



Wireless Software and Hardware platforms for Flexible and Unified radio and network control

Project Deliverable D6.2

First implementation of the Portable Testbed

Contractual date of delivery:	31-12-2015
Actual date of delivery:	23-12-2015
Beneficiaries:	TUB, IMINDS, NCENTRIC
Lead beneficiary:	TUB
Authors:	Piotr Gawlowicz (TUB), Mikolaj Chwalisz (TUB), Anatolij Zubow (TUB), Vincent Sercu (IMINDS), Pieter Becue (IMINDS), Ingrid Moerman (IMINDS), Spilios Giannoulis (IMINDS), Robin Leblon (NCENTRIC)
Reviewers:	Spilios Giannoulis (iMinds), Ingrid Moerman (iMinds)
Work package:	WP6 – Testbed on the move
Estimated person months:	13
Nature:	R
Dissemination level:	PU
Version:	4.4

Abstract:

This public document gives a detailed description of the capabilities and operation of the Portable Testbed in year 1. This deliverable also provides guidelines for packaging, transport, insurance and initial setup of the hardware. This deliverable includes the improvements and extensions to be implemented in year 2.

Keywords:

Portable Testbed, Packaging, Transportation, Wireless Network Control, Testbed Management

Executive Summary

This Year 1 deliverable reports on the status of Portable Testbed.

The **overall structure of Portable Testbed** (PT). has been defined, whereby a clear separation was made between the System-Under-Test (SUT) Network, which is the wireless network the experimenter wants to validate in his experiment, and the Backbone Network (BN), which is the network of nodes that control the Device-Under-Test (DUT) nodes in the SUT. Each entity that is part of Portable Testbed is characterized and its role is identified. Moreover, we have motivated the hardware that was chosen for each particular node, powering solutions, as well as extensions that we developed to make them more compact and easy to transport. We also have selected the software that is used for operating the Portable Testbed.

The Portable Testbed currently supports the following **functionality**:

- *discovery and integrity monitoring mechanism*: this mechanism is crucial during the deployment and operation of the Portable Testbed
- *interface between Testbed Management Server (TMS) and Backbone Network Controller (BNC)*: this new interface is implemented and used for discovery of available DUT nodes, passing list of channels used in the SUT network. This interface is also used for the configuration of Queueing Disciplines for experiment and SUT network control traffic.
- *Prioritization of control flows*: in current version, the Backbone Network Controller supports prioritization of control flows in Network and Medium Access Control (MAC) layers using the Traffic-Control subsystem of the Linux operating system and exploiting the EDCA extension of IEEE 802.11 (Enhanced Distributed Channel Access).
- *WiSHFUL Unified Programming Interfaces (UPIs)*: The BNC uses the Unified Programming Interfaces defined within the WiSHFUL project for the configuration of the nodes in the Backbone Network (BN).
- *Channel Policy Agreement (CPA)*: Upon reception of list of channels used in the experiment, the BNC performs the so-called Channel Policy Agreement in order to ensure that the BN network works on orthogonal frequencies relative to the SUT network.

For the **packaging and transportation** of the hardware constituting the Portable Testbed flight cases are used. Due to heavy weight, we switched from single flight case solution to multiple flight cases. Separated cases are more practical for transportation.

The following **improvements and extensions**, to be implemented in the next release by the end of Year 2, have been identified:

- *automatic DUT-BN node mapping discovery*: this mechanism will facilitate deployment and reduce the number of maintenance operations of Portable Testbed.
- *DUT power-supply control*: each BN node will be able to control the power supply of the attached DUT node. This functionality will be useful for releasing resources when the experiment is finished.
- *improved Channel Policy Agreement*: the goal is to prevent an experimenter from using different channels than the ones he reserved and as such avoid interference with the BN.
- *channel access optimisation*: an additional mechanism will be implemented to avoid congestion in the mesh Backbone Network.

List of Acronyms and Abbreviations

AC	Access Category
AMQP	Advanced Message Queuing Protocol
ARM	Advanced RISC Machine, a family of processors
ATA	Air Transport Association
BN	Backbone
BNC	Backbone Network Controller
COTS	Commercial off-the-shelf
CPA	Channel Policy Agreement
CSMA	Carrier Sense Multiple Access
DCF	Distributed Coordination Function
DDR	Double Data Rate
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DUT	Device Under Test
EC	Experiment Controller
EDCA	Enhanced Distributed Channel Access
EMF	Experiment Management Framework
EMS	Experiment Management Server
FIFO	First In, First Out
GPIO	General Purpose Input/Output
HP	Hewlett-Packard
L2	Layer 2
L2TP	Layer 2 Tunneling Protocol
LAN	Local Area Network
MAC	Media Access Control
NET	Network
NFS	Network File System
NUC	Next Unit of Computing, a family of Intel Mini-PCs
OLSR	Optimized Link State Routing
OMF	A Control and Management Framework for Networking Testbeds
OML	OMF Measurement Library
OPS	Operations Server
OS	Operating System
PCB	Printed Circuit Board

PCI	Peripheral Component Interconnect
pfifo	Packet FIFO Queuing Discipline
PLA	Polylactic acid
PnP	Plug and Play
PoE	Power over Ethernet
PRIO	Priority
PT	Portable Testbed
PUR	Polyurethane
PXE	Preboot Execution Environment
QDisc	Queuing Discipline
RC	Resource Controller
SFA	Slice-based Facility Architecture
SMA	SubMiniature version A connector
SoC	System on Chip
SUT	System Under Test
SUT-NC	SUT Network Controller
TMF	Testbed Management Framework
TMS	Testbed Management Server
TOS	Type-of-Service
UPI	Unified Programming Interface
USB	Universal Serial Bus
UTP	Unshielded twisted pair
VLAN	Virtual LAN
VM	Virtual Machine
Wi-Fi	Wireless Fidelity

Table of contents

1	Introduction	7
2	First Portable Testbed release	7
2.1	Portable Testbed overview	7
2.2	Testbed and Experiment Management	9
2.2.1	Server configuration	9
2.2.2	Network configuration	9
2.3	SUT Network	9
2.4	SUT WiSHFUL controller	9
2.5	Devices Under Test (DUT's)	10
2.5.1	Physical node configuration	10
2.5.2	Network and power configuration	12
2.6	Portable testbed	13
2.6.1	Backbone Network	13
2.6.2	Backbone WiSHFUL controller	14
2.6.3	TMS-BNC Interface	14
2.6.4	BN Node	15
3	Implemented capabilities	16
3.1	BN node discovery	16
3.2	Integrity monitoring	16
3.3	Interference management	16
3.4	QoS support	17
4	Packaging and transport	17
4.1	Flight-case	18
4.1.1	Single box solution	18
4.1.2	Separate briefcase solution	18
4.1.3	Future battery powered solution	20
5	Future improvements and extensions	21
5.1	Automatic DUT node to BN node mapping discovery	21
5.2	DUT Power-supply control	21
5.3	Channel Policy Agreement improvement	21
5.4	Channel Access optimization	22
6	Conclusion	23

7	References	24
----------	-------------------------	-----------

1 Introduction

This public document provides a detailed description of current operational status and capabilities of Portable Testbed. We present the description of general hardware setup and functionalities supported by each testbed entity. Moreover, we provide guidelines for packaging and transport. Finally, we give a description of the improvements and extensions that will be implemented in Year 2.

