



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

Project Deliverable D6.1

Specification of testbed on the move

Contractual date of delivery:	30-06-2015
Actual date of delivery:	29-06-2015
Beneficiaries:	TUB, IMINDS, nCENTRIC
Lead beneficiary:	TUB
Authors:	Mikolaj Chwalisz (TUB), Anatolij Zubow (TUB), Piotr Gawłowicz (TUB), Adam Wolisz (TUB), Pieter Becue (IMINDS), Vincent Sercu (IMINDS), Ingrid Moerman (IMINDS), Robin Leblon (nCENTRIC), Giuseppe Bianchi (CNIT), Ilenia Tinnirello (CNIT), Alice Io Valvo (CNIT)
Reviewers:	Peter Ruckebusch (IMINDS)
Work package:	WP6 – Testbed on the move
Estimated person months:	3.5
Nature:	R
Dissemination level:	PU
Version:	4.6

Abstract:

This deliverable provides the specification for the testbed controller, wireless backbone network, wireless control channel, deployment strategies for the portable testbed and equipment portability. This document will serve as a guideline for the preparation of equipment and the development of the first implementation of the portable testbed that will be available at the end of year 1.

Keywords:

wireless backbone, control channel, quality of service, portability

Executive Summary

This deliverable provides the specification for the portable testbed implementation. As the portable testbed is one of the use cases of the WiSHFUL project, it has to be controlled through the same UPIs as the solution under test network (SUT). The deployment strategy for the testbed controller and its functionalities are described in this document. Since the backbone network is the main building block of the portable testbed, two solutions, namely IEEE 802.11s and the NCENTRIC mesh solution based on OLSR, allowing fulfilling its requirements are examined. Next, the Wireless Control Channel and its requirements are defined. The control channel is split into five flow types with different QoS requirements. The flows are grouped into three priority classes according to their importance. Then, two possible solutions allowing providing differentiation and prioritization of these classes are described. We examined the usage of the *pfifo_fast*, which is the basic Queuing Discipline available in the Linux and IEEE 802.11e standard, which defines four different traffic classes. Furthermore, two optimizations of the wireless control channel are proposed. The specification for packing and deployment strategies are described and several powering solutions are investigated. Finally, the specification for the SUT nodes and few possible candidate devices are presented.

List of Acronyms and Abbreviations

AC	Access Class
AIFS	Arbitration inter-frame spacing
AP	Access Point
BE	Best Effort
BK	Background
COTS	Commercial off-the-shelf
CPU	Central Processing Unit
CW	Contention Window
DCF	Distributed coordination function
EDCA	Enhanced Distributed Channel Access
EMS	Experiment Management Server
FET	Field Effect Transistor
FIFO	First In First Out
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISM	Industrial, scientific and medical radio bands
ISO/OSI	International Organization for Standardization/Open Systems Interconnection
KPI	Key Performance Indicator
MAC	Medium Access Control
MAP	Mesh Access Point
MP	Mesh Point
MPP	Mesh Portal
MPR	Multi Point Relay
NAT	Network Address Translation
OLSR	Optimized Link State Routing Protocol
PC	Personal Computer
PLC	Power Line Communication
PoE	Power over Ethernet
QDisc	Queuing Disciplines
QoS	Quality of Service
RTT	Round Trip Time
SDR	Software Defined Radio
SON	Self-Organizing Network

STA	Station
SUT	Solution Under Test
TC	Traffic Control
TCP	Transmission Control Protocol
TOS	Type of Service
TXOP	Transmission Opportunity
UPI	Unified Programming Interfaces
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
VI	Video
VO	Voice
WiFi	Wireless Fidelity

Table of contents

1	Introduction	7
2	Portable testbed controller.....	7
3	Selection of most suitable technology for a first portable testbed implementation	9
3.1	IEEE 802.11s.....	9
3.1.1	Architecture of 802.11s mesh network	9
3.1.2	Mapping 802.11s to requirements for backbone network.....	10
3.1.3	Configuration for the portable testbed.....	11
3.2	NCENTRICMesh (OLSR-based)	11
3.2.1	Architecture of NCENTRIC OLSR based mesh network	11
3.2.2	Mapping of OLSR to requirements for backbone network.....	13
3.2.3	Configuration for portable testbed.....	14
4	Wireless control channel.....	14
4.1	Wireless control channel requirements.....	15
4.2	Wireless control channel specification	15
4.2.1	Backbone Wireless Control Program	16
4.2.2	SUT Resource Reservation and Provisioning.....	16
4.2.3	SUT Experiment Control.....	16
4.2.4	SUT Experiment Monitoring	17
4.2.5	SUT Wireless Control Program.....	17
4.3	Presence of control flows in time	17
4.4	Quality of Service	18
4.4.1	QoS requirements.....	18
4.5	QoS implementation	19
4.5.1	Flow Classification.....	19
4.5.2	Serving flows	20
4.5.3	Usage of <i>pfifo_fast</i> queuing discipline.....	20
4.5.4	Usage of 802.11e standard	22
4.6	Wireless Control Channel Optimisation.....	24
4.7	Connection of Backbone and SUT nodes	24
4.8	Monitoring backbone network.....	25
5	Portable testbed packaging and deployment strategies.....	25
5.1	Deployment duration.....	26

5.1.1	Short term	26
5.1.2	Mid Term	26
5.1.3	Long Term	26
5.2	Cabling and power	26
5.2.1	Power toggling	27
5.2.2	Powering nodes using batteries.....	27
5.2.3	Power over Ethernet.....	28
5.2.4	Using AC wires and fixed infrastructure.....	29
5.3	Backbone network	29
5.3.1	Power control functionality	29
5.3.2	Wi-Fi cards in WISHFUL boxes.....	30
5.3.3	Use of external USB sensor nodes	30
5.3.4	Separate wireless backbone node	30
5.4	Solution Under Test	31
6	Conclusion	31
7	References	33

1 Introduction

This deliverable reports on the work in WP6 "Testbed on the move". In particular, it provides the specification that will be used for the portable testbed implementation during the first year of the project. Note that the term "Testbed on the move" is mainly referred to as "portable testbed" since it better reflects the actual concept, e.g. a testbed that is portable to any location rather than a testbed that moves.

The document first provides the description of the portable testbed controller. It defines the usage of the WiSHFUL UPIs and outlines the functionalities expected in the portable testbed. Second, in section 3, two candidate backbone technologies, namely IEEE 802.11s and the NCENTRIC mesh solution based on OLSR, are investigated. This section contains a brief description of each candidate backbone technology and maps the support for the portable testbed requirements on each of them. Finally, a description of the configuration for each technology is given to support the portable testbed backbone.

Section 4 lists the specifications for the Wireless Control Channel. The five different flow types expected in Wireless Control Channel are defined. Then, the required Quality of Service for each flow type is discussed and the changes that should be applied to meet the QoS requirements are described together with the recommendations for optimizing the Wireless Control Channel. Since, the same UPI is used for configuration of the backbone and the SUT (Solution Under Test) nodes, it is important to provide some level of protection that prevent experimenters to change backbone network configuration. The solution allowing such protection is also provided in this section.

In section 5, the portable testbed packing and deployment strategies are described. This section defines a few scenarios for portable testbed usage and covers the power and cabling solutions. Possible approaches how to provide power for the SUT and backbone nodes are investigated. Finally, the specification and few candidates for the SUT nodes are presented.

2 Portable testbed controller

The architecture of the portable testbed is presented in the Figure 1. As can be seen there are two distinct wireless networks (blue and yellow) present in the testbed, namely Backbone (Backbone) network and SUT (Experiment) network. It is expected that these two networks will be configured and controlled with the same WiSHFUL UPIs as illustrated in Figure 2.

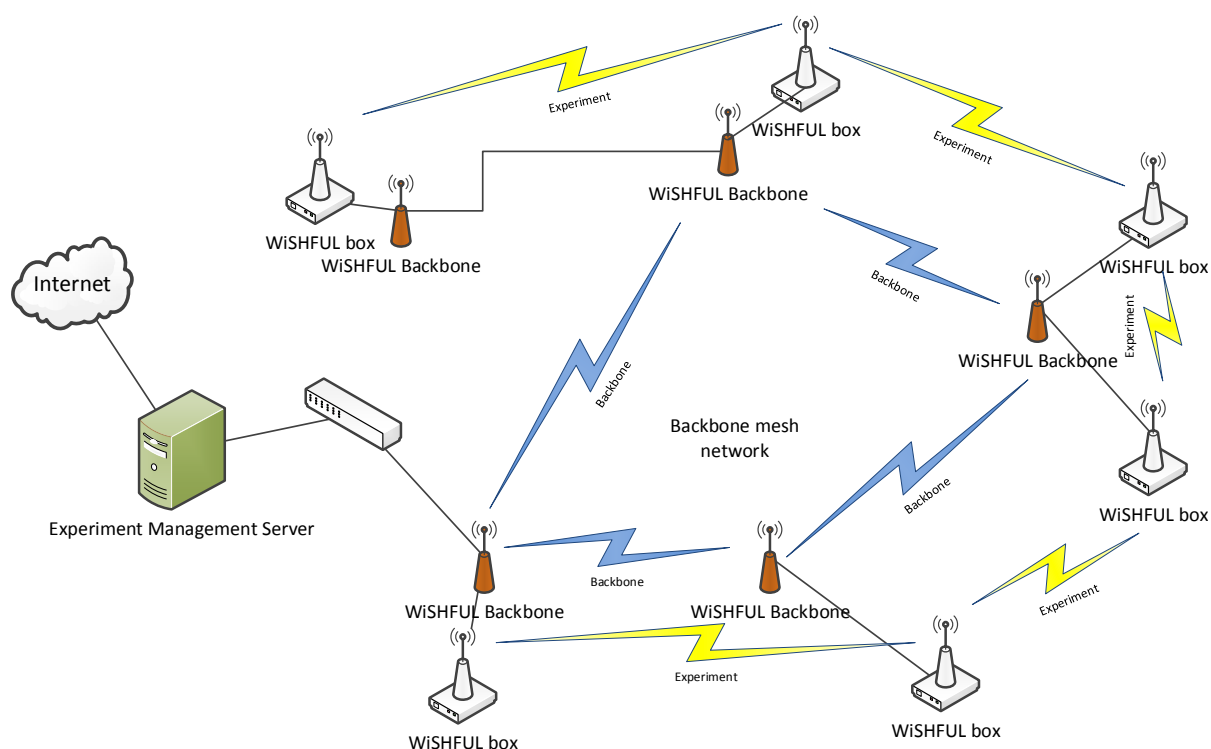


Figure 1. The portable testbed architecture.

