check the B-VID field of the PBB header of the incoming packets, which is used for the path indication, and then forwards these packets to the appropriate ETN according to this field. Although in this simple case, the destination ETN field could be used for the same purpose, in general path-ID based forwarding is necessary for defining multiple paths between ETN pairs as well as for inter-area scenarios.

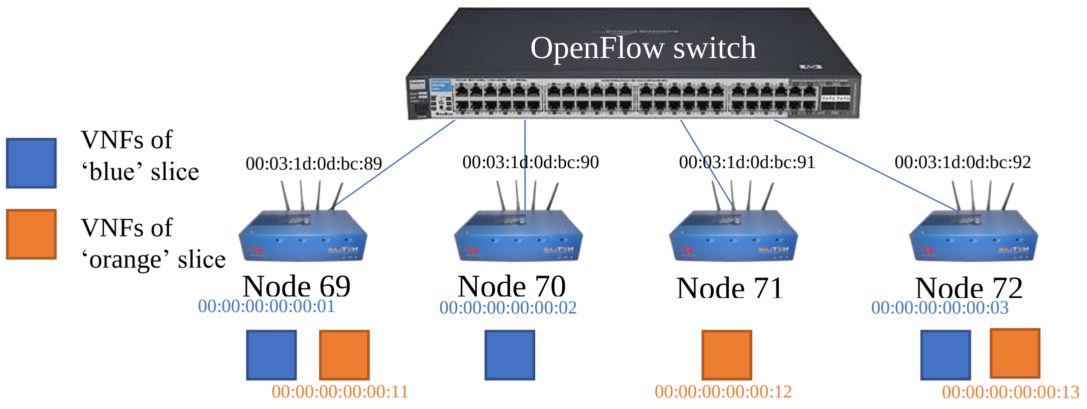


Figure 23: Network topology for the slicing experiment.

The Local Agents of the ETNs are executed at the same nodes, using the Ryu framework [11] for OpenFlow controllers. The datapath of each ETN is a userspace virtual OpenFlow switch at each node, which is implemented with use of the ofsoftswitch13 from CPqD [13], which supports the OpenFlow version 1.3. Note that we need OpenFlow v1.3 or newer in order for PBB to be supported. The Area Controller (AC) is located at the NITOS server that is connected with the NITOS nodes through a decoupled control network, not depicted in Figure 23. The AC uses a REST/HTTP API for both the southbound and northbound interfaces. The southbound interface (SBI) is responsible for controlling the Local Agents at the ETNs, while the northbound interface (NBI) facilitates the interaction between the AC and higher control layers. Note that in this experiment we invoke the NBI directly through either the command line utility curl or web tools such as Postman [14]. In an integrated environment implementing the entire control plane hierarchy, it would be the L1-Ctrlr which would make the request to the AC.

Using the NBI, it is possible to gradually build the slice topology, without having to know anything about the transport network internal operations. Specifically, VNFs of the slice are placed in ETNs, through simple HTTP PUT requests, where three parameters need to be specified: the ETN where the VNF is to be installed, the identifier of the virtual L2 segments it is attached to, and the VNF's MAC address. In case that VNFs attached to the same virtual L2 segment are attached to different ETNs, it is necessary to make sure that tunnels bridging these ETNs are already established. At the ETN level, this translates to specifying the tunnel ID to be inserted in the B-VID field of the PBB header, and is also implemented through a simple HTTP PUT request.

In this experiment, we use the Area Controller's NBI, listening at port 8080 of the testbed server, to build the topology of Figure 20. The VNFs of the 'blue' slice produced for the corresponding tenant are located at nodes 69, 70 and 72, while the VNFs of the 'orange' slice are located at nodes 69, 71 and 72. The three VNFs of each slice are assumed to belong to the same virtual L2 segment. Note that, although there is no overlap in the MAC addresses of the VNFs between the tenants in this example, this is just for clarity of presentation. The 'orange' slice could contain a VNF with MAC 00:00:00:00:00:01 for example, without any clash in the transport network, since these VNFs are attached to different virtual L2 segments.

4.1.2 Experiment results

As described, the first task is to assign tunnel IDs for connecting ETNs hosting VNFs of the same L2 segment. This is done with the NBI of the AC. Tunnels are considered unidirectional. As an example, for assigning tunnel ID 100 for a tunnel from node 69 to node 72, the request is as follows:

```
curl -X PUT -d '{"src_etn":"00:03:1d:0d:bc:89","dst_etn":"00:03:1d:0d:bc:92"}' http://nitlab3.inf.uth.gr:8080/tunnels/100
```

Note that the same tunnel (100) can be used to carry traffic for both tenants. This allows data plane scalability, as the tenant-specific details are hidden from the transport network until the packet reaches the destination ETN. The requests for setting up the rest of the required tunnels follow the same format.