This document is structured as follows. In Section 2, we present the overall structure of Portable Testbed and further characterize the different entities of the Portable Testbed. Section 2 motivates the hardware and software that is selected for the Portable Testbed. In Section 3, we present functionalities that are supported in the Year 1 release of the Portable Testbed. In Section 4, we describe how the hardware of the Portable Testbed can be packaged for transportation. Finally, in Section 5, a description of the improvements and extensions that will be implemented in year 2 is given.

2 First Portable Testbed release

This section gives the detailed description of current implementation status of Portable Testbed. We give an overview of hardware, software and capabilities of each entity of PT.

2.1 Portable Testbed overview

The Portable Testbed consists of a set of Backbone Network (BN) nodes and one controller. Each backbone node is expected to connect one Device-Under-Test (DUT) node, as shown in Figure 1.

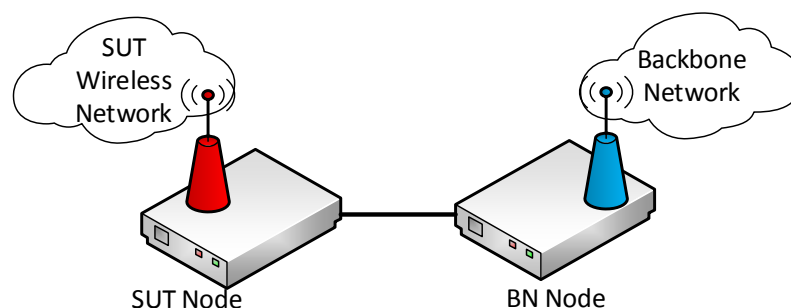


Figure 1: Basic building block of Portable Testbed

As the Portable Testbed introduces an additional network to an experiment, it is implemented in such a way, that an experimenter is not overwhelmed with additional and complicated configuration. In D6.1, it was decided that Portable Testbed has to follow the “Plug and Play” approach and an experimenter should be able to use the same Testbed and Experiment Management Frameworks as usual. It has to be noted, that an experimenter does not have the possibility to directly control the behaviour of BN network, but he is able to change the channel that Portable Testbed uses, through TMS. Moreover, we provide logical L2 networks to interconnect DUT nodes in order to make them unaware whether they are inter-connected through Portable Testbed. This approach also reduces the required configuration because an experimenter does not have to configure any routing in his DUT nodes.

The logical structure of the Portable Testbed is presented in Figure 2, while Figure 3 shows an example of its physical structure.

In the following subsections, we provide a detailed description of each entity that is part of the Portable Testbed.