The two controller applications (one for backbone and one for SUT network) are installed in the Experiment Management Server (EMS). The application for controlling the backbone network can be accessed only by the testbed owner, so some protection and authentication is required. There is an option to separate control of the backbone and SUT network into two servers. This solution would provide a physical separation of control functionalities, but the same result can be achieved by virtualization techniques.

The controller of the backbone network is expected to provide at least the following functionalities:

- Scheduling the experiment based on experimenters request
- Reservation of resources for a particular experiment
- Provision of the required software and images to the SUT nodes
- Traffic shaping configuration
- Monitoring of the backbone network
- Configuration of prioritization (setting QDisc, classes and filters that are available in *tc* tool)
- Testing of connectivity between any pair of backbone nodes
- Performing automatic setup and recovery of the network
- Monitoring the transmission parameters of the backbone network (like TCP RTT, packet loss, throughput)
- Providing the possibility to change the channel of the backbone network on every node in the network in a consistent way (without losing the connectivity)

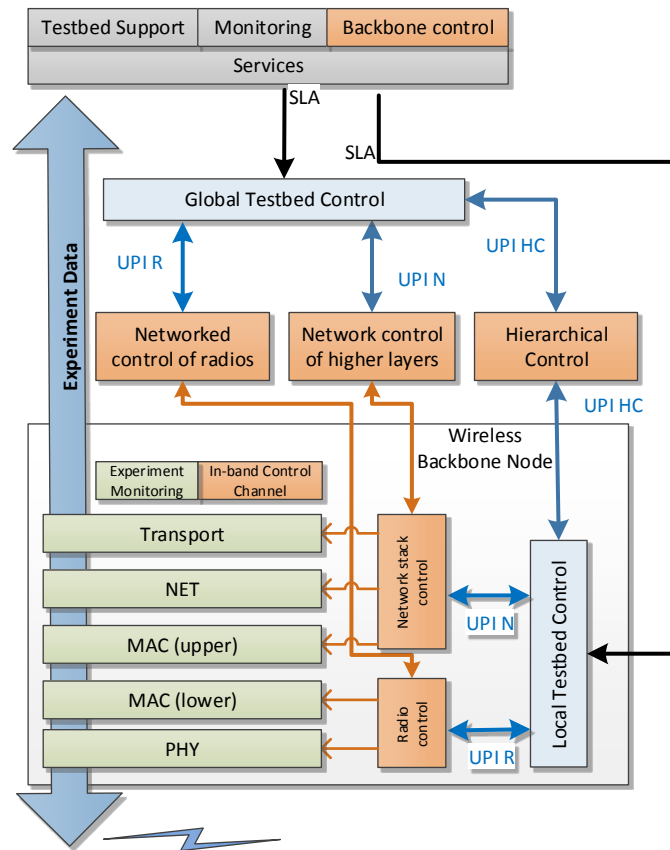


Figure 2. WiSHFUL UPIs are used to control the wireless backbone of the experiment network.

3 Selection of most suitable technology for a first portable testbed implementation

This section provides a description of the technologies that allow implementing the backbone of the portable testbed.

3.1 IEEE 802.11s

IEEE 802.11s is an extension to the IEEE 802.11 standard, which allows multiple wireless nodes to communicate with each other without requiring an AP between them, i.e. without infrastructure. It operates on Layer 2. With the 802.11s mesh standard, the nodes can form a multi-hop network where all of the links are wireless. This means that no wired infrastructure is required to setup the network. This feature is very crucial, especially in the case of remote deployments, when no wired infrastructure is available. However, if needed and possible, wired links can be used to directly connect nodes or provide connection between different wireless mesh networks over a long distance.

3.1.1 Architecture of 802.11s mesh network

The IEEE-802.11s mesh network architecture is presented in Figure 3. The IEEE 802.11s defines three kinds of nodes [1]:

1. **Mesh Point (MP)** is a simple station with support for mesh functionality (for example a mesh frame format), i.e. it is capable to join and participate in mesh network. A MP uses the Peer Link Management protocol to discover neighbouring (those in range of an MP) nodes and keep track of them. To reach nodes over multiple hops, a MP uses the Hybrid Wireless Mesh Protocol (HWMP), which is the routing protocol for Layer 2 and supports two kinds of path selection protocols.

2. **Mesh Portal (MPP)** supports the same functionalities as a MP, but it is also supports interconnecting two networks, i.e. a MPP provides the integration of mesh network with other network (for example the Internet). It is achieved by bridging at least two interfaces to provide gateway functionality. A MPP usually announces its presence to the other mesh members.
3. **Mesh Access Point (MAP)** is an AP with additional mesh functionality. Being an AP and a mesh node at the same time, it provides station (STA) connectivity with the mesh network and, in some cases, the MPP.

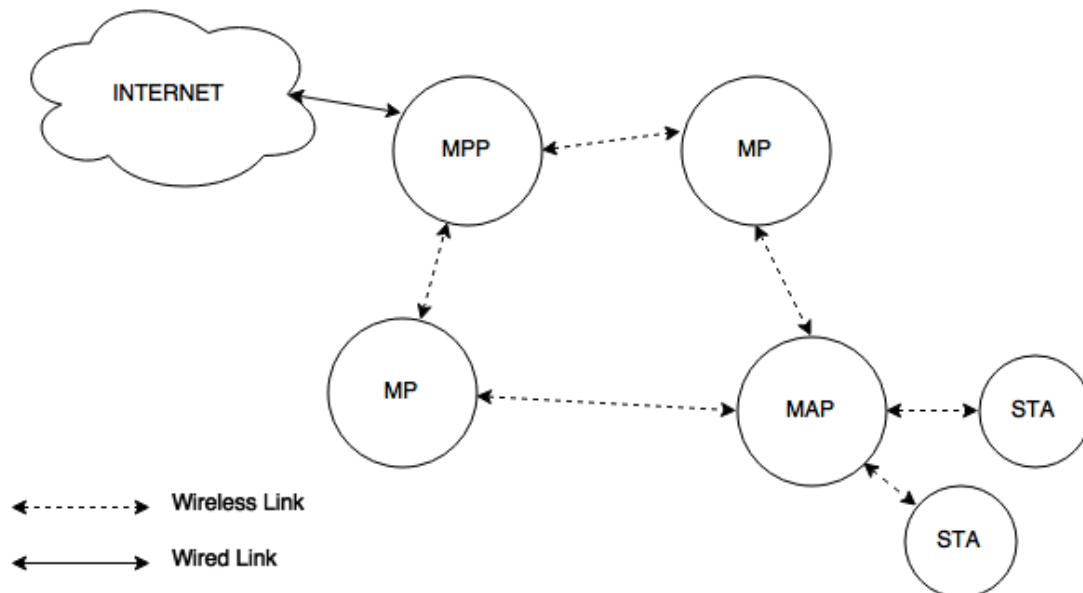


Figure 3: IEEE-802.11s mesh network architecture

3.1.2 Mapping 802.11s to requirements for backbone network

In Table 1 Functional Requirements related to the backbone network defined in deliverable D2.1 are mapped to IEEE 802.11s technology.

Table 1. Functional Requirements for the backbone network mapped to 802.11s

UC#	Actors	UC name and short description	Support level in 802.11s
2.1	Backbone Nodes	Wireless backbone interference	Medium
	Global Testbed Control	The wireless backbone network must not interfere with the experiment network.	802.11s operates on standard WiFi technology, so there is the possibility of changing the channel. Nevertheless, one channel is occupied by the backbone network and is not available for experiment
2.2	Backbone Nodes	Backbone connectivity	High
	Global and Local Testbed Control	The wireless backbone has to provide a fully connected network i.e. each WISHFUL box can connect with each other. If the backbone nodes are not in the direct communication range, relaying has	Thanks to the path selection algorithms (routing for layer 2), the 802.11s protocol is able to provide full connectivity of the network.

		to be provided.	
2.3	Backbone Nodes	Backbone transparency	High
	Global Testbed Control	The wireless backbone network has to be transparent to the experimenter.	802.11s operates on Layer 2 of the ISO/OSI standard. Assuming all configurations are done over IP, the backbone network is transparent for experimenter.
2.4	Testbed Nodes	Testbed auto-configuration	High
	Global Testbed Control	It should be possible to deploy the testbed with minimal effort. That means nodes should be able to automatically setup the network that can be used by experimenter.	Using the Peer Link Management protocol, 802.11s is able to discover new nodes and also notice node disappearance. The network topology is recalculated by the path selection algorithm.

3.1.3 Configuration for the portable testbed

Each backbone node should be configured as a Mesh Point and should be connected in a single network. It is possible to provide the initial configuration with channel ID and Mesh ID to make all nodes to form a mesh network just after being switched on. To prevent unexpected nodes from accessing the mesh network and make the network secured, the *wpa_supplicant* or *authsae* tools should be used. These tools are responsible for handling all authentications and peering, the user needs only to provide configuration and password.

The best way to connect an EMS with the mesh network would be to configure one backbone node as Mesh Portal and connect the EMS to it. The Mesh Portal has to announce that it is connected to external network by setting *mesh_gate_announcement* parameter to 1.

Internet connectivity for the mesh network should be provided through the EMS. This will allow for controlling the Internet traffic flow using traffic shaping and filtering.

The connection between a backbone node (Mesh Point) and a SUT node is presented in section 4.7.