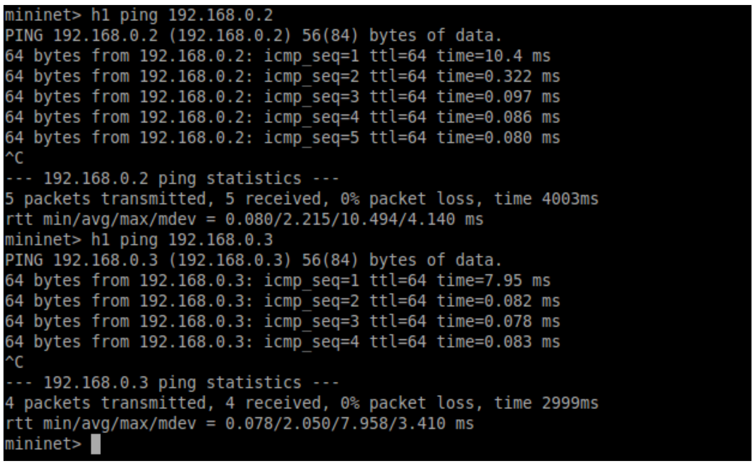
After setting up the tunnels, the VNFs of the 'blue' tenant are connected to the transport network, by issuing the following HTTP requests:

1. PUT {"etnid":"00:03:1d:0d:bc:**89**"} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**20**/MAC_addresses/00:00:00:00:00:**01**
2. PUT {"etnid":"00:03:1d:0d:bc:**90**"} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**20**/MAC_addresses/00:00:00:00:00:**02**

3. PUT {“etnid”:“00:03:1d:0d:bc:**92**”} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**20**/MAC_addresses/00:00:00:00:00:**03**

In the URI of each of the above requests, the corresponding VNF is identified using its MAC address and the ID of the virtual L2 segment where this VNF is attached (L2SID). Moreover, in the data part, the ETN hosting this VNF is identified by its own MAC address, which is the MAC address of the Ethernet interface of the corresponding NITOS node. This latter MAC address is the source MAC address of the PBB packets coming out from the ETN, as well as the destination MAC address of the packets destined to this ETN.

After issuing the above HTTP requests, the VNFs are able to ping each other. We can see this in Figure 24, where VNF “01” successfully pings the other VNFs “02” and “03” of the “blue” slice. Keep in mind that the three nodes have hostnames h1, h2 and h3 and IP addresses 192.168.0.1, 192.168.0.2 and 192.168.0.3 respectively.



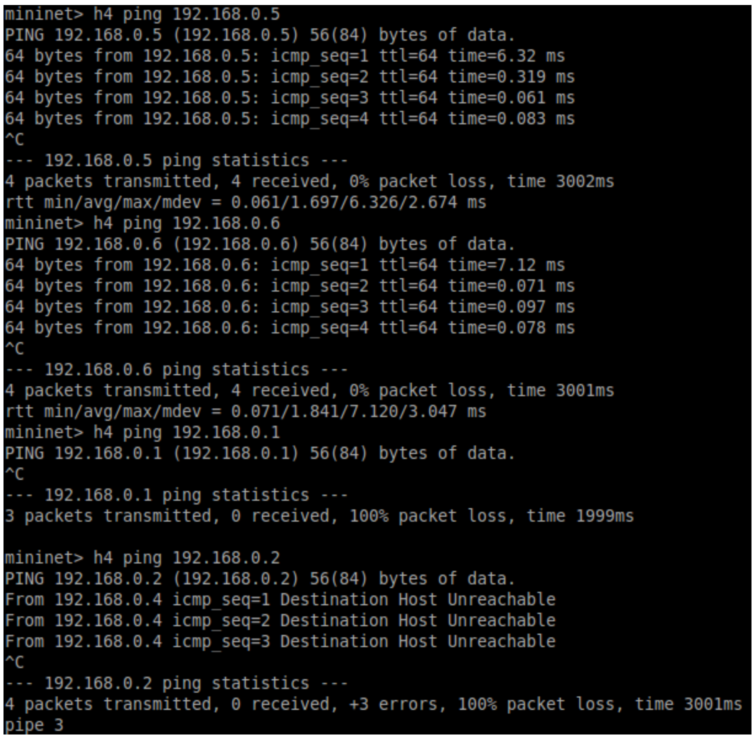
```
mininet> h1 ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=10.4 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.322 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.086 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.080 ms
^C
--- 192.168.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.080/2.215/10.494/4.140 ms
mininet> h1 ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=7.95 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 192.168.0.3: icmp_seq=3 ttl=64 time=0.078 ms
64 bytes from 192.168.0.3: icmp_seq=4 ttl=64 time=0.083 ms
^C
--- 192.168.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.078/2.050/7.958/3.410 ms
mininet> █
```

Figure 24: VNFs of the same slice are connected.

Then, the VNFs of the “orange” slice are created, which are attached to the L2 segment with ID 30:

1. PUT {“etnid”:“00:03:1d:0d:bc:**89**”} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**30**/MAC_addresses/00:00:00:00:00:**11**
2. PUT {“etnid”:“00:03:1d:0d:bc:**91**”} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**30**/MAC_addresses/00:00:00:00:00:**12**
3. PUT {“etnid”:“00:03:1d:0d:bc:**92**”} http://nitlab3.inf.uth.gr:8080/vifaces/l2sids/**30**/MAC_addresses/00:00:00:00:00:**13**