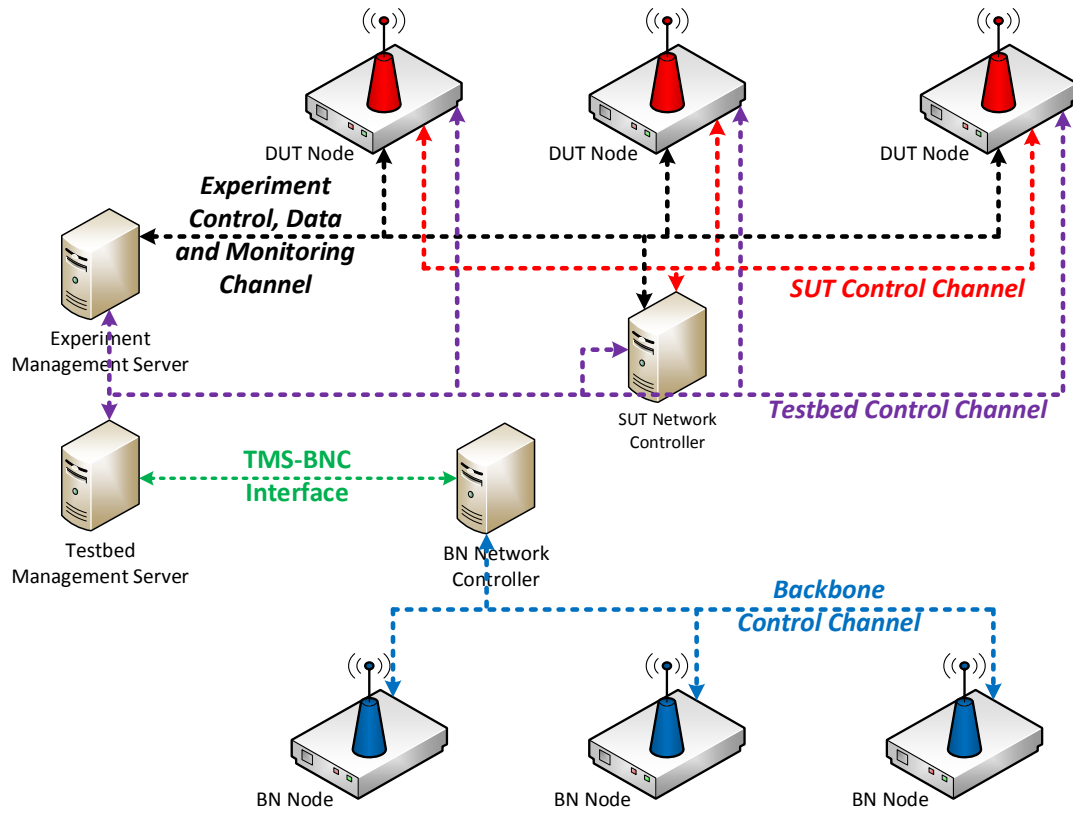


Figure 2: An overview of the logical structure of the Portable Testbed

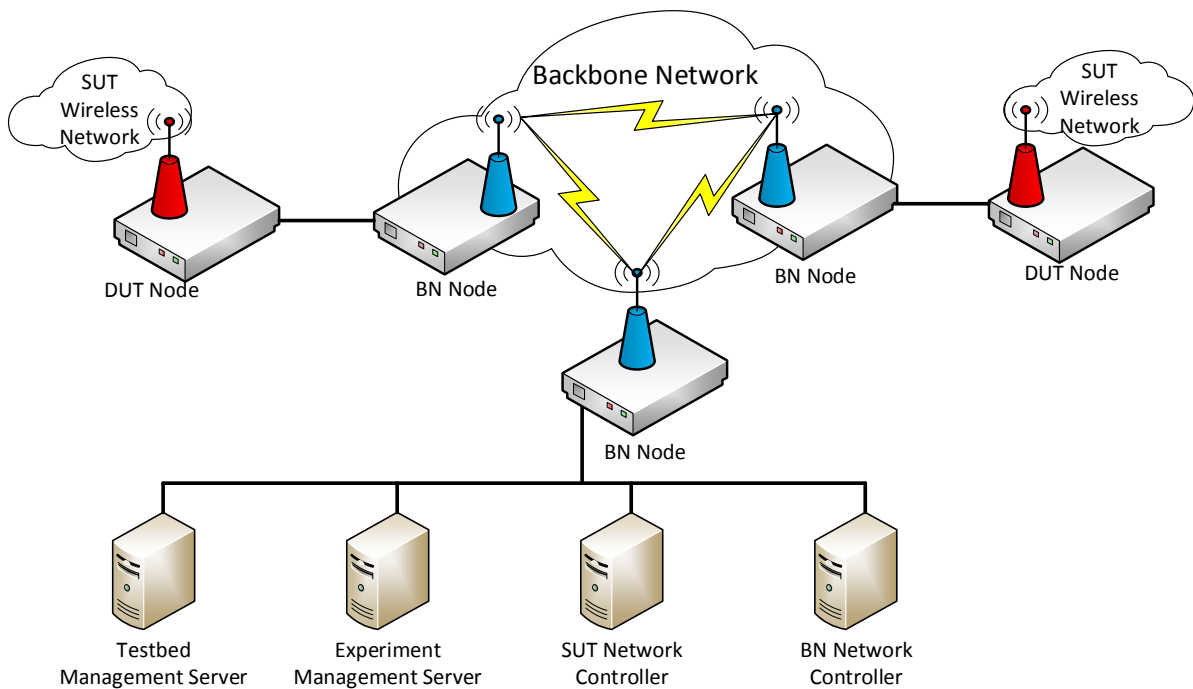


Figure 3: An example of physical structure of the Portable Testbed

2.2 Testbed and Experiment Management

The management of the portable testbed is done by a single node. It is mounted in the same flight case as the switch (see section 4.1.2). The following sections describe the configuration of the servers and the networking environment.

2.2.1 Server configuration

The node is a powerful SoC Supermicro X10SDV-TLN4F with an Intel® Xeon® processor D-1540/1541, 16GB DDR4, two 10Gb LAN interfaces and a 500GB hard disk.

This node runs the VMWare ESXi Operating System that hosts all management Virtual Machines. Currently, the following VMs are deployed:

- Emulab BOSS Server: This server provides an SFA GENI v3 interface and takes care of all testbed management services like DHCP, DNS, PXE-booting, disk imaging).
- Emulab OPS Server: This is a file server that provides NFS-mounted storage that can be shared between experimenters in the same project.
- AMQP server: This server enables the OMF EC to communicate with the OMF RC's that are pre-installed on the DUT nodes. The OMF EC should be installed on one of the DUT nodes.
- OML Database server: This server is used by default by the DUT nodes to store all experiment measurement data.

The Fed4FIRE compliant testbed management framework along with its corresponding client tools is documented in detail in D5.2.

2.2.2 Network configuration

For connecting the management server to the devices under test the 24 ports HP ProCurve 2920 Power over Ethernet Plus (PoE+) Layer 3 Managed switch is used. In that way a gigabit Ethernet and a 30W PoE power supply can be delivered to the DUT nodes.

2.3 SUT Network

The Portable Testbed is only to assure a communication between all DUT nodes, but we do not constraint an experimenter to use any specified type of DUT nodes. Moreover, he even does not have to contain any SUT Network Controller, if he is testing distributed control algorithms.

However, in order to demonstrate how the Portable Testbed operates, we set up a SUT WiSHFUL controller in our testbed and run WiSHFUL Agent applications in each DUT node.

2.4 SUT WiSHFUL controller

SUT WiSHFUL controller manages the System-Under-Test network that consists of several DUT nodes. The controller is commanding DUT nodes using Unified Programming Interfaces (UPIs) that have been defined within the WiSHFUL project. The WiSHFUL [1] framework is used to implement the example SUT controller.

An Intel NUC mini-PC D54250WYKH¹ with an Intel i5 processor has been selected to run the SUT network controller application.

2.5 Devices Under Test (DUT's)

The devices under test are homogenous COTS Intel NUC (Next Unit of Computing) devices of model D54250WYKH¹. These are basically headless barebone PCs. They consist of an Intel Core i5 4250U processor, 4GB of ram, a gigabit Ethernet port, various USB3 devices (for sensor node extensions, etc...), a 320GB hard disk and two Wi-Fi cards: one 802.11n (WPEA-121N/W) and one 802.11ac card (WLE900VX 7AA).

2.5.1 Physical node configuration

The nodes can be bought with an internal Wi-Fi card, which is using internal antennas. To facilitate two Wi-Fi cards, and to mount their five external antennas, an extra support structure was needed.

IMINDS designed and developed this prototype by creating a 3D model and printing it using an Ultimaker 2 3D printer.

The goal was to reuse as much of the NUC case as possible and to make the assembly of the nodes as easy as possible. The final design was created in the OpenSCAD suite. Figure 4 shows a screenshot of the code and preview. The original base plate fits nicely into the structure. This is not only positive for ensuring good airflow, it is also a great assembly aid because the SMA cables can be easily reached and tightened (while the structure is already attached on the NUC).