3.2 NCENTRICMesh (OLSR-based)

The routing inside a NCENTRIC OLS based mesh is based on a heavily customized OLSR network. The Optimized Link State Routing Protocol (OLSR) is developed for mobile ad hoc networks. It operates as a table driven, proactive protocol, exchanging topology information with other nodes of the network regularly. Each node selects a set of its neighbour nodes as "multipoint relays" (MPR). In OLSR, only MPRs, are responsible for forwarding control traffic, intended for diffusion into the entire network.

3.2.1 Architecture of NCENTRIC OLSR based mesh network

The OLSR mesh network architecture is presented in Figure 4. OLSR standard defines two kinds of nodes:

- **Multi Point Relays (MPR)** are nodes that relay messages between nodes. They also have the main role in routing and selecting the proper route from any source to any desired destination node. MPRs advertise link-state information for their MPR selectors (a node selected as a MPR) periodically in their control messages.
- **Stations (STA)** are nodes not selected as MPR. Note that those nodes could also be selected as MPR when the network topology changes.

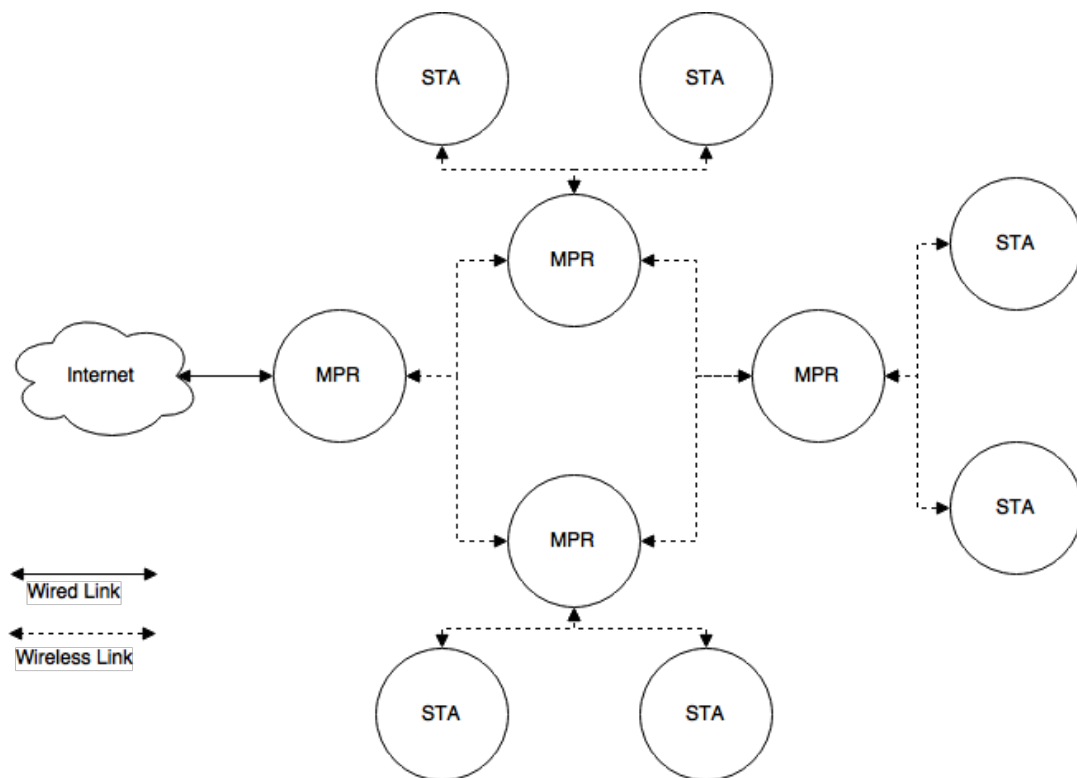


Figure 4 OLSR mesh network

Using the extensions provided by nCentriC each MPR mesh device can be configured as an Internet access device, providing either wired or wireless access. The different access interfaces can be grouped together to form one L2 network similar to other wireless devices. The access network can have several modes:

- **Standalone or generated:** In this mode the device simply configures all related IP address information, makes sure that any required AP's are started and advertises the availability in the mesh. All data from and to the network is then routed without any modifications.
- **L2-Roaming:** This mode sets up L2 tunnels between mesh devices. A custom point-to-multipoint tunnelling solution has been developed that supports direct data delivery if a destination's location is known and uses multicasting if not or if the data is being broadcast. 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 NCENTRIC software by setting up a tree from a source mesh device to the different destinations. Forwarding is done by installing *iptables* rules.
- **Gateway:** Any mesh device can be configured to act as a gateway to reach external networks. Wired, wireless and 3G uplinks are supported.
 - **Wired:** With a wired uplink a check is made if a physical link is detected. If so the interface is configured using DHCP or through a static IP address, gateway and DNS server from the configuration.
 - **Wireless:** To set up a wireless uplink, one of the wireless interfaces is set in to client mode and instructed to connect to the configured SSID. Open access points, WEP encryption and WPA are supported (the latter through *wpa_supplicant*). Once a connection with the AP is made the interface is configured using DHCP or through a static IP address, gateway and DNS server from the configuration.
 - **3G:** With the proper hardware connected to a board a 3G uplink can be set up by the software.

The software can handle multiple uplinks and will select the best uplink based on pre-configured weights for the different uplinks. These weights can be reconfigured depending on the required use case. Other gateways that are discovered in the mesh are also taken into account in this process.

3.2.2 Mapping of OLSR to requirements for backbone network

In **Error! Reference source not found.**Table 1Table 2 the functional Requirements related to the backbone network defined in deliverable D2.1 are mapped to the NCENTRIC OLSR based solution.

Table 2 Functional requirements for the backbone network mapped to NCENTRIC OLSR

UC#	Actors	UC name and short description	Support level in nCentriC OLSR based solution
2.1	Backbone Nodes	Wireless backbone interference	High
	Global Testbed Control	The wireless backbone network must not interfere with the experiment network.	OLSR operate on Wi-Fi technology, so there is possibility of changing the channel in the ISM-band. There is also a possibility to use frequencies such as 2.4GHz ISM, 900 MHz ISM, 3.5GHz licensed, etc. These bands have been tested by NCENTRIC and performed as expected.
2.2	Backbone Nodes	Backbone connectivity	High
	Global and Local Testbed Control	The wireless backbone has to provide a fully connected network i.e. each WiSHFUL box can connect with each other. If the backbone nodes are not in the direct communication range, relaying has to be provided.	OLSR is a L3 routing protocol that is designed and optimized for mobile and wireless ad hoc networks. It is hence a perfectly possible to create a fully connected backbone network using OLSR.
2.3	Backbone Nodes	Backbone transparency	High
	Global Testbed Control	The wireless backbone network has to be transparent to the experimenter.	Within certain bandwidth and latency constraints, the use of OLSR in the backbone network is transparent for experimenters. The actual constraints vary for each set-up based on the frequency, transmission range and external wireless interference.
2.4	Testbed Nodes	Testbed auto-configuration	High
	Global Testbed Control	It should be possible to deploy the testbed with minimal effort. That means nodes should be able to automatically setup the network that can be used by experimenter.	OLSR is designed to dynamically form a network between nodes by exchanging hello message pro-actively. It will hence automatically form and maintain a mesh network, transparent for the experimenter.

3.2.3 Configuration for portable testbed

As stated the default OLSR implementation was extensively modified by NCENTRIC in order to form a more robust wireless network that can serve as a backbone in very challenging conditions. For this purpose the following changes were necessary:

- The neighbour discovery process using HELLO messaging has been replaced by information obtained from the Scanning and Binding modules.
- The default Best Effort OLSR routing algorithm has been implemented. The current release of the software uses an extension of this algorithm that makes a choice between equal shortest-path routes by selecting the route that meets the criteria in the two-hop neighbourhood (note that we do not take into account metrics from the entire route, nor are they signalled in the TC messages).
- In real-world applications we have noticed that the signal strength indication is not necessarily a good indication of the actual quality for sending data of the link. The cost-based routing algorithm works like a normal shortest-path routing algorithm, except that any link can have an additional cost between 0 and 250 assigned. The best route towards a destination is the route with the smallest sum of the hop-count and costs along the route. This implies that if you have two routes with equal cumulative cost, that the shortest one will be preferred. Costs of links can be assigned manually or can be increased automatically based on the statistics obtained by periodic bursts of ping messages on the links. Setting the cost of a link higher than the maximum hop-count in the network will make sure that this link is not used unless there is no alternative route.
- The implementation also provides message aggregation as proposed by the OLSR RFC.

Devices can be configured using a text-based configuration file. Part of the configuration file is static, the other part is generated by a start-up bash script (e.g. to add node-specific settings). A configuration class is available for reading from and writing to configuration files. It also supports setting default values for certain keys in case the key is not set in the configuration file.

Each device also offers the option of configuring it via a graphical web interface, accessible through HTTP (WebUI). The backend of the configuration process has been separated into a WebAPI. Any WebUI instance can connect to the API of another device and configure it, given that it is able to authenticate.

The current version of the software supports 802.11 a/b/g/n networks based on ath9k driver and *hostapd*.

4 Wireless control channel

In this section the specification for the Wireless Control Channel is described. A wireless control channel (backbone) plays a crucial role while experimenting on the portable testbed. It is required for situations where a wired backbone network is impossible or not desired to setup. In Figure 1, the Wireless Control Channel is depicted as blue links, interconnecting the WiSHFUL backbone nodes. In Figure 5 all flow types expected in the testbed are presented. The current design provides separate channels for Control and Data traffic. Flow types marked blue are responsible for controlling the backbone network and setting up the SUT nodes (image/program provisioning, reservation and configuration of SUT nodes). Usually, they are used by testbed owner to configure and manage experiments for testbed clients. Flows marked red are responsible for controlling the SUT network. They are a part of the interface that is available for the testbed experimenters and allow them to configure their experiments and nodes, and collect the monitoring data.

