Deliverable	D3.1
Title	Definition of the performance monitoring mechanisms
Date	December 2016
Milestone	MS4
Authors	Jose Saldana (UNIZAR, Editor), Luis Sequeira (UNIZAR), José Ruiz-Mas (UNIZAR), Julián Fernández-Navajas (UNIZAR), Ali Arsal (AirTies)
Reviewers	Faycal Bouhafs (LJMU), Michael Mackay (LJMU), Qi Shi (LJMU)
Approved by	General Assembly
Dissemination	Public

Wi-5

What to do With the Wi-Fi Wild West
Funding scheme: H2020-ICT-2014-1
Grant number: 644262

Horizon 2020
European Union Funding
for Research & Innovation

Abstract

The *What to do With the Wi-Fi Wild West* H2020 project (Wi-5) combines research and innovation to propose an architecture based on an integrated and coordinated set of smart Wi-Fi networking solutions. The resulting system is able to efficiently reduce the interference between neighbouring Access Points (APs) and provide optimised connectivity for new and emerging services. The project approach is developing and incorporating a variety of different solutions, which are being made available in academic publications, in addition to other dissemination channels.

This document includes the specification of the performance monitoring mechanisms of the Smart Access Point Solutions, which are being deployed within WP3 of the Wi-5 project. After the Literature Review, the Wi-5 performance monitoring mechanisms are presented in detail. This includes monitoring the information of the STAs (wireless stations) connected to each AP, which is averaged and sent to the central controller. In addition, the parameters which have to be monitored during the handoff of a STA between APs in different channels are reported.

The monitoring of other APs in the neighbourhood has also been included in the framework, being able to send special beacon frames that are captured by other APs, and reported to the central controller. Finally, the possibility of a timely detection of real-time flows of emerging services appearing in the network is also included, in order to let the controller make better radio management decisions.

The main innovations are the possibility of performing fast handoffs between APs in different channels, employing virtual APs. For that aim, the monitoring functionalities have to work in a timely and accurate manner. In addition, the separation of the Data and Control planes permits us to integrate the concept of SDN within the solution.

Wi-5 Consortium

	Liverpool John Moores University	2 Rodney Street Egerton Court Liverpool, L3 5UK United Kingdom
	Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek	Anna van Buerenplein 1 2595 DA Den Haag Netherlands
	Universidad de Zaragoza	Calle Pedro Cerbuna 12 Zaragoza 50009 Spain
	AirTies Kablosuz İletişim San ve Dış. Tic. A.Ş	Mithat Uluünlü Sokak No:23 Esentepe, Şişli P.K 34394 İstanbul Turkey

Contents

List of Figures

Glossary

AP	Access Point
BSSID	Basic Service Set Identifier
CPU	Central Processing Unit
CSA	Channel Switch Announcement
DHCP	Dynamic Host Configuration Protocol
Diffuse	DIstributed Firewall and Flow-shaper Using Statistical Evidence
DOI	Digital Object Identifier
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LVAP	Light Vırtual Access Point
MAC	Media Access Control
ML	Machine Learning
MMORPG	Massively Multiplayer Online Role Playing Game
MPDU	MAC Protocol Data Unit aggregation
MSDU	MAC Service Data Unit aggregation
PBS	Periodic Background Scan
QoE	Quality of Experience
QoS	Quality of Service
SDN	Software-Defined Network
SDWN	Software-Defined Wireless Network
SOHO	Small Office / Home Office
SSH	Secure Shell
SSID	Service Set Identifier
STA	Wi-Fi Station
TCP	Transmission Control Protocol
Tx/Rx	Transmission / Reception

UDP	User Datagram Protocol
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol
WEKA	Waikato Environment for Knowledge Analysis
WLAN	Wireless Local Area Network

Executive Summary

This document will provide a definition of the mechanisms that will be included in the Wi-5 APs with the aim of monitoring their performance in terms of interference level; channel load seen at each available channel in the interested frequency bands; the number of clients associated with an AP; the amount of data passing through each AP; and the type of service of the automatically detected flows. The interfaces to be used by the AP features in order to get this information will also be defined. These are of crucial importance, as accurate and timely monitoring information is required in order to make good resource management decisions.