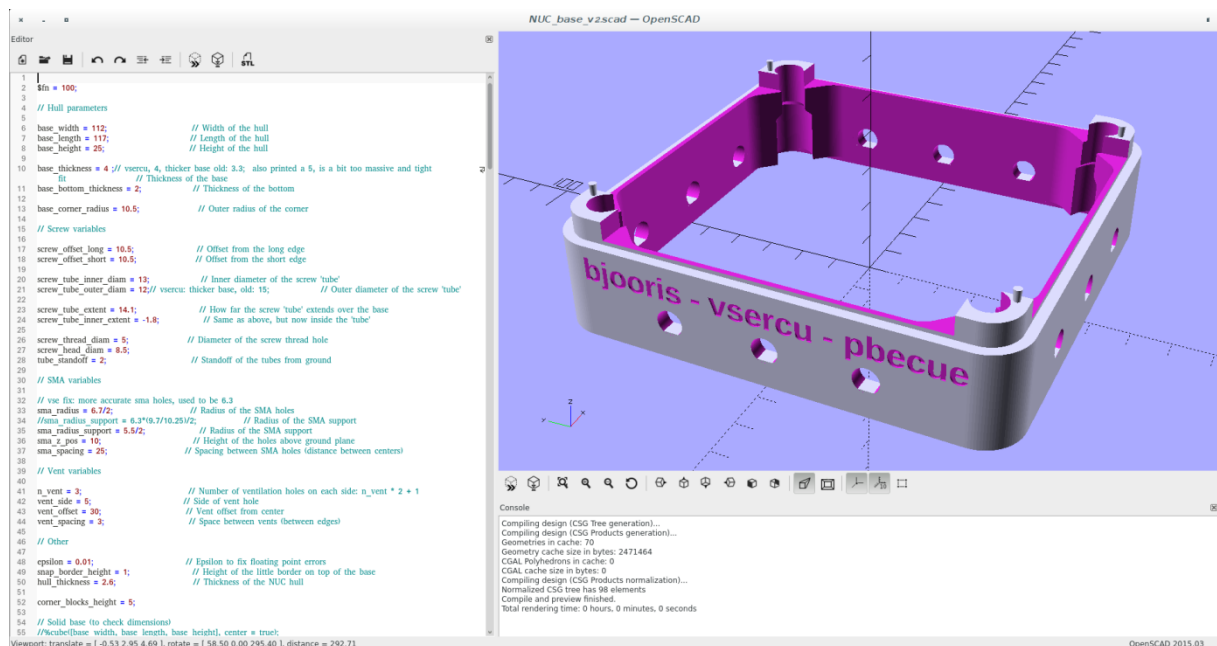


Figure 4: NUC-base design in OpenSCAD

This model has room for 12 SMA-connectors to make the mounting of the antennas flexible. Also the four shafts in the corners fit perfectly within the NUC case. The small towers on each shaft were created in a specific way so creating a cast of the 3D print for easy replication is possible.

¹ <http://www.intel.com/content/www/us/en/nuc/nuc-kit-d54250wykh.html>

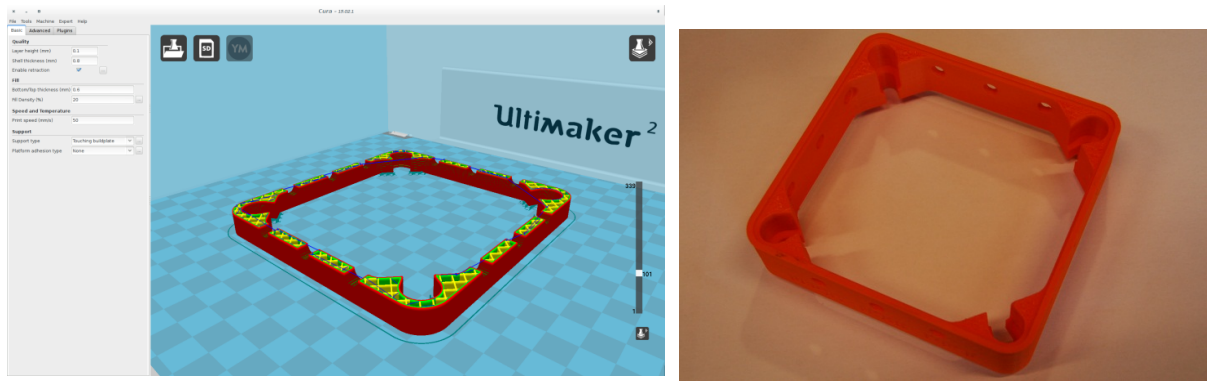


Figure 5: Ultimaker2 software and final prototype

The model is printed using PLA plastic and a final preview is shown in

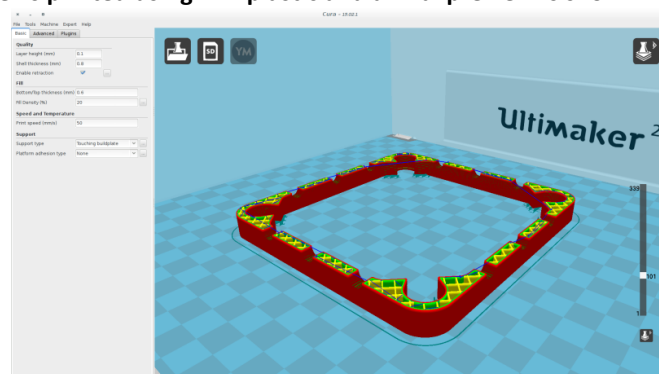


Figure 5.

To mount the model, special screws were designed with the same look as the original ones. Illustration of the technical drawing can be seen in Figure 6.

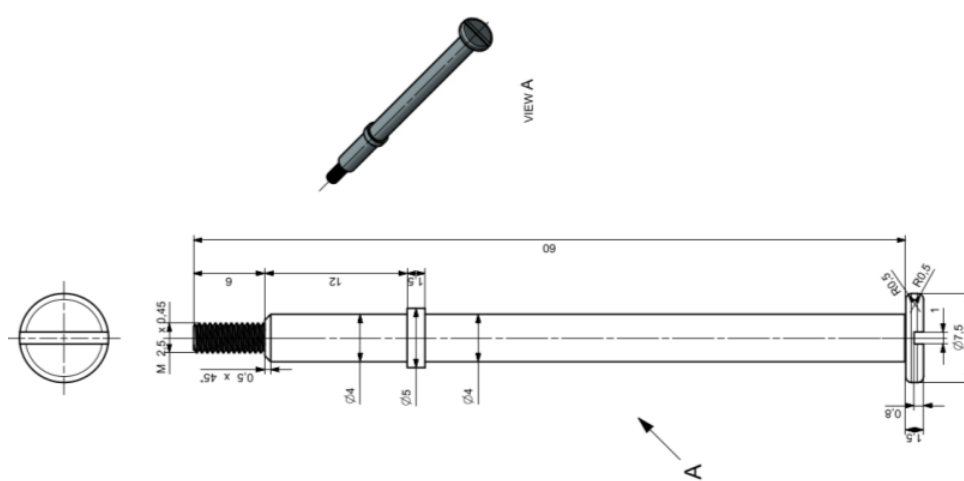


Figure 6: Special screws

The printing of each model takes quite some time (8 hours). To cope for this, IMINDS explored a polyurethane (PUR) solution. Using a silicon cast of the 3D model, two liquid agents (polypol and isocyanate) can be mixed and poured inside. The agents harden within 15 minutes. Details and pictures of this production process will follow, as it is still work in progress.