After their execution, the three VNFs are able to ping each other, while they are not able to see the VNFs of the other slice, as it is depicted in the Figure 25. The new VNFs have hostnames h3, h4 and h5 and IP addresses 192.168.0.3, 192.168.0.4 and 192.168.0.5 respectively.



```
mininet> h4 ping 192.168.0.5
PING 192.168.0.5 (192.168.0.5) 56(84) bytes of data.
64 bytes from 192.168.0.5: icmp_seq=1 ttl=64 time=6.32 ms
64 bytes from 192.168.0.5: icmp_seq=2 ttl=64 time=0.319 ms
64 bytes from 192.168.0.5: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 192.168.0.5: icmp_seq=4 ttl=64 time=0.083 ms
^C
--- 192.168.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.061/1.697/6.326/2.674 ms
mininet> h4 ping 192.168.0.6
PING 192.168.0.6 (192.168.0.6) 56(84) bytes of data.
64 bytes from 192.168.0.6: icmp_seq=1 ttl=64 time=7.12 ms
64 bytes from 192.168.0.6: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 192.168.0.6: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 192.168.0.6: icmp_seq=4 ttl=64 time=0.078 ms
^C
--- 192.168.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.071/1.841/7.120/3.047 ms
mininet> h4 ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms

mininet> h4 ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
From 192.168.0.4 icmp_seq=1 Destination Host Unreachable
From 192.168.0.4 icmp_seq=2 Destination Host Unreachable
From 192.168.0.4 icmp_seq=3 Destination Host Unreachable
^C
--- 192.168.0.2 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3001ms
pipe 3
```

Figure 25: VNFs from different slices are not able to ping each other.

4.2 TE Sub-6 demo

In this section, we perform a set of experiments to demonstrate some of the control plane mechanisms proposed in deliverable D3.2 [REF]. In particular, we demonstrate Traffic Engineering (TE) mechanisms for Sub6 GHz wireless transport areas, including the main and backup path allocation policies (chapter 3 in D3.2), the Fast Local Link ReRoute (FLRR) recovery agent (chapter 3 in D3.2), and the Mobile Network to Backhaul Interface (MN2BH) that instantiates transport flows based on the GTP identifier used in each LTE bearer (chapter 5 in D3.2).

4.2.1 Setup, test definition, metrics

To demonstrate TE for the Sub6 wireless backhaul, the RF-isolated indoor testbed in NITOS is equipped with an eNB, an EPC, and a plenitude of wireless nodes that can be used as wireless backhaul nodes or LTE-enabled UEs. The following Figure 26 shows some of the LTE equipment that is used in NITOS.

Figure 26: LTE femtocell and UE in NITOS.

We configure 8 of the testbed nodes to form a wireless backhaul, representing a Sub6 GHz control plane area according to the 5G-XHaul control plane architecture. Each NITOS node is equipped with up to 2 wireless interfaces, thus supporting simultaneous communications on up to 2 different channels. It is worth highlighting that in this experiment NITOS nodes instantiate a Transport Node (TN) of the 5G-XHaul architecture, and not an ETN as in the previous experiment.

The topology used for the experiments and the channels on which the nodes communicate are shown in Figure 27. The wireless backhaul includes an entry node (s0) that connects the eNB with the backhaul and two gateway nodes (s2, s7) that connect the backhaul to the EPC.

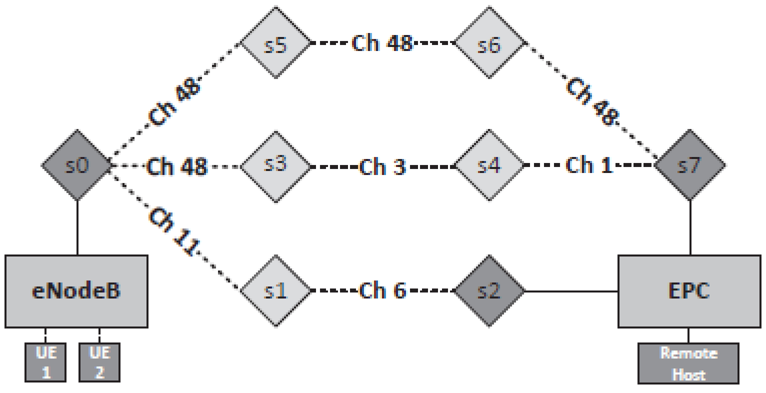


Figure 27: Network topology in the NITOS testbed.

The SDN controller in charge of managing the data plane of the backhaul runs on a remote virtual machine connected to the backhaul via a VPN instantiated on the node s7. The SDN controller is hosted by i2CAT in Barcelona. As destination or origin for the LTE traffic interchanged during the experiments, we set up two nodes as LTE UEs. A node connected to the wired backbone behind the EPC serves as destination or origin of data for communications with the UEs.

To instantiate paths for the transport tunnels, the sequential policy, described in section 3.1.2 of deliverable D3.2 [10], is applied in the controller, with a channel occupancy threshold equal to 0.6. Above this threshold, the channel is considered to be congested and an alternative path allocation is performed.

As initial step of the experiment, we set up the LTE connections of the two UEs. In this process, the control traffic between the eNB and the EPC traverses our backhaul over a pre-provisioned in-band control path. As soon as the UEs establish their connection, each UE is assigned a unique GTP TEID for its corresponding upstream and downstream flows.

With the knowledge of the GTP TEIDs for each of the UEs, the 5G-XHaul Area Controller (AC) assigns main and backup paths for the correspondent end-to-end flows. These paths start at the entry node (s0) and go to any of the gateway nodes (s2, s7) for the upstream traffic and the opposite direction for the downstream traffic. Following the sequential policy, the controller assigns the main path for both upstream flows to the lower branch (s0-s1-s2) and the backup paths over the upper branch (s0-s5-s6-s7). The downstream flows are assigned to use the upper branch as main and the middle branch (s7-s4-s3-s0) as backup paths, respectively. To initially allocate the paths, a virtual load of 1 Mbps for each end-to-end flow is assumed.

4.2.2 Experiment results

We evaluate the previous policies through two experiments, based respectively on UDP and TCP traffic. In particular, these experiments will evaluate how the control plane reacts when a link breaks, and the data flows carried over the backhaul have to be relocated. Next, we present the results of these experiments.