First, a Literature Review is presented, covering the monitoring of the wireless environment in IEEE 802.11 APs, with a special focus on the problems that appear when Light Virtual APs (LVAPs) are used. The problem of real-time detection of flows belonging to different services is also addressed.

The main section of the deliverable presents the monitoring mechanisms in detail. It starts with a subsection explaining their role within the architecture, also highlighting their importance. Next, a basic description of the Wi-5 implementation is provided, including the software and hardware elements being used. The original platform (called *Odin)* and the improvements that have been required are next summarised. One of the key improvements consists of the addition of an auxiliary interface, which can be used for monitoring purposes, thus avoiding the need for interrupting the service to the connected STAs.

From here, the different monitoring features are explained in detail:

- The gathering of the information about the STAs connected to each of the Wi-5 APs. This information is collected by the APs, and forwarded to the controller, which can then have a global vision of the status of the network.
- The monitoring information that is required in order to perform fast handoffs between APs, which also include a channel switch of the STA.
- The features included in the controller and the agents to monitor the interference caused by other APs in the neighbourhood. This includes the sending of special beacons and the detection of their power levels, using the auxiliary interface. This information is then forwarded to the controller, which can therefore build different "interference maps" between APs. They can also be used for measuring interference levels between APs in adjacent channels.
- The integration of the automatic detection of real-time flows in the Wi-5 scheme. A centralised approach has been followed, in which the detector is integrated within the router, and a *flow report* machine is in charge of taking the packets, grouping them in flows and sending this information to the controller.

The document ends with our Conclusions, surveying the presented work. The main innovations are the possibility of performing fast handoffs between APs in different channels and employing virtual APs. For that aim, the monitoring functionalities have to work in a timely and accurate manner. Moreover, the integration of the service detection tools within the global scheme is also an innovative approach used by Wi-5.

1   Introduction

1.1   Background to project

The last few years have witnessed a significant increase in the use of portable devices, especially smartphones and tablets which, thanks to their functionality, user-friendly interfaces and affordable prices, have become ubiquitous worldwide. Most of these devices make use of IEEE 802.11 wireless standards, commonly known as Wi-Fi.

Given this increasing demand, Wi-Fi is facing mounting issues of spectrum efficiency due to its utilisation of non-licensed frequency bands, so improvements continue to be added in order to enhance its performance. For example, as Wi-Fi saturation increases in congested scenarios such as business centres, malls, campuses or even whole cities, the interference between these competing Access Points (APs) can negatively impact the experience of the users.

At the same time, real-time services with tight latency constraints are becoming ubiquitous. For example, VoIP services such as Skype are now very popular. Furthermore, some instant messaging applications (e.g. *WhatsApp*) also include VoIP features. Finally, online games are no longer exclusive of high-end PCs, but many have been ported to tablets or even smartphones. Therefore, these real-time services share the same connection with "traditional" applications, such as e-mail, video download or Web browsing, but have different Quality of Experience requirements.

In addition, the availability of these services in mobile devices has a consequence: whereas the mobility of a user with a laptop can be considered as *nomadic* (i.e. the user may move, but he/she will stay for a long time in the same place), smartphone and tablet users may also walk while using these real-time services.

The *What to do With the Wi-Fi Wild West* H2020 project (Wi-5) combines research and innovation to propose an architecture based on an integrated and coordinated set of smart solutions able to efficiently reduce interference between neighbouring APs and provide optimised connectivity for new and emerging services. Cooperating mechanisms are being integrated at different layers of the protocol stack with the aim of meeting a demanding set of goals such as seamless hand-over, reduced congestion, increased throughput and energy efficiencies. The project is developing a variety of different solutions, which are being made available in public software repositories, and explained in academic publications, in addition to other potential dissemination channels for industrial exploitation and standardisation.