2.5.2 Network and power configuration


There are several ways to connect the DUT nodes to the central management machine that hosts the experiment provision and control VMs.

Power and network solutions often go hand in hand. The different combinations are explained in Table 1 (see also Deliverable D6.1 section 5.2).



Figure 7: The PoE box on top fits nicely with the rest of the NUC chassis.

Table 1: Network and power delivery options for the DUT

Solution	Backbone Network	Power	Pros	Cons
Wired	UTP, 1000Base-T (Gigabit Ethernet)	Power over Ethernet	Low latency and fast connectivity One cable for power and network	Require (long) cables Limited to cable length Max. 30 watts power
Ethernet over Power	Ethernet over Power devices, ~300Mb/s 	Wall socket	Fast connectivity Power can be delivered from AC grid	Deployment is limited to locations of AC plugs
Battery	Wireless backbone (further explained in section 4.1.3)	Power bank or battery pack	Deploy virtually anywhere	Duration of the experiment limited to battery capacity Backbone not designed to provision node images (gigabyte magnitude)

Up to now, the wired solution is used (illustrated in Figure 7). The black box on top of the NUC is the PoE splitter.

2.6 Portable testbed

The operation of a set of BN Nodes is harmonized by a dedicated BN Controller (BNC). To facilitate this purpose the Backbone Control Channel is utilised. This channel might be transported by different technologies – either wired or wireless, the latter one being unavoidable in case of tests with widely separated nodes and tests under heavy environmental conditions, i.e. outdoors, manufacturing environment, etc.

2.6.1 Backbone Network

The Portable Testbed supports advanced self-configuration – essentially start being operational "out-of-the-box", i.e. after having connected pair-wise DUTs nodes with their peer BN node, the whole set-up is booted up and the Backbone network establishes full connectivity.

We prepared default OS image (Linux-based OS) for each BN node and proper configuration files that are applied after system boots-up. Proposed approach eliminates overhead of configuration that has to be done on the BN nodes. Thanks to default configuration, all BN nodes join the wireless backhaul network on default channel after successful start-up. An experimenter does not have

access or control over BN nodes, but by using TMS he has the ability to change the channel that will be used by Backbone network during an experiment and configure Queuing Disciplines to prioritize control traffic over monitoring traffic.

In order to make Portable Testbed work as intended, the default configuration that is applied to each BN has to contain the following:

Channels that may be used for Backbone – depending on the wireless technology and frequency band available, a number of channels can be configured. It is at the PT operator's discretion to select channels that do not interfere with SUT.

A unique network name and key – each backbone network is uniquely identified and separated by a name and security key.

The Backbone is a mesh routing network, which is used to connect TMS, EMS, SUT-NC and DUT nodes. We used OLSR as routing protocol. To make it transparent and keep all mentioned nodes unaware of being inter-connected through portable testbed, dynamic logical L2 network over the backbone network is created. In D6.1 it was stated that static routing will be used to setup communication between DUT nodes, but we decided to change to logical L2 networks. The advantage of this approach is that it does not require configuration of routes in DUT nodes as it believes to be connected to an ordinary switch. To achieve this, we dynamically set up L2 tunnels between mesh devices. We have developed an L2TP-based point-to-multipoint tunnelling solution that supports direct data delivery if a destination's location is known and uses multicasting if not or if the data needs to be broadcasted. The tunnelling is handled by a custom kernel module which works as a learning switch for discovering the location of MAC addresses. The multicasting is handled by the software by setting up a tree from a source mesh device to the different destinations. Forwarding is subsequently done by installing *iptables* rules.

We have also included a so-called chaining feature, which allows extension of the OLSR meshing over wired interfaces. This way the PT operator can choose to (temporarily) interconnect one or more BN's with an Ethernet connection. Essentially, it allows the operator to overcome network partitioning or increase bandwidth at bottleneck wireless links. The chaining feature relies on a dedicated VLAN and will always attach the lowest OLSR weight to a wired link.

2.6.2 Backbone WiSHFUL controller

A WiSHFUL Backbone Controller (BNC) is an entity that controls behaviour and parameters of BN nodes. We implemented BNC using UPI functions defined in the WiSHFUL project.

The current version of BN controller supports BN node discovery – section 3.1, and integrity monitoring – section 3.2. Moreover, it uses UPI to install defined Queueing Disciplines in BN nodes – section 3.4, and to set operating channel for BN wireless network – section 3.3.

An Intel NUC mini-PC with Intel i5 processor is used to execute the Backbone Controller application.

2.6.3 TMS-BNC Interface

An additional interface between TMS and BN Controller was introduced. Using TMS-BNC interface, the TMS do not have to talk to each BN node individually. Instead, it talks only with BNC, which is then responsible for controlling and managing the BN nodes.

2.6.4 BN Node

When deploying a Portable Testbed, it is safe to assume this might happen under any kind of unpredictable conditions. We have therefore selected a hardware platform that can cope with harsh, outdoor conditions, if need be. Currently we rely on the Gateworks Laguna platform, which offers the following interesting features in its GW2388-4 model:

- Dual Core ARM11 SoC (600MHz)
- 256MB DDR2 RAM
- Four High-Power Type III Mini-PCI Sockets
- Two GbE Ethernet Ports
- Real Time Clock with Battery Backup
- Voltage and Temperature Monitor
- 8 to 48VDC Input Voltage Range or PoE powered
- -40°C to +85°C Operating Temperature

When paired with a proper outdoor enclosure (see Figure 8) it provides a solution that can be deployed either in or outdoor anywhere in the world, without any additional measures required.



Figure 8: IP67 enclosure for Gateworks Laguna GW2388-4

In Year 2, we will transition to the newer Gateworks Ventana platform, with the following features:

- Quad Core ARM Cortex A9 SoC (1GHz)
- 1 GB DDR2 RAM
- Four High-Power Mini-PCIe Sockets
- Two GbE Ethernet Ports
- Real Time Clock with Battery Backup
- Voltage and Temperature Monitor
- 8 to 60V DC Input Voltage Range or PoE powered
- -40°C to +85°C Operating Temperature

The WiSHFUL agent running on the nodes relies on a querying component that is part of the main software and exposes all necessary internals as key-value pairs. These are “translated” to be exposed through WiSHFUL UPI’s. Work is ongoing to cut out this translation step, and integrate the UPI’s directly into the main components.

3 Implemented capabilities

3.1 BN node discovery

During deployment of Portable Testbed, it is important to find proper initial configuration that allow each BN node to be connected to Backhaul. In order to be able to know which nodes manage to connect to BN, a discovery procedure was implemented.