UDP based experiment

As performance metrics, we measure how the overall throughput evolves before, during, and after the link break. Based on the initial setup described earlier, the first experiment consists in launching two UDP-based *iperf* data streams, one from each of the UEs, towards the server located behind the EPC. The first flow (f1) is set to 1 Mbps, whereas the second flow (f2) is set to 3 Mbps. The sum of the weight of the flows is intended to lie below the actual capacity of the wireless links, being limited on the sender side by the *iperf* application. Both flows are launched at second 5 of the experiment. In Figure 28, the aggregate traffic sent over the radio by s0, the entry node of the upstream flows, is visualized. Initially, a total of 4 Mbps is maintained over the lower branch (s0-s1-s2). The previous flow data rates are selected based on the capabilities of the radio technologies available in the NITOS testbed, which are also affected by severe interference due to having all nodes closely packed to each other. Notice though that the focus of these experiments is on the control plane mechanisms, not the data plane performance.

After 90 seconds, we interrupt the link between s1 and s2 by manually shutting down the wireless transceiver on s2. The FLRR agent detects this break at node s1, which triggers a change of the rules on s1: Following the FLRR protocol, described in D3.2, the flows pointing towards the gateway s2 are modified to point back towards s0. Any further packets received by s1 towards the gateway node s2 are subsequently returned to s0. As this rule takes effect, the regressing packets start matching the switch rule at s0 redirecting the packets of both upstream flows over s5 towards s7. This behaviour is detected by the FLRR agent in s0 which, in return, modifies the default rule for the affected upstream flows to no longer point towards s1, but to s5 instead.

The change can be recognized in Figure 28 as the measured load switches from the link s0-s1 to the link s0-s5. The whole process of detecting the link break and redirecting the traffic over the backup path takes less than 2 s. During this process, we observe that the throughput during one *iperf* report interval of 1 s drops by a small amount of 200 kbps, after which the full throughput of 4 Mbps is reestablished. Further, in this process we only measure 9 packets that enter the 'dead' lower branch after the link break and before the switch rule at s0 redirects the packets over the backup path, and no packet loss, thus proving the minimal disruption introduced by the FLRR agent.

At this point of the experiment, f1 and f2 both have switched from the broken main path on the lower branch to the backup path on the upper branch. The links on the backup path are on the same channel and, as a result of both flows using the same backup path, the channel occupancy increases drastically up to 71%. This exceeds the maximum allowed channel load threshold of 60% configured in the 5G-XHaul Area Controller and triggers a new path computation. At second 130, the controller decides to reallocate f1 over the middle branch of the topology, separating the two end-to-end flows. As a result, the peak channel load observed over the upper branch is reduced to 54%, with only one active flow. Notice that f1 is not relocated immediately, because a 15 second interval is used to collect OF statistics and the controller applies filtering to the received statistics to avoid ping-pong effects. Given that FLRR quickly resolves a link break, a larger polling interval is considered appropriate to reduce signaling overhead. However, these values can be configured more aggressively if a faster overall optimization of the network is required.