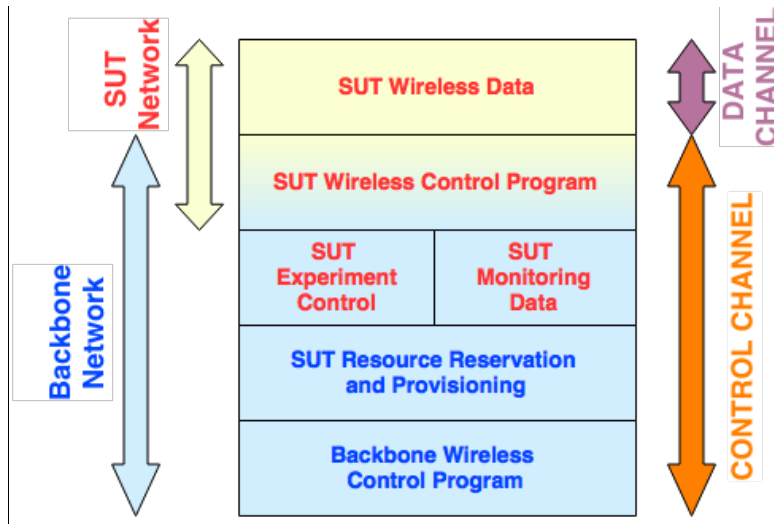


Figure 5: Flow types expected to be present in the portable testbed

4.1 Wireless control channel requirements

The Wireless Control Channel (backbone) is a key enabler for the portable testbed deployment. It allows a flexible and inexpensive method for interconnecting WiSHFUL nodes, which now can be deployed at almost any place. As the wireless backbone is replacing the wired backbone, it has to provide the same functionality as its wired version. The use of a wireless backbone should be transparent for the testbed experimenter, i.e. the same functionalities and operations need to be available and there should not be any degradation of the KPIs. Moreover, setup of the control network should be as simple as possible and not require an extensive configuration effort from the testbed operator. Plug-and-play deployment of a Wireless Control Channel requires self-organizing network (SON) functionality. A SON provides three functionalities:

- Self-Configuration
- Self-Optimization
- Self-Healing

Thanks to the self-configuration functionality, all WiSHFUL backbone nodes should interconnect and form a backbone network after booting. If there are new nodes trying to connect or leave the existing backbone network, it has to reconfigure itself, using the self-optimizing functionality, again providing the required KPIs. In case of node failure, the self-healing functionality should make sure that the backbone network reorganizes rapidly to mitigate its effects. It has to be noted, that all WiSHFUL and backbone nodes have to be equipped with an Ethernet port to make initial configuration or reconfiguration possible.

The operation of the wireless backbone network cannot have any influence on the performance of the SUT. Because of this reason, the backbone network has to operate on an orthogonal frequency with the wireless experiment channel.

4.2 Wireless control channel specification

The backbone network provides the functionality to control the portable testbed. As depicted in Figure 5, different flow types are transmitted by the backbone and SUT network. Control data is carried mostly by the backbone network, but the *SUT Wireless Control Program* data can be transmitted by both the SUT and the backbone network. We expect that the *SUT Wireless Control Program* data will be mostly transmitted using the SUT network, but in the first portable testbed implementation (until the end of year 1), we will use the backbone network for it.

In the following section all flow types expected to use the wireless control channel are described.

4.2.1 Backbone Wireless Control Program

The wireless backbone also requires configuration and monitoring. For instance, it must be possible to change the wireless frequency (channel) of the backbone network. For this purpose the same WiSHFUL UPIs as for the SUT nodes control are used. Note that, the control channel for the backbone is in-band with the control for SUT and experiment.

The Backbone Wireless Control Program is used for configuring backbone networks. It provides the global control for WiSHFUL backbone nodes. Since it is the most important part and provides basic functionality to configure the portable testbed, the testbed needs to be able to install it just after the WiSHFUL nodes are booted and without any additional work from the network operator. This requires that an initial, basic, configuration is available, which allows all nodes to form the backbone network and exchange the Backbone Wireless Control Program data.

It is essential to provide the network operator with a tool to check the status of the backbone network, i.e. user must have the possibility to check connectivity between all nodes. In case the backbone channel interferes with the SUT wireless transmissions, the network operator should have a way to reconfigure the backbone network, i.e. operate on a different frequency. However, during reconfiguration, all nodes must remain accessible. For instance, if the testbed operator starts changing frequency of the backbone network on each node sequentially, it can happen that the connectivity chain will be broken and it will be not possible anymore to undo configuration without a manual intervention. One possible solution for this issue is to schedule frequency change on all nodes somewhere in the future (a few seconds should be enough), allowing all nodes to switch to the new frequency at the same time, recreate the backbone network and check connectivity with all other nodes. If during this process, one or more nodes cannot be reached anymore, all nodes should rollback to the previous configuration.

The Backbone Wireless Control Program flows are real-time flows, i.e. they require low latency transmissions. The reliability of the communication is also crucial, which requires special measures to deal with packet loss. On the other hand, only few control messages of small size are expected to be carried by these flows, so there is no need for a high bandwidth.

4.2.2 SUT Resource Reservation and Provisioning

These types of flows are used by the testbed operator to setup the portable testbed, provision the required programs or software images and configure them. These flows are not available for the experimenters.

Since images and programs can be transmitted, a high bandwidth is needed. Fortunately, the configuration data and images/programs are transmitted only once before the start of the experiment. These flows are non-real time (elastic) flows and are used only during the bootstrapping phase of the portable testbed, i.e. a high latency can be tolerated.

4.2.3 SUT Experiment Control

The SUT Experiment Control is used by experimenter to setup and configure experiments, for example the experimenter can start/stop experiments on all nodes at the same time. With the usage of this flow, experimenters can execute different actions on all available nodes. For example, if the experimenter wants to use the *iperf* tool, he/she can execute a client application on one node and a server application on another one. These flows are real-time flows, i.e. low delay is required. Since, only small control messages are sent, a low bandwidth is sufficient. On the other hand, because these flows are carrying control data, communication must be reliable and special measures must be used to cope with packet loss.

4.2.4 SUT Experiment Monitoring

The SUT Experiment Monitoring flow type is used for the transmission of large amounts of data. They carry testbed measurements data requested by the experimenters from all or set of nodes and are directed to the EMS for storage. In most cases, the flows of this type require a high bandwidth, but can tolerate a high latency, hence this channel can use a best effort type of service. The reliability is not crucial, because in case of packet loss, the data can be recovered from the SUTs after the experiment. Nevertheless, monitoring data should be transmitted with reliable protocols, such as TCP.

4.2.5 SUT Wireless Control Program

The SUT Wireless Control Program is similar to the Backbone Wireless Control Program, but it is used by the experimenters. Basically, an experimenter can configure all SUT nodes via the WISHFUL UPI. Since the SUT Wireless Control Program flows carry control messages, the reliability is crucial. Also a low latency is required. The bandwidth is rather low, because only small control messages are expected to be transmitted.

4.3 Presence of control flows in time

In Figure 6 the different flows are mapped on the experiment life-cycle. In the first phase T1, the backbone network is formed and also some connectivity and performance test can be executed. During T1 only the Backbone Wireless Control Program is active. The T2 is the experiment bootstrap phase, during which the SUT nodes are setup, the resources are reserved and the firmware images and software programs are provisioned. Phases T1 and T2 are performed by the testbed operator, who prepares and configures the portable testbed for use by the experimenter. From T3, the experimenter starts execution of the experiment. He/she has to configure the experiment using the SUT Experiment Control program. For example an experimenter can setup a client-server application in a pair of SUT nodes and schedule a transmission between them. Then in T4, the experiment is started and the experimenter can control the settings of the wireless network using the SUT Wireless Control Program. SUT Experiment Monitoring data can also be gathered, but it is recommended to cache measurement data and collect them after the experiment finalizes. In T5, the actual SUT experiment is executed. In T6, experimenters can collect the monitoring data and still configure the SUT nodes, but the SUT experiment is finished. In T7, experimenters have the possibility to download the SUT Experiment Monitoring data, which was cached in backbone nodes. In T8, the experiment cycle is finished and the testbed resources are released.

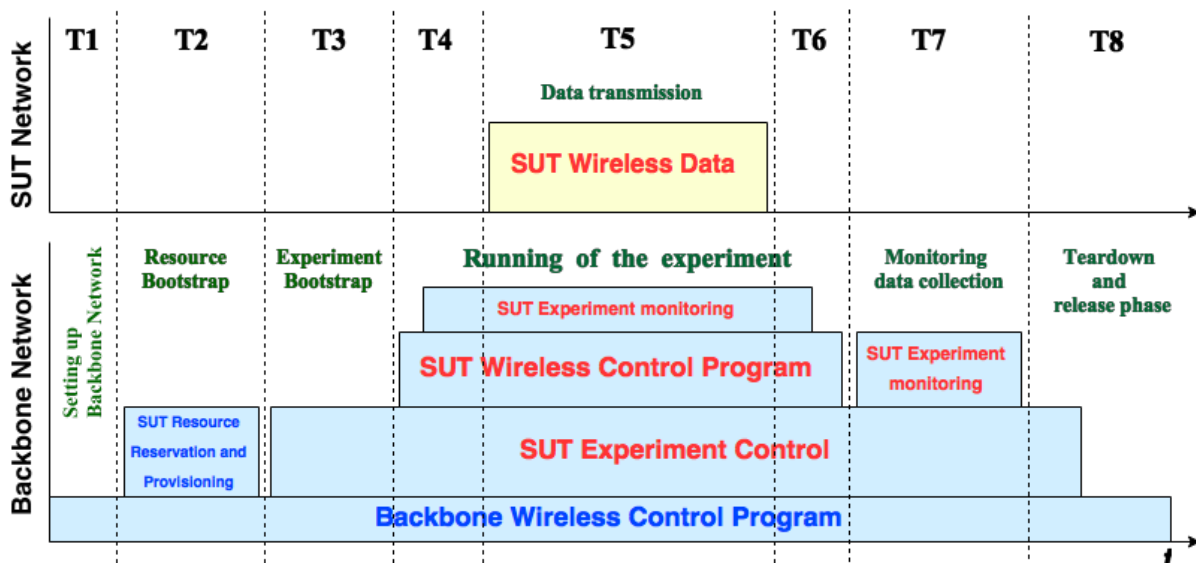


Figure 6: Presence of control flows during experiment

As can be seen, some flows for the SUT Resource Reservation and Provisioning are used only once in the beginning of the experiment and some others like the Backbone Wireless Control Program flow are active during the entire time of the experiment. It should also be noted, that during the experiment four flow types are active at the same time, so it is important to prioritize control flows over the data flows.