The current version of BNC allows a testbed operator to define a list of BN nodes that have to be connected to the Backbone network. A BN node is defined using its hostname. After boot-up, each BN node tries to connect to BNC using an IP address that was provided in a configuration file. Upon successful connection, it sends some information, like hostname, OS image version, connected TUD, etc., to BNC. The BNC collects these data from each node.

The BNC reports to the testbed operator, which BN nodes are connected, along with a list of nodes that did not manage to connect. Having this knowledge, an operator can reorganize which nodes require reconfiguration/relocation.

3.2 Integrity monitoring

As the wireless medium is not as reliable as a wired one, constant monitoring of presence of connected nodes is required. BN Controller is responsible for keeping status information about all connected nodes, thus it needs to have a way to continuously track presence of each node in the network.

The BNC have to inform the TMS (which informs an experimenter) if BN nodes disappear, because in such case experiment should not be continued as results may be corrupted. Of course, outage time is use-case specific, some control algorithms and experiments need to interact with DUT nodes very frequently, while others may send only commands on rare occasion. We provide default value of outage time - 10s, but we keep the possibility for experimenter to redefine it.

We implemented a mechanism based on exchange of Hello messages between each BN node and BNC to monitor the status of each BN node. After successful connection, each BN node has to send Hello message to BNC every few seconds. The time interval for Hello messages equals 3s. Also BNC sends Hello messages to each BN node with the same frequency. If Hello message is not received more than timeout, node assumes that its peer has disappeared.

3.3 Interference management

In case of a wireless backbone, other wireless networks might be operating on the same area as SUT network, possibly interfering with it and corrupting wireless experiment results. To overcome this issue, an additional mechanism was implemented that makes (if possible) SUT and BN networks work on orthogonal frequency bands.

In order to avoid interferences between SUT and BN networks, an additional step is performed before experiment starts called Channel Policy Agreement (CPA). Using TMS, an experimenter specifies channels that will be used during experiment. The TMS sends this list to BNC, which further decides which channels can be used in BN network. These two frequency bands sets cannot overlap. Then BNC configures BN nodes to use one of available channels. The UPI function *setChannel()* is used for that purpose. Of course, if an experimenter claims to use all channels supported by BN network, it is not feasible to avoid interferences. In such case, proper notification is presented to the experimenter and he is then able to decide whether to abort the experiment or

start it with possible interference between SUT and BN networks. The portable testbed uses IEEE 802.11 as wireless communication technology for the BN network.

3.4 QoS support

The wireless network is often a bottleneck in terms of throughput. Moreover, long delays and packet loss may have a huge negative impact on control plane. As some flows are more important than others to successfully complete an experiment, they have to be recognized and given higher priority over others.

We used Linux *Traffic-Control* subsystem to differentiate and prioritize flows in layer 3 of protocol stack (NET) and we made use of the Enhanced Distributed Channel Access (EDCA), which is an extension of the basic DCF defined in IEEE- 802.11e standard, to provide prioritization of flows in layer 2 (MAC).

We provide default configuration of Queueing Discipline (QDisc) for flows expected to be present during an experiment (see D6.1). In this configuration, the Priority Scheduler (PRIO) is the root QDisc attached to Backbone interface of BN node. By default, it contains four FIFO queues (pfifo) (in priority order): 1) Backbone Mesh Traffic Queue, 2) Testbed Management Traffic Queue, 3) BN WiSHFUL Control Traffic Queue, and 4) Default Queue. Such configuration allows prioritizing traffic related to Portable Testbed maintenance over traffic related to an experiment and SUT Network control. All traffic is classified with set of defined filters, which segregate packets with rules based on 5-tuple.

As we do not constrain an experimenter to use experiment management tools nor particular implementation of controller, we are not able to provide default QDisc configuration to prioritize flows related to an experiment and SUT Network control over monitoring and result collection traffic. Instead, we give an experimenter a possibility to define additional packet queues to prioritize this traffic. Note that it is not mandatory, and if such configuration is not provided, all traffic will go through Default Queue.

We use UPI function, *installEgressSheduler()*, to setup defined QDisc in each BN node.

In order to provide prioritization in MAC layer, we exploit the fact that EDCA mechanism provides four access categories (AC): *AC_BK (background)*, *AC_BE (best effort)*, *AC_VI (video)* and *AC_VO (voice)*. Each AC is characterized by specific values for the channel access parameters: *i)* Minimal Contention Window (*cwmin*) value; *ii)* Maximal Contention Window (*cwmax*) value; *iii)* Arbitration Inter-frame Space (*aifs*) value; *iv)* Transmission Opportunity (*txop*) value. Different values of these parameters allow to statistically prioritizing channel access for one AC over another. As there are four packet queues associated with each AC, it is possible to assure prioritized access for crucial packets (flows).

In EDCA, packets are classified to proper queue based on Type-of-Service value in its header. Thus, we used UPI function, *setFlowTransmissionQueue()*, which configures iptable rules to set proper TOS value in packets of defined flow.

4 Packaging and transport

In order to provide flexible means of transport for the portable testbed, several paths were explored. An easy to carry, robust and spacious case was the desired solution. It also needed protective material on the inside so the delicate electronics are not damaged during transport.

The theatre and music industry uses “road-”, “ATA-” or “flight cases” for this purpose. These boxes are largely made of (ply) wood with metal reinforcements on the edges and corners. They also feature secure locking mechanisms.

4.1 Flight-case

4.1.1 Single box solution

There are several suppliers who build flight cases, IMINDS chose “All Cases” in Belgium because of good pricing and fast response.

The first idea was to build a single large case on wheels, with some handles at the edges. A schematic is displayed in Figure 9. It would be approximately 52cm wide, 70cm high and 53cm deep or 19 inch wide and 15 units high (these standards are also used in data racks).

The idea was to mount three drawers of approximately 20cm in height that would house the portable testbed hardware.