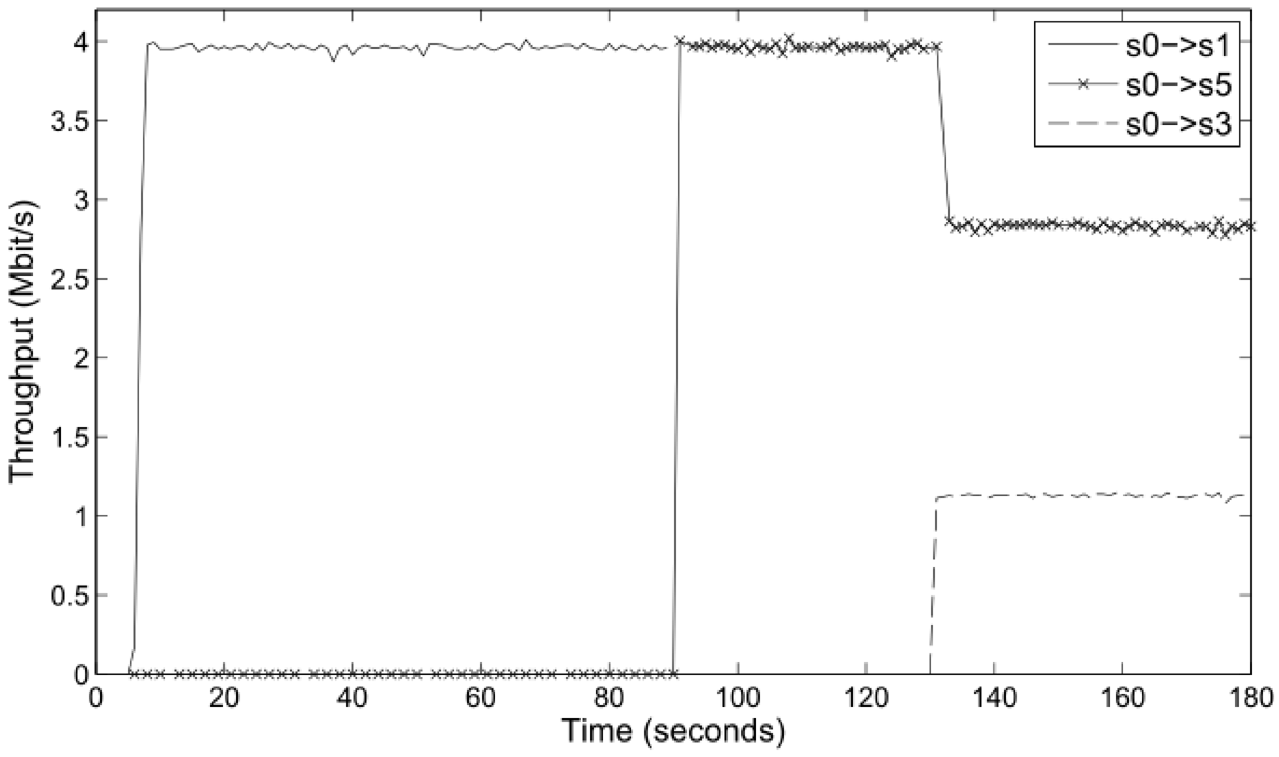


Figure 28: UDP experiment

TCP based experiment

The second experiment is performed with the same base setup, but using a single TCP flow launched from the first UE towards the server behind the EPC. In this case, *iperf* does not limit the throughput, which depends on the rate adaptation mechanisms of TCP and on the variable bandwidth in the LTE link and the wireless backhaul links. Following the sequential policy, the controller allocates this flow on the lower branch and assigns the backup path on the upper branch. The transmission lasts for 41 seconds, after which the link between s1 and s2 is manually broken by shutting down the wireless transceiver of s2. As shown in Figure 29, which plots the throughput measured at the node s0, the FLRR agent reacts immediately by redirecting the flow over the backup path (from s0-s1 to s0-s5). The variations of throughput measured over the course of the experiment are caused by fluctuations in the LTE link and in the wireless backhaul links[1]. The fact that each of the three hops on the backup path now is on the same radio channel reduces the achievable end-to-end throughput as the wireless transceivers of s0, s5, and s6 are competing for the channel access. In addition, since the channel load now exceeds the maximum threshold of 60%, the Area Controller reallocates the flow to the middle branch, via s3, on second 59. The reallocation from the upper to the middle path happens almost immediately, a reduction of the throughput is only observed during second 59.

Once reallocated, the average throughput of the end-to-end flow increases noticeably, as the three links of the middle branch lie on separate channels.

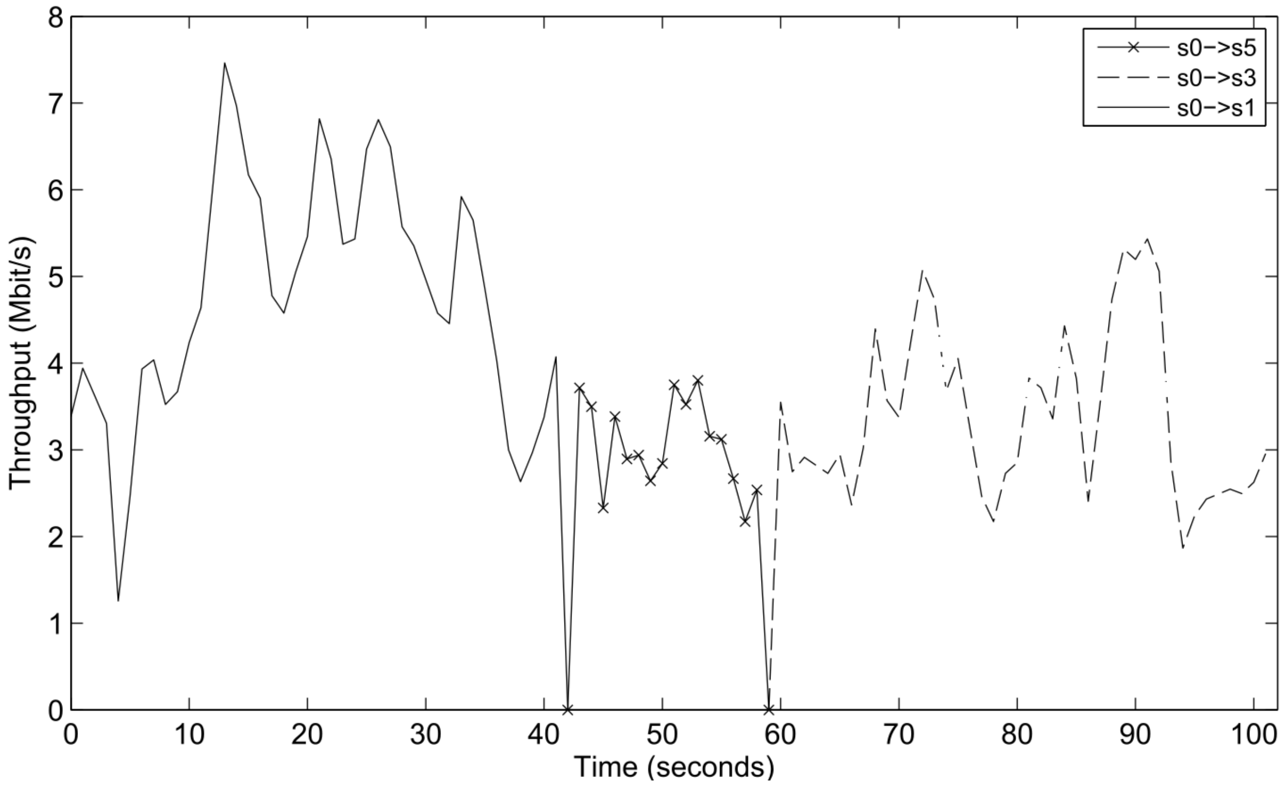


Figure 29: TCP Experiment.