4.4 Quality of Service

The idea of the wireless backbone is to replace the wired backbone in order to deliver a more flexible and cost-effective solution for the portable testbed. However, the same performance of wired networks can, in most cases, not be achieved in wireless networks. The purpose of this section is to provide the specification of the Quality of Service configuration, allowing a transparent mitigation from the wired to the wireless backbone without degradation of the KPIs.

4.4.1 QoS requirements

Wireless communication gives rise to a number of challenges. The factor that has the biggest impact on performance is the presence of interference. Because of it, the latency, jitter and packet loss increases and the throughput reduces. Moreover, if the nodes are mobile, the QoS will be further decreased. One solution that improves the performance is to avoid interference by using a different frequency in the backbone network and in the SUT experiment network, i.e. they should operate on orthogonal channels. The robustness and reliability of the communication in the backbone network can be further increased by using a mesh network and multi-path routing protocols.

As mentioned in section 4.2, five flow types are expected to be present in the backbone network. Each type is carrying traffic with different QoS requirements. Unfortunately, since all flow types depend on the WiSHFUL UPI, which is not yet fully defined, it is not possible to give the detailed and numerical QoS requirements. In Table 3, the estimated QoS parameters are presented using a three-level scale: *Low*, *Medium* and *High* to express the importance of the requirement.

Table 3. Backbone Channels Requirements

Flow type	Type	QoS Parameter		
		Latency	Reliability	Bandwidth
Backbone Wireless Control Program	Real-time	Low	High	Low
SUT Resource Reservation and Provisioning	Elastic	---	---	High
SUT Experiment Control	Real-time	Low	High	Medium
SUT Experiment Monitoring	Elastic	---	---	High
SUT Wireless Control Program	Real-time	Low	High	Low

Since all flow types are transmitted over the same backbone network and some of them are operational at the same time (as illustrated in Figure 6), they compete for resources. It is important to ensure that the QoS requirements of each flow are met. To guarantee the required level of QoS, some flows have to be prioritized over other. The basic idea is to prioritize the control traffic over non-real-time (elastic) traffic like experiment monitoring data.

In Table 4 the channel priority configuration is presented. The Backbone Wireless Control Program data has the highest priority (value of 0), because it carries the most important flows. The *SUT* Experiment Control Program and *SUT* Wireless Control Program are equally important, so they have the same priority of value 1. Since the *SUT* Resource Reservation and Provisioning and *SUT* Experiment Monitoring are carrying non-real-time (elastic) traffic, they are served in Best Effort (BE) manner.

Table 4. Channel priority configuration (lower value = higher priority)

Flow type	Priority
Backbone Wireless Control Program	0
<i>SUT</i> Resource Reservation and Provisioning	BE
<i>SUT</i> Experiment Control	1
<i>SUT</i> Experiment Monitoring	BE
<i>SUT</i> Wireless Control Program	1

4.5 QoS implementation

To implement the QoS, the Traffic Control subsystem of the Linux Kernel will be used. It is a set of programs that allows a user to have complete control over the packets passing through the system. The QoS implementation involves classifying the network traffic into categories and differentiating the handling of traffic based on its classification.

4.5.1 Flow Classification

As can be seen in Table 4, there are only three flow classes defined: *0,1* and *Best Effort (BE)*. These three classes are mapped on different Type of Service (TOS) values. The TOS is set of four bits in the IP header. When these bits are set, the datagrams can be handled differently by routers.

The following values for the TOS are defined [2]:

- **Normal Service** – no TOS value is set, so the traffic is served in a Best Effort manner.
- **Minimum Cost** – used when it is important to minimize the cost of a data transmission. The packets should be forwarded over the links that are the cheapest from a financial viewpoint.
- **Maximum Reliability**– used when it is important that packets reach their destination without requiring retransmissions. Packet with this TOS values should be sent via links with the highest reliability.
- **Maximum Throughput** – used when latency is not particularly important, but the end-to-end throughput is, for example bulk data transfers (e.g. large files). The network operator should configure the routers to send packets with this TOS via high-bandwidth routes.
- **Minimum Delay** – used when a low latency is important. For example, if there are few paths to the destination in the network, each one with different latency. Routers should sent packets with this TOS value over the path with the lowest latency.

It is important to note that usage of TOS value is reasonable, if all routers in the network can read it properly and are applying the same policy to the same flow class. The most important thing is that the same mapping is used on all nodes in the network.

The TOS value can be set by the *netfilter* tools (for example *iptables* tool) or directly by the application generating the packet. The former option requires the configuration of rules that match a

proper class. For example if one wants to set the correct value of the TOS field for an application that is using TCP protocol and destination port of 80, one should execute the following command on the Linux machine:

```
iptables -A OUTPUT -t mangle -p tcp --dport 80 -j TOS \
--set-tos Minimize-Delay
```

This approach requires that the type of the flows available in the Wireless Control Channel can be discovered. This can be done by defining the port ranges on which each flow type operates and then creating *iptables* rules that match these port ranges. This solution might be a bit complicated comparing to the latter one, where application is responsible for setting the TOS value in IP packet header by manipulating the socket with *setsockopt()* function.

4.5.2 Serving flows

As some flows are more important than other, it is crucial to prioritize the flows and favour flows with highest priority. In section 4.4, three different flow classes for the Wireless Control Channel were defined. The idea is that the real-time control flows (e.g. interactive flows) are giving priority over the elastic ones.

To meet the QoS requirements of all control flow types, two possible solutions are considered:

1. Multiplexing all flow types into a single wireless interface
2. Split real time and elastic flow types between two wireless interfaces

In solution 1, only one wireless interface is used, i.e. only one wireless channel will be occupied for the backbone network. In Solution 2, two wireless interfaces are used, allocating more bandwidth for the backbone network and allowing physical traffic separation of different control flows. Using this solution two wireless channels are used, resulting in two blocked channels for experimenters during their tests. Since, WiFi is used as the technology for building the backbone network, it is important to give experimenters as many WiFi channels as possible.

Hence, we decided to use only a single interface for the Wireless Control Channel. This maximizes the channels that experimenters can use during their experiments but requires that the different flow types can be multiplexed into a single wireless interface. To ensure that the priorities of each flow are respected, we propose to use the Queuing Disciplines (QDisc) available in the Linux kernel. In the following subsections, we propose two possible solutions for implementing QDisc:

- Usage of *pfifo_fast* queuing discipline, which is available in Linux kernel
- Usage of 802.11e standard, which defines 4 priority classes for traffic

4.5.3 Usage of *pfifo_fast* queuing discipline

This queuing discipline is the default one for all interfaces in Linux. It is based on a traditional First In First Out (FIFO) queuing discipline, but also provide some prioritization. It consists of three different bands, each containing one FIFO queue – Figure 7. Each band is associated with a different priority. The highest priority traffic is placed into band 0, the medium priority traffic into band 1 and the lowest priority traffic into band 2. As long as there are packets waiting in band 0, band 1 will not be processed and the same rule applies for band 1 w.r.t. band 2, i.e. packets are processed from the head of a given queue only if all other queues with a higher priority are empty. The *pfifo_fast* queuing discipline thus gives the possibility to define three classes of traffic with three different priorities. This also implies that the priorities defined in section 4.4 can also be mapped.

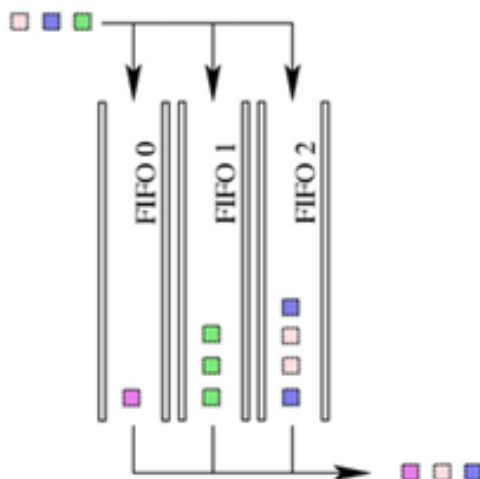


Figure 7: pfifo_fast queueing discipline [3]

Packets are directed to different band based on the Type of Service value of IP header. The basic mapping between TOS value and priority and band ID is shown in Table 5. For example, the Linux kernel inserts Minimum-Delay packets into band 0. There is nothing configurable for the end user about the *pfifo_fast* qdisc, but as packet priority is based on its TOS value, modification of this field is enough to take advantage of the scheduling features. The second column lists the way the Linux kernel interprets the TOS bits. For example, packets with TOS value of 12, are mapped to Linux priority value of 4, and are directed to band number 1.

Table 5. Mapping between TOS, priority and band in pfifo_fast queueing discipline [4]

TOS value	Linux Priority	Band
0	0 - Best Effort	1
1	1 - Filler	2
2	0 - Best Effort	1
3	0 - Best Effort	1
4	2 - Bulk	2
5	2 - Bulk	2
6	2 - Bulk	2
7	2 - Bulk	2
8	6 - Interactive	0
9	6 - Interactive	0
10	6 - Interactive	0
11	6 - Interactive	0
12	4 - Int. Bulk	1
13	4 - Int. Bulk	1
14	4 - Int. Bulk	1
15	4 - Int. Bulk	1

Table 6, lists the mapping of priorities and TOS value on each Wireless Control Channel flow. The control application is responsible for assigning each packet in the Wireless Control Channel flows the proper TOS value. The Linux kernel can then use the TOS value in the IPv4/IPv6 header to steer each packet automatically into the right band.