1.2   Scope and structure of the deliverable

This document will provide a definition of the mechanisms that will be included in the Wi-5 APs with the aim of monitoring their performance in terms of interference level; channel load seen at each available channel in the interested frequency bands; the number of clients associated with an AP; the amount of data passing through each AP; and the type of service of the automatically detected flows. The interfaces to be used by the AP features in order to get this information will also be defined. Therefore, the deliverable summarises the work performed in Task 3.1 "Performance monitoring," which finishes at MS4, i.e. the end of the second year of the Wi-5 project (31st Dec 2016).

As shown in Figure 1, monitoring functionalities have been included in different elements of the Wi-5 architecture: they are present in the Wi-5 controller and the Wi-5 APs but not in the STAs. The

monitoring information provided by the APs is crucial in order to let the algorithms, running in the controller, make adequate network configuration decisions.
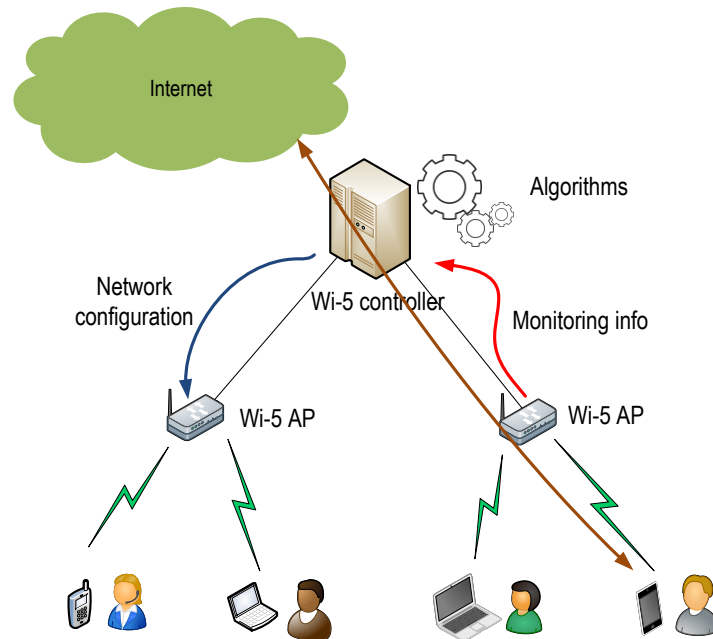
Figure 1: Wi-5 elements: controller and APs

First of all, we will summarise the modifications included in the APs in order to make them able to gather the required information. This includes the use of an auxiliary interface on each AP. Once this has been explained, the functionalities able to monitor the STAs[1] associated to the AP are described in detail. Next, the mechanisms for monitoring other APs in the neighbourhood are reported, and finally the inclusion of tools for automatic detection of the services is explained.

1.3 Relationship to other deliverables

D3.3: This document is closely related to D3.3, Specification of Smart AP solutions version 2. It is obvious that the Smart AP solutions rely on the information being gathered by the monitoring tools. If this information is provided in a timely and accurate manner, the functionalities making use of them will be able to provide a better user experience.

D4.2: The information from the monitoring tools is required by the Cooperative Functionalities, being deployed in WP4, as explained in detail in D4.2, Specification of Cooperative Access Points Functionalities version 2. These functionalities are tightly related with the Smart AP Solutions, since they have to run in an integrated and seamless way, in order to provide the desired performance. It can be said that having good and timely information of the wireless environment and the services being used is required by the optimisation algorithms to run flawlessly and to make the correct decisions regarding channel assignment, load balancing, etc.

D2.5i and D5.1: This document is also related to D2.5i, which describes the latest version of the global Wi-5 architecture. Finally, there is a relationship with D5.1, Testbed description and definition of the tests, as it includes a description of the platforms for the evaluation tests.