[1] Note that in the UDP experiment, the links were not saturated, since the traffic was limited on the application side and thus not showing those fluctuations.

5 Joint Sub-6 and mmWave experiments

This demo integrates the contributions and the technologies evaluated in the previous experiments, including also some new ones, such as the emerging and well-promising wireless technology of mmWave. The purpose of this demo is to validate the efficient operation of a transport network area that supports the provision of isolated virtual networks and exploits appropriately the benefits of the Sub6 and mmWave wireless technologies. This demo is the seminal work that will pave the way for the final demo in Bristol, where we will also integrate the optical technologies.

5.1 Deployment setup, controllers, interfaces

This experiment is aiming at evaluating the potential and the challenges of a network, which jointly leverages on both wireless technologies, Sub6 and mmWave. For the purpose of deploying this network, both NITOS and mmWave nodes have been exploited to support WiFi and mmWave wireless connectivity respectively. The network used in this experiment is depicted in Figure 30. This network is a transport network that could be booked and used by mobile broadband operators for placing their core elements, organised as VNFs, and be connected with their access networks. In this experiment, there are three edge-points, the ETNs hosting the customers VNFs, which are three NITOS nodes. The three ETNs are interconnected through four other NITOS nodes and two mmWave nodes, operating as TNs that exploit either the Sub6 (Wi-Fi) or the mmWave technologies.

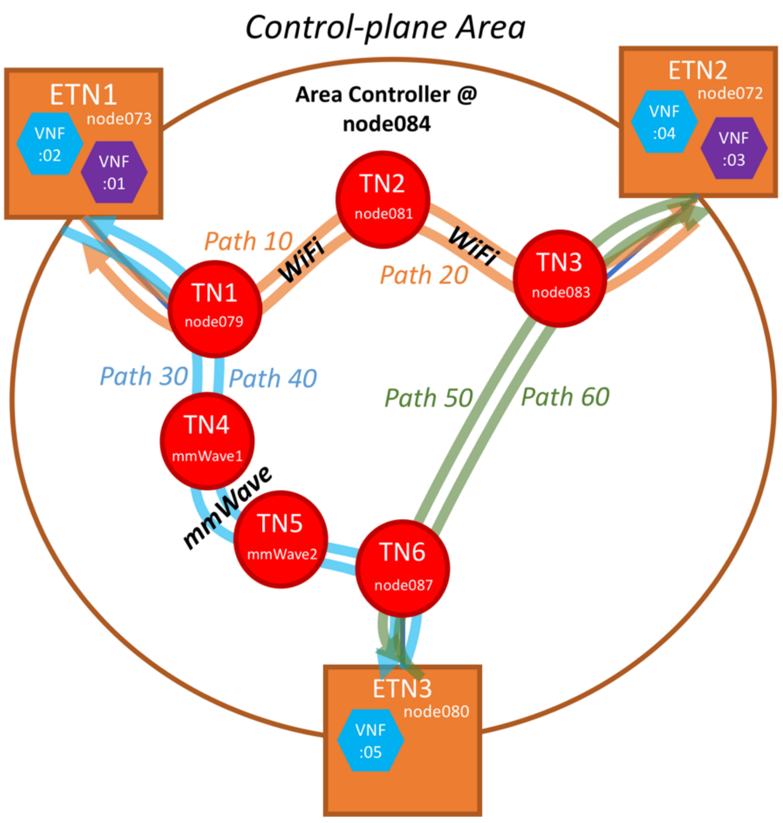


Figure 30: Experimentation topology of joint sub6 and mmWave experiment.

Each node (ETN/TN) operates as virtual OpenFlow switch that controls its interfaces and its ingress/egress traffic through an OpenFlow controller. For each ETN, the corresponding OpenFlow controller is collocated with the virtual OpenFlow switch at the ETN, while all ETN OpenFlow controllers are connected with the ETN-aggregate controller that is located in an extra NITOS node. This controller is the same as the one used in the experiment described in Section 4.2. The ETN-aggregate controller and the TN controller are also interconnected to jointly operate as the Area controller, who is responsible for configuring the whole area of the transport network.

Specifically, the TN controller exposes a REST API for creating tunnels between endpoints. This controller was hosted at i2CAT premises during the experiment. The API is as in the following example:

POST http://<host_of_tn_controller>/restconf/operations/fivegxhaul:create-tunnel
Data: {“input”:{“tunnel-id”:100, “source-node”:”ETN1”, “destination-node”:”ETN2”}}

5.2 Scenario description

We start our experimentation with two UDP streams, which are initiated from VNF :02 to VNF :04 and from VNF :02 to VNF :05. The two streams correspond to video streams of UHD and HD quality, thus they require throughput almost 5Mbps and 30Mbps respectively. The tenant who owns the VNFs (the ‘blue’ tenant) is flexible on the location of the VNFs, enabling their placement in the ETNs that are connected with the less loaded tunnels. Thus, the orchestrator decides to place VNF :02 at ETN1, VNF :04 at ETN2 and VNF :05 at ETN3, since ETN1 is connected with ETN2 and ETN3 through tunnels that have capacities almost 10Mbps and 500 Mbps respectively.