Table 6. Mapping between flow types of Wireless Control Channel and TOS Value

Flow type	Priority	TOS value
Backbone Wireless Control Program	0	8
SUT Resource Reservation and Provisioning	BE	4
SUT Experiment Control	1	12
SUT Experiment Monitoring	BE	4
SUT Wireless Control Program	1	12

4.5.4 Usage of 802.11e standard

The second option to implement QoS for the Wireless Control Channel is the usage of the IEEE-802.11e standard, which defines the Enhanced Distributed Channel Access (EDCA). EDCA is an extension of the basic DCF, introduced to support prioritized QoS. The EDCA mechanism defines four access categories (AC): AC_BK (background), AC_BE (best effort), AC_VI (video) and AC_VO (voice). The mapping between 802.1D to AC classes is presented in Table 7. Each AC is characterized by specific values for the access parameters. Different settings allow to statistically prioritizing channel access for one AC over another.

Table 7. AC relative priorities and mapping from 802.1D user priorities [6]

Priority	UP (same as 802.1D user priority)	802.1D Designation	AC	Designation
Lowest	1	BK	AC_BK	Background
	2	---	AC_BK	Background
	0	BE	AC_BE	Best Effort
	3	EE	AC_BE	Best Effort
	4	CL	AC_VI	Video
	5	VI	AC_VI	Video
	6	VO	AC_VO	Voice
Highest	7	NC	AC_VO	Voice

In EDCA, outgoing traffic is directed into four different queues (one for each AC). **Error! Reference source not found.** illustrates the default configuration with four ACs. Each queue is associated with an EDCA access function that uses the access parameters specified for each AC's.

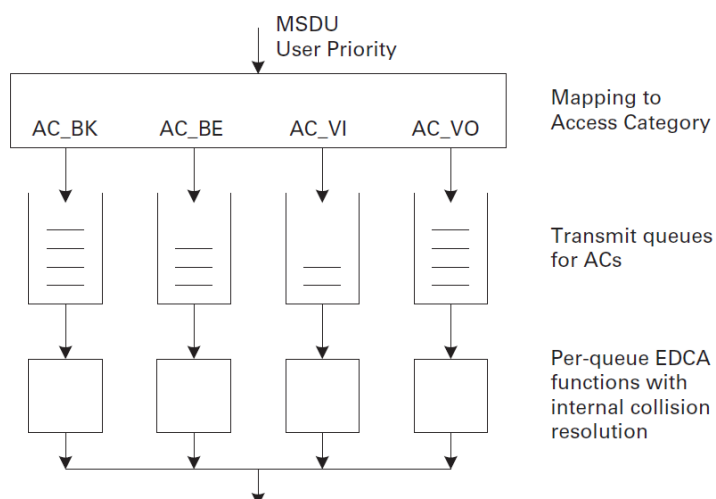


Figure 8: Architecture of 801.11e [5]

Table 8 lists the default EDCA access parameters for each AC. The parameters configured for EDCA are similar to ones used for the DCF, but here they are defined per AC, for example each AC has its own contention window value defined and referenced as CW[AC]. The EDCA for each AC works in a similar way as DCF. The EDCA functions compete for medium access by deferring for a fixed period of time, called the arbitration inter-frame space (AIFS). When two or more EDCA access functions get access at the same time an internal collision occurs. Such conflicts are resolved by allowing channel access to the AC with the highest priority. The other AC(s) behave as if an external collision has occurred. It doubles its contention window and tries to get access one more time. The access parameters can be manipulated by an AP.

Table 8. Default EDCA access parameters for each AC [6]

AC	CWmin	CWmax	AIFSN	TXOP limit
AC_BK	31	1023	7	0
AC_BE	31	1023	3	0
AC_VI	15	31	2	3.008 ms
AC_VO	7	15	2	1.504 ms

The EDCA access parameters describe the prioritization of one AC over another. The CWmin parameter gives strong differentiation of AC class priorities. The random back-off count is selected from the range [0, CWmin], using a lower CWmin value, also lowers the contention period of the AC statically. The CWmax is the maximal value of contention window that can be set for particular AC. Different values for AIFSN only has little impact on the priority differentiation of ACs. The transmit opportunity (TXOP) is a period during which a station can transfer data of a particular AC. Once the station starts its transmission, it can transmit or receive until the end of TXOP limit. If the TXOP limit has value of zero, it means that only one MSDU or management frame can be transmitted and then the procedure for getting access to the channel needs to be repeated. The TXOP limit therefore has a big impact on AC priority differentiation. AC's with a higher TXOP limit will get more air-time than AC's with lower TXOP value.

In the Linux operating system, it is possible to create four queues matching the ones available in 802.11e, i.e. each queue in the Linux kernel is connected to a corresponding queue in the IEEE-802.11e MAC. To sort the packets and direct them to a different queue, the TOS field in the IP header can be used.

As described in subsection 4.5.1, the control application is responsible to set the proper TOS value for each packet depending on its flow type. Then, using *netfilter*, it is possible to configure the filters used for directing the packets to the correct queue. Table 9, presents the proposed mapping between the Wireless Control Channel flow type and its priority, TOS value and AC.

Table 9. Proposed mapping between Wireless Control Channel flow types, priority, TOS value and AC

Flow type	Priority	TOS value	AC
Backbone Wireless Control Program	0	4	AC_VO
SUT Resource Reservation and Provisioning	BE	12	AC_BE
SUT Experiment Control	1	8	AC_VI
SUT Experiment Monitoring	BE	12	AC_BE
SUT Wireless Control Program	1	8	AC_VI

4.6 Wireless Control Channel Optimisation

The backbone network will not be congested during the experiment because the Backbone Wireless Control and SUT Wireless Control flows consist only of small control messages. This implies, however, that the SUT Measurement Data is not downloaded during the experiment otherwise the backbone network can be saturated. The data can be cached on each backbone node, until the end of the experiment. The EMC server can collect the SUT Measurement Data after the experiment and provide it to the experimenter.

When an experimenter needs the measurement data during an experiment (for example if an algorithm is tested that makes decisions based on live measurements in the network), more advanced measures need to be taken to avoid saturating the backbone network. Using prioritization, real-time control packets will be sent before elastic measurement data. If the wireless channel is still saturated, the maximum bandwidth will be limited for elastic flows by usage of Linux traffic shaping tool (*tc*). Traffic shaping is a particularly good fit for TCP traffic, since this protocol tries to get as much bandwidth as is available.

4.7 Connection of Backbone and SUT nodes

The backbone network is also used by control flows responsible for the configuration of backbone and SUT nodes. It is important to make isolation between these flows, i.e. only the testbed operator should be able to configure the backbone nodes. An experimenter can, off course, configure the SUT nodes. Such solution is required to prevent that experimenters change the configuration of the backbone nodes, which could break the connectivity of the backbone network and as a result corrupt the experiment.

Figure 9 presents the connection of backbone and SUT nodes. Each node has 2 Ethernet interfaces. These interfaces can be physical or virtual. The *eth1* interface in both nodes is controlled by the testbed operator and is used for transmission of the *Backbone Wireless Control Program* and *SUT Resource Reservation and Provisioning*. Interface *eth0* in backbone node is also controlled by the testbed operator and is a connecting point for the SUT node. Through the interface *eth0* in SUT node experimenter is able to configure the experiment and collect the measurement data in a similar

fashion as on the fixed testbeds. It is expected that *eth0* in SUT node will also be used for the *SUT Wireless Control Program*. Wireless interface *wlan0* in the backbone node is controlled by testbed operator and interface *wlan0* in the SUT node is controlled by the experimenter.

All flows from *eth0* and *eth1* interfaces are transmitted over the *wlan0* interface in the backbone node, so this interface is the gate which allows for all configurations. The interfaces of SUT node have to be accessible via the backbone network. For this reason, IP addresses for each interface have to be configured. Then, to make them “visible” via the backbone network, a few solutions can be applied. One solution is to use the NAT (Network Address Translation) technique. NAT can be easily configured but implies that a connection can only be initiated from inside the network. This limits the many cases where the central controller starts transmission of control messages. The second solution is to configure static routes. It is easy to add static routes in Linux, but it requires a lot of work from the testbed operator to setup the initial configuration. The third solution is to use a routing protocol to propagate information about connected nodes in the network. Using a routing protocol allows to automatically react in case of network changes. On the other hand, it requires a routing daemon to run in backbone node all time and also implies more traffic into backbone network. Since initial configuration for backbone node is almost always the same and is not changing in time, we believe that static routes are the best choice.

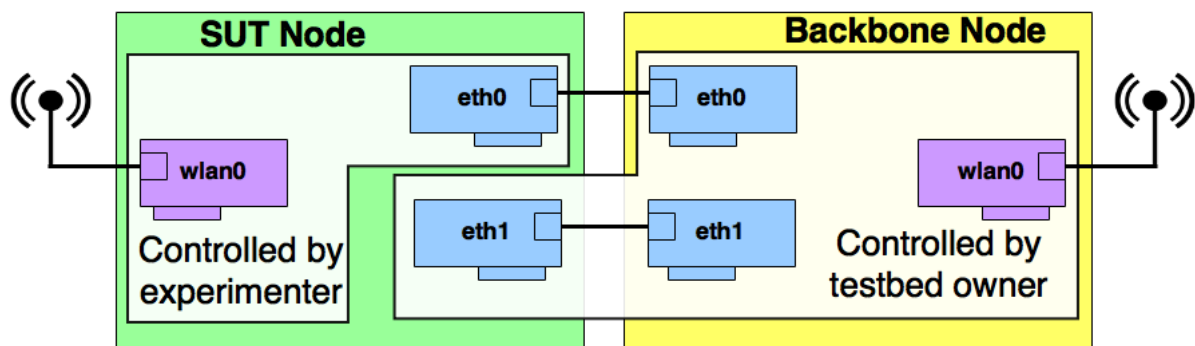


Figure 9: Connection of backbone and SUT nodes

4.8 Monitoring backbone network

It is important to provide the testbed operator and the experimenter a way to monitor the backbone network. The main parameters that they may be interested in are:

- connectivity, throughput, delay and packet loss between any pair of nodes;
- number of hops (intermediate nodes) between any pair of nodes;
- routing tables of backbone and SUT nodes;
- size of queues in backbone nodes ;
- TCP Round Trip Time (RTT).