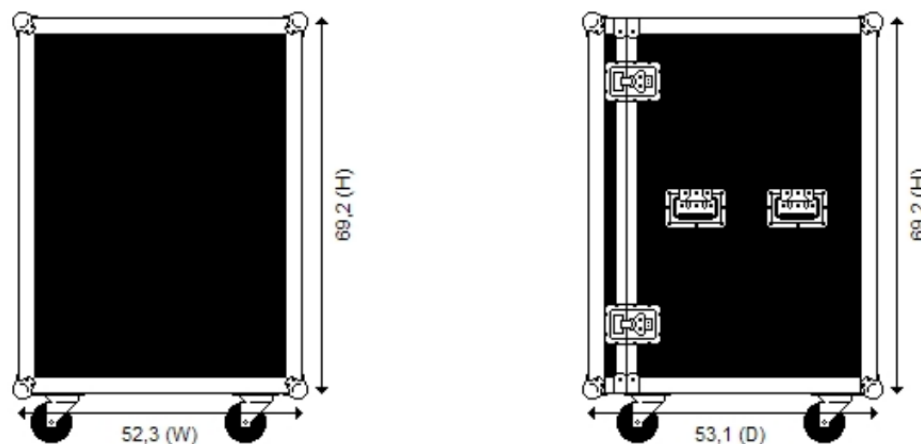


Figure 9: First flightcase design

The major disadvantage of this solution was the weight. After several discussions and estimations with the supplier, the finalized box would weigh about 250kg when empty. This is proved to be largely impractical. The Birch plywood on the sides covers large surfaces and needed to be 9mm thick to provide adequate strength. The drawers are a second layer of wood (or plastic) but are not really necessary because the outer shell is sufficient.

Another disadvantage was that if someone would open several drawers at once, the entire box could tip over. So a mechanism would be needed to avoid this (adding even more weight), or the drawers could only be opened for 75% (adding difficulty to deploy).

All in all, the single-box solution is too heavy and large to be really practical.

4.1.2 Separate briefcase solution

A second idea was to take the drawers outside the box (described in section 4.1.1) and fortify them. This approach has the advantage that there is only one layer of outer shell, which saves a lot of material and consequently, weight.

IMINDS thought of a design to house several components to their separate cases:

- One robust case to house the testbed server and main networking units (switch, power supply, ...)
- A set of two lightweight cases to hold the nodes.

The robust case is made of the same plywood (but a thinner, 7mm variant) and metal reinforcements described in section 4.1.1. This case will be heavier than the others, but it also houses the largest and most expensive equipment. The dimensions are (W, H, D): 52cm, 22cm, 42cm.

Again we chose here a 19 inch width, so rack mounts can be placed and a switch can be easily mounted. It is also possible to open the case on two sides, the front and rear panels are removable.

Figure 10 **Error! Reference source not found.** shows a schematic of the inside of the case.

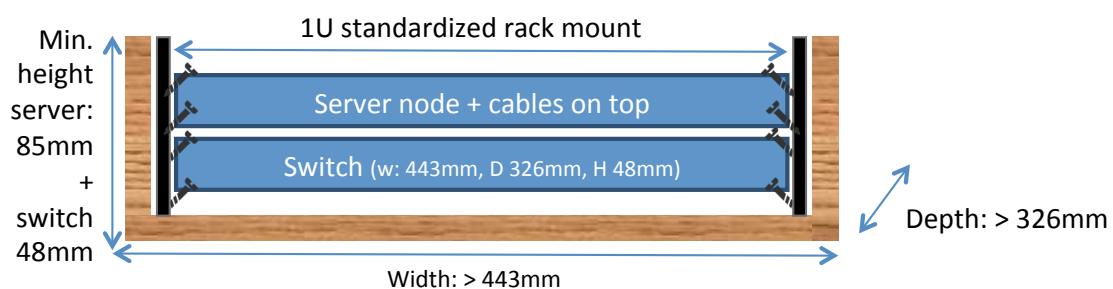


Figure 10: Schematic of the inside of the robust case

After finishing and wiring the switch inside, **Error! Reference source not found.** shows a picture of the actual case. Note that the empty space on top of the switch will need to be filled up with a 19-inch rack mount for the server node, which is currently being ordered.

The lightweight cases are made of hard (but mostly hollow) plastic and aluminium reinforcements. One lightweight case would offer space for 4 testbed nodes; a technical drawing is presented in Figure 11 **Error! Reference source not found.**

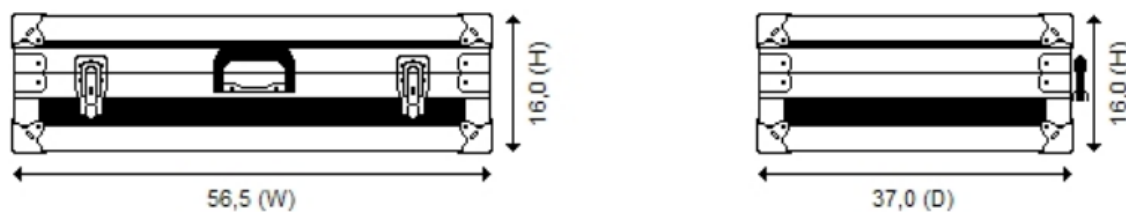


Figure 11: Dimensions and schematics of the lightweight flight cases

To fix the nodes inside the case, the foam option was chosen. A base of hard foam is glued to the bottom of the case, it is cut specifically to fit the devices under test that were described in section 2.5.1.

In the top of the briefcase, softer, more flexible polyurethane foam is used, as its only function is to push down softly on the nodes so they stay in place in the harder base (see Figure 12).

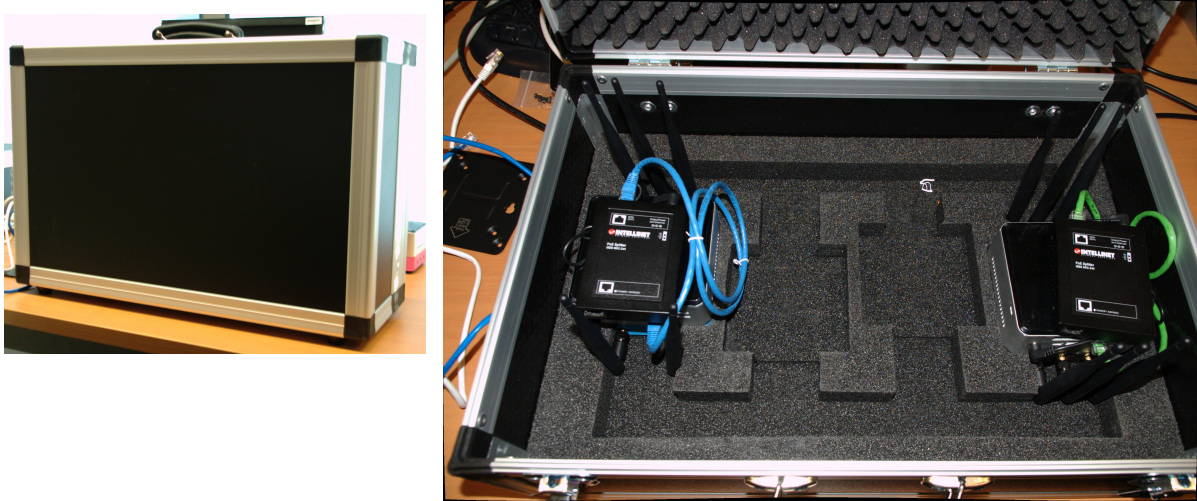


Figure 12: Actual lightweight flightcase with DUT inside

4.1.3 Future battery powered solution

In order to cope with battery powered idea in section 2.5.2 and also described in section 5.2.2 in the previous deliverable D6.1, bigger “lightweight” flight cases will be needed.