Then, a new tenant (the ‘purple’ tenant) brings its VNFs, VNF :01 and VNF :03, and requires these VNFs to be placed at ETN1 and ETN2 respectively. The purple tenant also requires the two VNFs to be able to stream each other with throughput of 8Mbps[2]. This new event forces the orchestrator to relocate some of the VNFs of the blue tenant, in order to be able to satisfy both the blue and purple tenants. As it is obvious from the network topology, the only solution is to live migrate the VNF :04 of the blue tenant from ETN2 to ETN3. Live migration means that VNF will be transferred from the one ETN to the other, with its state and all network connections. However, some packets that are being forwarded through the network during the migration, will be lost. The reason is that they are forwarded destined to ETN1 that was first hosting the migrated VNF, while the VNF is being transferred to ETN2 with a ‘frozen’ network state.

The following figure depicts the scenario:

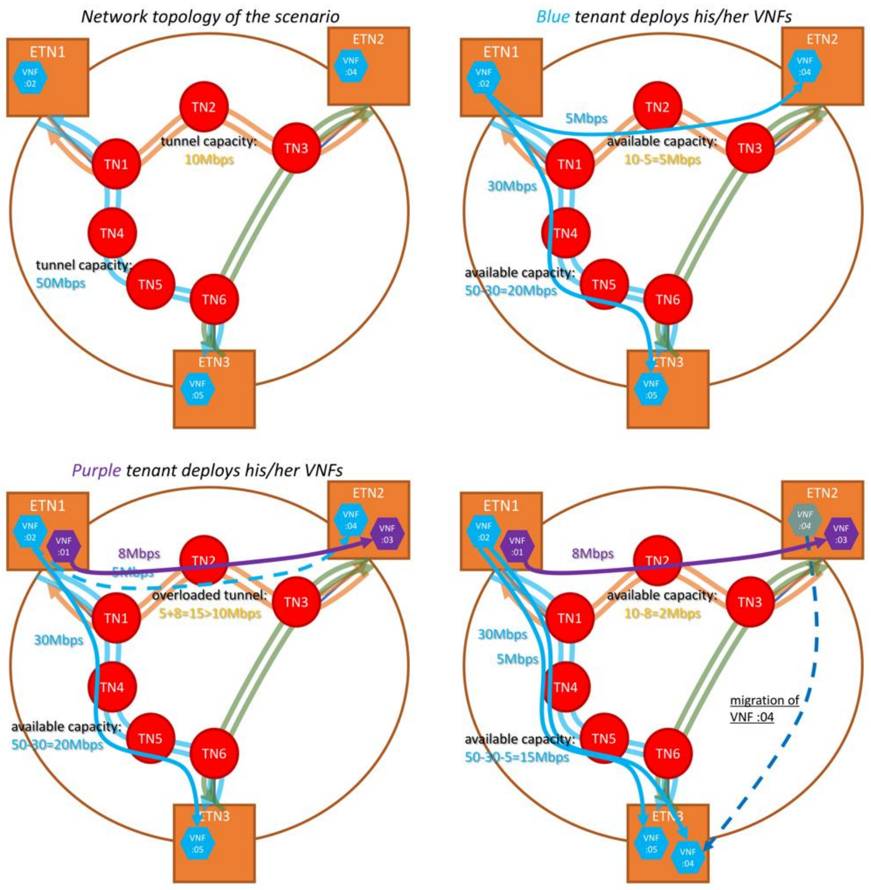

Network topology of the scenario
ETN1
VNF :02
ETN2
VNF :04
TN2
tunnel capacity: 10Mbps
TN3
TN1
TN4
tunnel capacity: 50Mbps
TN5
TN6
ETN3
VNF :05
Blue tenant deploys his/her VNFs
5Mbps
available capacity: 10-5=5Mbps
30Mbps
available capacity: 50-30=20Mbps
Purple tenant deploys his/her VNFs
VNF :01
VNF :03
8Mbps
5Mbps
overloaded tunnel: 5+8=15>10Mbps
30Mbps
available capacity: 50-30=20Mbps
8Mbps
available capacity: 10-8=2Mbps
30Mbps
5Mbps
migration of VNF :04
available capacity: 50-30-5=15Mbps

Figure 31: Scenario of the joint sub6 and mmWave experiment.

[2] Notice that this traffic is scaled down to adapt to the capabilities of the testbed, since the main purpose of these experiments is to validate control plane functionalities. The interested reader is referred to D5.2 for an evaluation of the data-plane capabilities of the technologies developed in 5G-XHaul.

5.3 Experiment results

The following Figure 32 shows the throughput of the UDP streams existing in the network. The throughput from VNF :02 to VNF :05 is continuously 30Mbps. The throughput from :02 to :04 is 5Mbps, except for a small interval, that is almost 10msec and the throughput is slightly reduced to 3Mbps. This interval is when the new stream from VNF :01 to VNF :03 is initiated.