This information makes it easier for the testbed owner and the experimenter to debug the network configuration and fix it if necessary. It is expected that WISHFUL UPIs will also be used for backbone network monitoring.

5 Portable testbed packaging and deployment strategies

The configuration and setup of the backbone nodes and solution under test, and the testbed in general can be one of three scenarios, mostly depending on the available infrastructure, hardware requirements and duration of the experiment.

The high-level methodology is depicted in Figure 10, and the paths are explained in the sections below. Note that no experiment is limited to each path, it can span several.

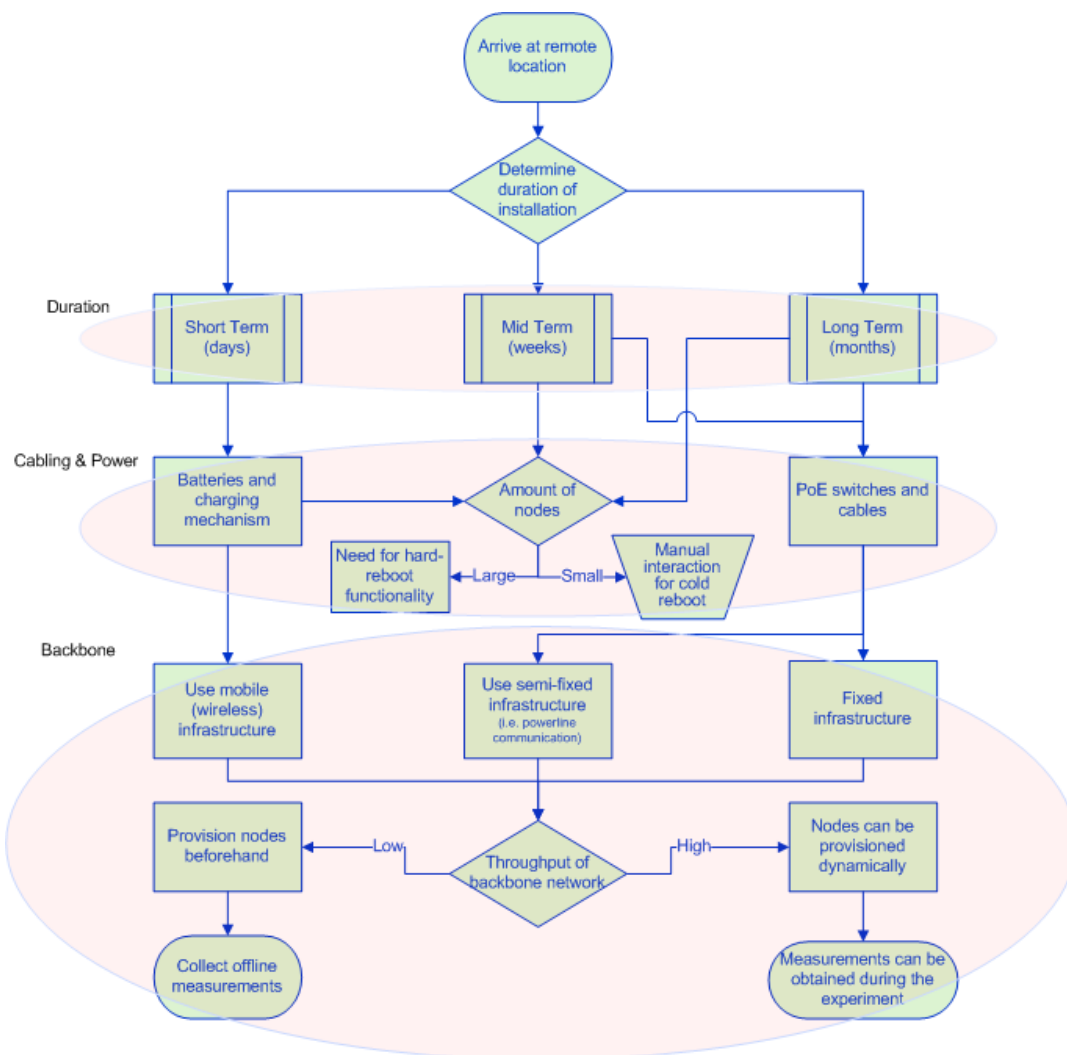


Figure 10: Deployment strategy flowchart

5.1 Deployment duration

5.1.1 Short term

Short term installations only span a couple of hours, days at most. They typically have the advantage that an operator will always be near and that the amount of nodes in the experiment will be relatively small.

5.1.2 Mid Term

These installations will span several weeks, might consist of more nodes than short term tests, and preferably run as automated as possible.

5.1.3 Long Term

Long term installations last months, typically contain considerably more nodes than short term test and run autonomously, without an operator needing to perform manual actions.

5.2 Cabling and power

Providing power to the WISHFUL boxes is essential. It can be done using fixed cabling: UTP and AC wires, or UTP and PoE (useful short-, mid- and long term tests) or, for a smaller amount of nodes it can be done using a battery pack.

5.2.1 Power toggling

Depending on the experiments ran on the mobile testbed and the amount of components (and nodes), there is need for a 'hard reboot' mechanism (cold reboot).

When running stable code, for instance when the experiment only consists of measuring, or when the scale of the experiment is sufficiently small to perform manual operations, this additional functionality may be omitted. But, when the testbed has to run experiments that are still under development, or use code still in testing phase, there is need for 'hard reboot'-functionality. Especially when there are a lot of nodes involved, making manual interaction a time consuming effort.

Hard reboot functionality can only be realised by interrupting the power supply to the WiSHFUL box. Depending on the infrastructure chosen, the mechanism to trigger can vary:

- When using PoE power, interacting with the switch can manipulate the power.
- When using battery or AC power, an intermediate component will have to trigger the power.

To achieve the second bullet, the backbone solution needs to be capable of triggering the hardware IO-pins, which in turn can relay electrical current. More details can be found in section 5.3.1.

5.2.2 Powering nodes using batteries

Powering nodes via battery packs implies having a wireless backbone network, as the node will be fully decoupled (facilitates better positioning) and will have no wired link to the EMS. Thus, the node will need a sufficiently powerful WiSHFUL backbone (see section 5.3 for more details).

Having a battery pack subsequently implies having a charging mechanism. The coupling interface between the WiSHFUL box and the flight-case has to be rigid and, ideally, capable of both transferring a typical 19V working current and an Ethernet connection.

The physical contact can be made using the pogo-pins (golden spring contacts) described in (D2.1). However, these pins are rather thin and might break when transporting the flight case. These pins are now used to charge external batteries of the robots (part of the controlled mobile nodes of the IMINDS w-iLab.t, see the FP7 OpenLab project and Figure 11), in a clean and controlled environment and they do not experience much mechanical stress as robots dock very slowly and gently.



Figure 11: Pogo pins on a mobile node in IMINDS w-iLab.t

A more rigid alternative solution could be one similar to the charging mechanism on a battery operated drill (see Figure 12). There is no need for extending spring contacts in the flight case, a rigid connection suffices. A handy feature would be a connection that is capable of locking the WiSHFUL boxes in place while they are being transported or provisioned.



Figure 12: Typical connection for battery operated power tools

An Ethernet connection can be provided when the deployable nodes are inside the flight case using PLC devices (PowerLine Communication). COTS PLC products can be modified to run over a 19V transmitter (as proven in FP7 OpenLab, also visible as white boxes in Figure 11). An architecture using a PLC solution is drawn in Figure 13.

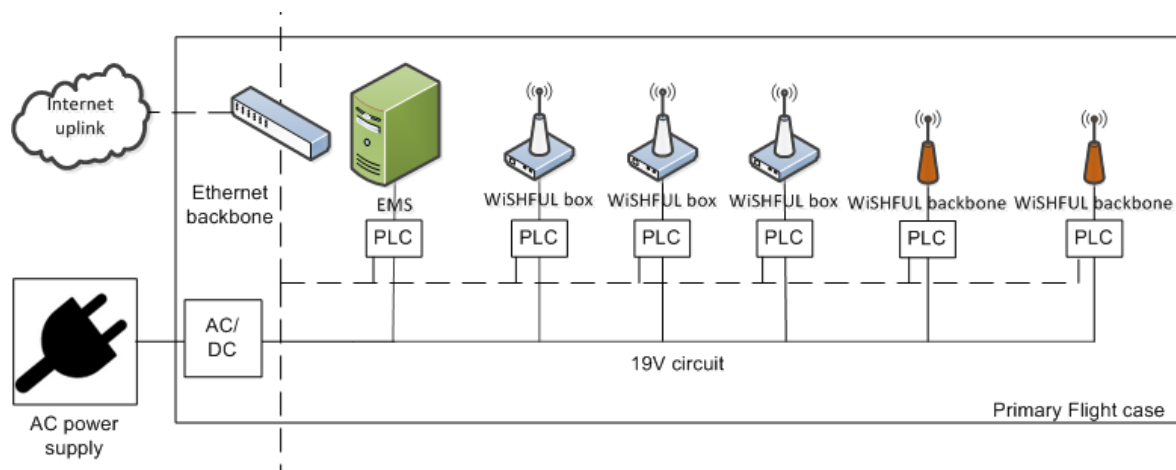


Figure 13: Configuration of battery powered devices inside a flight case, or PLC setup

5.2.3 Power over Ethernet

When the setup has to remain operational for several weeks or longer, batteries are not practical anymore. There might be need for a fixed cable that links the WiSHFUL boxes with the EMS (Experiment Management Server) for re-provisioning of nodes, high bandwidth measurement collection and obviously, power.