An approach to provide current and connectivity (while in the flight case) to the node was via the “power-tool” mechanism (see Figure 13).



Figure 13: Power-tool battery

This implies that the form and size (mostly the height) of the lightweight boxes will need to be modified to fit this solution. A case will have to be designed that attaches to the DUT device to the battery and to the charging mechanism.

It is also desired that the battery pack has the functionality to be discharged while being charged. In that way, an experimenter does not have to cope with power loss (and rebooting DUT) when deploying the DUT (thus disconnecting the battery from its charger).

Lastly, it is also practical that a data stream can be transmitted over this physical connection, for provisioning the experiment while the DUT is still in the flight case.

5 Future improvements and extensions

This deliverable will also present improvements and extensions that are planned to be implemented in Year 2.

5.1 Automatic DUT node to BN node mapping discovery

Since, we do not want to restrict an experimenter to connect particular DUT with proper BN node nor make testbed operator to maintain static mapping between these two nodes manually, additional DUT-BN mapping discovery mechanism has to be implemented. A BN node has to be able to get to know what DUT node is connected on the other side of the wire, create a mapping entry and send it to BNC that aggregates all entries and pass created list to TMS.

There are two possible solutions:

- Install simple client application on DUT that will try to connect to server application installed on BN node. After successful connection, DUT sends all available information to BN.
- Register DUT node in TMF database with MAC of wired interface as key and node information as value. Then, a BN node sniff packets that it receives on wired interface and extract source MAC address out of header.

The former solution requires implementation and installation of additional application on both DUT and BN nodes while the latter does not. On the other hand, with former solution experimenter does not have to register node in TMF database, because all needed information can be embedded on DUT.

5.2 DUT Power-supply control

After time of experiment is over, TMS calls Resource Release procedure which frees resources by withdrawing the access right of the experimenter. As we allow an experimenter to upload his own OS image with own configuration to any DUT node in our testbed, we need a way to regain the control over resources. Our solution is to reboot DUT node and make it boot with default OS image with known configuration.

The main problem is to make DUT node to reboot. Currently BN node provides only connection to DUT node, but it does not power it. One possible solution is to use controllable power switch (on a specially designed PCB, this was described in section 5.3.1; D6.1) that will be installed between power supply and DUT power input. A BN node has to be able to control this switch (probably using GPIO). Having this functionality and proper DUT-BN mapping list, BNC can provide TMS proper functions (over TMS-BNC interface) to reboot particular DUT node, when needed.

5.3 Channel Policy Agreement improvement

It is certain that BN network will not use any other channels besides those that are assigned to it. But we need a way to prevent an experimenter from using different channels than the ones he reserved. In case of Wi-Fi technology, TMS can generate new regulatory database and install it in DUT nodes in OS system (if it supports these functionality) deployed by experimenter. Thanks to this solution, an error message will be displayed when experimenter tries to set channel that was not reserved. This solution has one main drawback, i.e. experimenter is free to change any configuration of DUT nodes and in particular he can reinstall his own version regulatory database.

In case an experimenter does not behave fair and for any different technology than Wi-Fi, we need to monitor frequency bands used by BN network. If interferences coming from SUT network are discovered, the testbed maintainer has to be notified about such incident and experimenter should receive warning on his email account.

5.4 Channel Access optimization

Due to the multi-hop transmission that is frequent in wireless mesh networks, some links (especially, links near gateway) are aggregating traffic from other nodes and can become bottleneck in the network. In order to mitigate this issue, a mechanism for channel access optimization needs to be implemented. The basic idea is to measure the current load of each link in network, then compute and assign proper *back-off time* for each one. The *backoff-time* for a link with large traffic is assigned a smaller value than that for a link with smaller traffic, thus congested links can be activated more often. As it is expected to improve mesh network performance, it should be also integrated to be used in portable testbed.

In [2] similar mechanism, called *Fixed Backoff-time Switching* (FBS) was presented. Authors conducted experiments and proved that their solution performs better than pure CSMA protocol. The FBS method is completely distributed, i.e. each node monitor load on all its links and computes *back-off time*. Our idea is to gather all links load information in WiSHFUL Backbone Network controller and having this knowledge compute *back-off time* for each wireless link.

In order to control channel access probability, we will make use of already implemented UPI that allow to change the parameters of DCF function for each hardware queue in wireless interface, namely: *i)* Minimal Contention Window (cwmin) value; *ii)* Maximal Contention Window (cwmax) value; *iii)* Arbitration Inter-frame Space (aifs) value; *iv)* Transmission Opportunity (txop) value. The description of mentioned UPI function is provided in D4.2.

6 Conclusion

Within Year 1 of the WiSHFUL project, we have implemented the first release of the Portable Testbed.

First, we have defined the role of each entity that is part of the Portable Testbed. We have specified the hardware and according software for each entity of the Portable Testbed, and also motivated the selection of the hardware and software. We have developed extensions for easy transportation and powering the hardware.

The following **capabilities** are available in the first release of the Portable Testbed release:

- *discovery and integrity monitoring mechanisms* for facilitating the deployment and maintenance of Portable Testbed.
- *a new TMS-BNC interface for DUT node discovery*, passing the list of used channels to the BNC.
- *a mechanism for flow prioritization* to prioritize control traffic over monitoring traffic
- *a 'Channel Policy Agreement' procedure*, called by the BNC in order to select channel for the BN network that is orthogonal relative to SUT network.

For the **packaging and transportation** of the hardware constituting the portable Testbed flight cases are used. We decided to use a set of smaller cases, which are more practical for transportation.

Improvements and extensions, to be implemented in the next release by the end of Year 2, have been identified. The main goal is to further facilitate the deployment of experiments using the Portable Testbed, reducing the configuration and maintenance overhead, and improving the performance of the backbone network.

7 References

- [1] Source code available at: https://github.com/WirelessTestbedsAcademy/wishful_upis/tree/master
- [2] S. Sukaridhoto, N. Funabiki, T. Nakanishi, K. Watanabe, A Linux Implementation Design of Fixed Backoff-time Switching Method for Wireless Mesh Networks, IEICE Technical Report NS2012-67