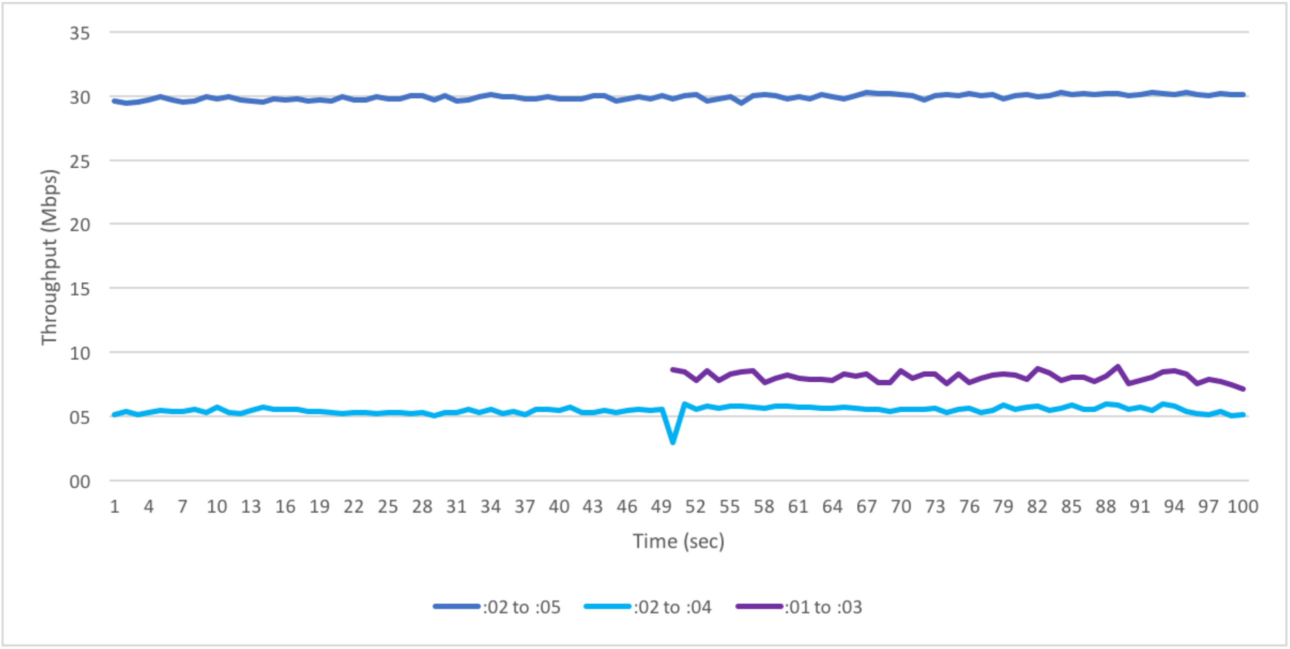


Figure 32: Throughput results.

6 Summary and Conclusions

This deliverable provides the necessary information for understanding the experimentation done in NITOS and how the produced results validate the functionality and efficient operation of the 5G-XHaul contributions on the SDN control plane. The developed SDN functionality has been tested over heterogeneous RAN technologies, such as Sub-6 and mmWave and it will be ported in the final project demonstrator in the BIO testbed.

7 References

[1] Network Implementation Testbed using Open Source software (NITOS), http://nitos.inf.uth.gr/.

[2] HP 3800 OpenFlow Switch, https://www.opennetworking.org/sdn-openflow-products/639-hp-3800-switch-series.

[3] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon and Ivan Sescar, "OMF: a control and management framework for networking testbeds", In ACM SIGOPS Operating Systems Review, vol. 43, no. 4, pp. 54-59.

[4] Rederated Resource Control Protocol (FRCP), https://github.com/mytestbed/specification/blob/master/FRCP.md.

[5] Anadiotis, et al., "A new slicing scheme for efficient use of wireless testbeds", 2009.

[6] Open-vSwitch, http://openvswitch.org/.

[7] Deliverable D4.11, "Wireless backhauling using Sub-6 GHz systems", 5G-XHaul Project.

[8] OpenDaylight OpenFlow controller, https://www.opendaylight.org/

[9] Deliverable D3.1, "Analysis of state of the art on scalable control plane design and techniques for user mobility awareness. Definition of 5G-XHaul control plane requirements", 5G-XHaul Project.

[10] Deliverable D3.2, "Design and evaluation of scalable control plane, and of mobility aware capabilities and spatiotemporal demand prediction models", 5G-XHaul Project.

[11] Ryu OpenFlow controller, https://osrg.github.io/ryu/

[12] Mininet, http://mininet.org/

[13] CPqD/ofsoftswitch13 (OpenFlow 1.3 compatible user-space software switch implementation), https://github.com/CPqD/ofsoftswitch13

[14] Postman, https://www.getpostman.com/

8 Acronyms

Acronym	Description
5G	Fifth Generation Networks
AC	Area Controller
API	Application Program Interface
BH	Backhaul
BIO	Bristol is Open
CN	Core Network
C-RAN	Cloud Radio Access Network (aka Cloud-RAN)
DC	Data Centre
DL	Downlink
e2e	end-to-end
ETN	Edge Transport Node
FH	Fronthaul
FIB	Forwarding Information Base
IATN	Inter-Area Transport Node
KPI	Key Performance Indicator
L2SID	Layer 2 Segment ID
LA	Local Agent
LoS	Line-of-Sight
LTE	Long Term Evolution
MAC	Medium Access Control
mmWave	Millimetre Wave
NBI	North-Bound Interface
NFV	Network Function Virtualisation
NLoS	Non-Line-of-Sight
ODL	OpenDayLight
ONF	Open Networking Foundation
OS	Operating System
P2P	Point-to-Point
PBB	Provider Backbone Bridge
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational state transfer
RSSI	Received Signal Strength Indicator
SDN	Software Defined Networking

SLA	Service Level Agreement
TAF	Transport Adaptation Function
TC	Transport Class
TCP	Transmission Control Protocol
TE	Traffic Engineering
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink
vDP	Virtual datapath
VLAN	Virtual Local Area Network
VM	Virtual Machine
VN	Virtual Network
VNF	Virtual Network Function
WP	Work Package