All this can be solved by using power over Ethernet. The IEEE 802.3at-2009 PoE-plus can deliver up to 25W of power and is sufficient to power a node (see 5.4). The cables can be up to 100m in length without amplification, implying large coverage.

The need of having an extra component serving as WiSHFUL backbone also becomes redundant. The PoE switch can power cycle the nodes when needed (see 5.2.1), and measurements can be transmitted over the Ethernet link.

5.2.4 Using AC wires and fixed infrastructure

If the tests are located in a building where fixed cabling is already pre-installed, the experiment can make use of this infrastructure. It can provide both AC power and a network connection.

For power cycling the nodes an extra component is needed (see 5.2.1).

If only AC power is available (e.g. a home environment or large warehouse), COTS PLC's can be plugged in the power outlets to provide a network connection to the various end points. Note that these PLC devices need to be unmodified and preferably of the same manufacturer and type.

5.3 Backbone network

The WiSHFUL backbone network comes in various forms, depending on their complexity, cost and partner effort. This infrastructure mainly acts as experiment control network and could also be responsible for real-time measurement gathering. The available functionality highly depends on the throughput of the backbone network. In case of high bit-rate, it is possible to collect measurement data and provision (flash or partial upgrade) WiSHFUL boxes and other devices under test (i.e. sensor nodes, routers or smart-phones). In case of a low bit-rate backbone, only basic experiment control will be available.

5.3.1 Power control functionality

As described in 5.2, the backbone network can be used by the EMS to forcefully toggle the power of a crashed node. One method to achieve this is via hardware IO-pins. These can be toggled in software and will trigger a relay, transistor or FET to (dis-)connect an electrical circuit.

Using these pins the EMS can forcefully cold reboot a testbed node. This has proven very useful in practice, when for example a warm reboot is impossible because a node is not accessible remote due to i.e. wrongfully disabled Ethernet interfaces, software crashes, boot-issues...

Power control PCB's (named Chassis Manager) were developed in FP7 OpenLab, and are used to cold reboot nodes on top of robots in IMINDS w-iLab.t.



Figure 14: Power control (Chassis Manager) card on a robot in w-iLab.t

5.3.2 Wi-Fi cards in WiSHFUL boxes

The most trivial solution to a backbone node is to use a second Wi-Fi card built into the WiSHFUL boxes. They can be set up on a completely different channel than the one being experimented on. Using simple Ad-hoc or infrastructure Wi-Fi networks, one can establish a reliable communication network with the experiment nodes and the EMS.

An obvious disadvantage is that this interface or ISM-band cannot be used in the experiment, as it will generate interference and influence the experiment, nullifying the results.



Figure 15: Use the second Wi-Fi card

5.3.3 Use of external USB sensor nodes

A mesh of USB-powered sensor nodes can be used as a simple out of band, low-bit-rate backbone network to provide a very simple experiment control interface.

The sensor nodes can communicate via the serial USB-port through a proxy-application that translates the commands into experiment control instructions. (E.g.: start application X with parameters A and B).

This solution is much more complex: a reliable firmware with appropriate network and MAC-stack has to be implemented on the embedded device. The low bit-rate does not allow collecting measurements, making this a second disadvantage.

However, the sensor node will have hardware pins enabling hard reset functionality. Also the radio used on the embedded device can be chosen to work completely outside of the ISM band (i.e. 868 MHz), so it cannot interfere with Wi-Fi experiments.

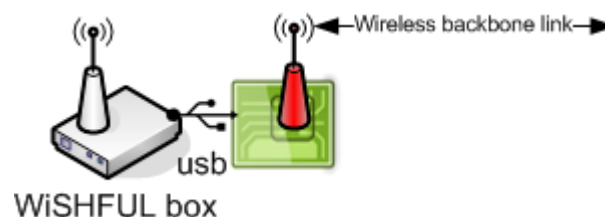


Figure 16: Use a wireless USB sensornode

5.3.4 Separate wireless backbone node

As a last alternative, a separate wireless node can be used to provide network access. The node would resemble a hardware device such as a wireless router. The connection to the WiSHFUL box would preferably be made using Ethernet. Hardware IO-pins on the device to toggle power would be a plus.

A disadvantage could be the dynamic configuration of these devices. The interface they provide should be easy to script and remotely configure.

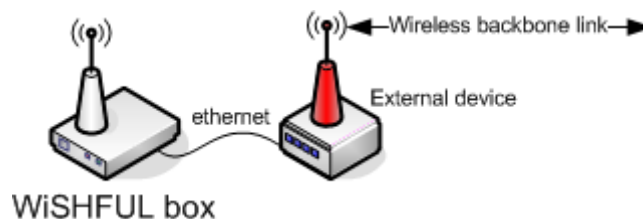


Figure 17: Use a separate node

5.4 Solution Under Test

The SUT has to comply with two fundamental requirements:

- It is power supply should not exceed the maximum available power (as described in 5.2), this should be either: AC-power, the maximum power throughput of 30 watts for PoE+ and the maximum capacity of the batteries.
- It has to have a network connection (or use a proxy node that has one, e.g. sensornode with an embedded PC) in order to provision, configure and control the resource. Ideally it is also connected to the WISHFUL backbone network to transmit measurement data.

If the SUT (Solution Under Test) completes these power and experiment control requirements it can be plugged into the testbed.

When testing sensor nodes, it is imperative to have a parent device to not just power the embedded system but also to gather measurements and provide the necessary input to bootstrap the experiment.

Some candidates for nodes are:

- Intel NUC [7] has plenty of USB ports so one can attach different technologies to it (Zigbee transceiver, sensing devices, etc...). They can be powered using PoE and batteries, and don't consume a lot of power. They also have a powerful i5 or better CPU.
- Zotac ZBOX [8] has the same characteristics as a NUC, but is slightly bigger so less of them fit in a flightcase.
- ALIX devices [9]: highly efficient barebone PC, but with low CPU power. Typically runs Voyage Linux. Ideal for attaching sensor nodes.
- Routers capable of running OpenWRT [10].
- "Super Nodes" or "smart USB hubs" [11]: ideal for running Serial Forwarder to control simple sensor nodes.
- The Gateworks Laguna [12] a cost-effective SBC range based on ARM11 dual core Cavium Econa processor.
- The Gateworks Ventana [13], a newer line of SBC solutions based on ARM Cortex-A9 quad core Freescale i.MX6 processor.

6 Conclusion

In this deliverable we have provided the specification for the portable testbed implementation that will be followed during the project. A description of the portable testbed controller is given in the document. The functionalities expected from testbed controller are also outlined.

One of the main features of the portable testbed is its ability to automatically setup the backbone network after all nodes are switched on. To support this feature, two possible technologies, namely

IEEE 802.11s and the NCENTRIC mesh solution based on OLSR were investigated. Although, WiFi is currently used as the technology for implementing the backbone network, we would like to still have a possibility to switch it if needed to different technology. Since the IEEE 802.11s standard operates in Layer 2 of ISO/OSI model this cannot be achieved. Hence the OLSR implementation modified by NCENTRIC, operating in Layer 3 (Network layer), is used as primary technology.

This deliverable also provides a specification for the Wireless Control Channel and defines five different flow types used to control the SUT and backbone networks. All these flows are sent over a single wireless interface and are competing for resources. The QoS requirements were defined for each flow and solutions were discussed that enable prioritizing certain flows. The basic rule is to prioritize real-time control flows over the elastic data flows. Some possible optimizations of the Wireless Control Channel were also defined. Since the same UPIs are used to control the backbone and SUT networks, protection mechanisms were discussed that prevent an experimenter (testbed user) to change the configuration of the backbone network.

Finally, the portable testbed specification for packing and deployment are included in this deliverable. Several solutions of providing power for SUT and backbone nodes are investigated. The few possible scenarios of experiments expected to be performed with portable testbed are described. Requirements that SUT nodes have to fulfil to be part of portable testbed are defined and few possible candidate devices are presented.

7 References

- [1] Jerome Henry, Marcus Burton, “802.11s Mesh Networking Whitepaper”, November 2011
- [2] Olaf Kirch, Terry Dawson, “Linux Network Administrators Guide”. [Online]. Available: <http://www.oreilly.com/openbook/linag2/book/ch09.html#X-087-2-FIREWALL.TOS.MANIPULATION>
- [3] Martin A. Brown , “Traffic Control HOWTO”. [Online]. Available: <http://tldp.org/HOWTO/Traffic-Control-HOWTO/classless-qdiscs.html>
- [4] Bert Hubert, “Linux Advanced Routing & Traffic Control HOWTO”. [Online]. Available: <http://lartc.org/howto/lartc.qdisc.classless.html>
- [5] LAN/MAN Committee of the IEEE Computer Society, “IEEE Std 802.11e-2005 Amendment 8: Medium Access Control”, Nov. 11 2005
- [6] Eldad Perhia, Robert Stacey, “Next Generation Wireless LANs, Second Edition”, Cambridge University Press 2013
- [7] MiniPC: Intel NUC Product Website. [Online]. Available: <http://www.intel.com/content/www/us/en/nuc/overview.html>
- [8] ZOTAC ZBOX, Product Website. [Online]. Available: <http://www.zotac.com/us/products/mini-pcs/zbox/product/zbox.html>
- [9] PC Engines ALIX, Product Website. [Online]. Available: <http://www.pcengines.ch/alix.htm>
- [10] OpenWRT Project Website. [Online]. Available: <https://openwrt.org/>
- [11] CREW Project Website. [Online]. Available: <http://www.crew-project.eu/twist>
- [12] GATEWORKS Laguna GW2388-4, Product Website. [Online]. Available: <http://www.gateworks.com/product/item/laguna-gw2388-4-network-processor>
- [13] GATEWORKS Vetana GW5100, Product Website. [Online]. Available: <http://www.gateworks.com/product/item/ventana-gw5100-network-processor>