---

[1] We will use the term "STA" to refer to an end device (e.g. a mobile phone, a tablet, a laptop, etc.), as it is common in the literature of this topic.

2   Literature Review

The concept of Software-Defined Wireless Networks (SDWNs) is being leveraged in the work being carried out within the Wi-5 project. In this document, we focus on our approach for monitoring the wireless environment and the detection of services. The literature presented in this section is related to Wi-Fi monitoring while other topics, e.g. Radio Resource Management in IEEE 802.11 [1], [2], [3], are summarised in deliverable D3.3.

2.1   Monitoring the Wireless Environment in SDWNs

The IEEE 802.11 standard establishes some approaches for a client to scan nearby APs. One common option is to establish some time intervals in which the AP can switch to other channels and scan. Two main approaches exist. Passive scanning can be used, in which the station just sets the desired channel and waits for the reception of beacons from other APs in the neighbourhood. In contrast, active scanning consists of sending a probe request, and waiting for a probe response from the AP [4].

There are many reasons why this is necessary. For example, in wireless scenarios where APs are distributed in different channels according to a centralised frequency planning approach (e.g. airport, business centre), if an STA moves away from the AP it is associated with, it will not be detected by APs in the neighbourhood since they will be on other channels. Therefore, periodic scanning should also be performed for detecting approaching STAs. For Wi-5, the use of multichannel LVAPs in SDWNs also makes it necessary to establish some periods for scanning on other channels in order to detect new clients. In [5] an optimisation method for the scanning period was proposed by making an active scanning only when needed, instead of using PBS (Periodic Background Scan) which results in a significant overhead.

2.2   Service Detection

Considerable research effort has been dedicated to the automated and real-time classification of IP traffic in recent years. This enables the seamless integration of QoS, since the detection of a service type allows operators to more easily associate it with some quality requirements. For example, in [6], an automated system for QoS control was proposed for an application with tight real-time requirements (an online game in this case). Another use of automatic detection is the prevention of intrusions, which was first proposed in [7].

Detection of flows can be made on a high level based on UDP or TCP port numbers, but this can present problems of a different nature: for example, certain applications may employ different ports, and some regulations may even prohibit the inspection of the content of the packets. Statistical classification is also being used for this, based on key features of the flows, such as packet size or inter-packet time patterns. Combined with machine-learning (ML) techniques, this can present many advantages. The approach followed here would include two stages: first, the algorithm has to be trained and then the devised rules can be employed for performing the classification. A number of proposals based on different algorithms have been presented for this. In [8], traffic with similar observable properties was clustered into different application types, and the ML algorithm was able to separate traffic into a small number of basic classes. In [9], the nearest neighbours were used, and Linear/Quadratic Discriminant Analysis ML algorithms were employed to map different network applications to predetermined QoS traffic classes. The work in [10] used an unsupervised Bayesian classifier to find the best cluster set

from the training data and a number of applications were studied, showing that some separation between them could be achieved.

In [11] a tool called *Diffuse,* able to identify real-time traffic patterns was presented. It was employed for the unsupervised detection of real-time services with tight delay requirements (VoIP and online games), obtaining an accuracy of about 90%. Diffuse uses the concepts of *recall* (percentage of members of a class correctly classified as belonging to that class among all members of the class) and *precision* (percentage of those instances that truly belong to a class among all those classified as belonging to that class).

2.3   Wi-5 innovation related to performance monitoring

Some gaps in the existing work have been identified, which are part of the Wi-5 objectives.

Radio monitoring: The need of integrating LVAPs in a multichannel and centralised scheme makes it necessary to have a mechanism which performs scanning and, at the same time, does not leave current users unattended. Towards that aim, the use of a secondary wireless interface on each AP has been proven as a feasible solution, as it does not significantly increase the cost per AP (the cost of a wireless interface is minimal). A basic scheme for this is shown in Figure 2, where two APs are able to perform a seamless handoff, thanks to the auxiliary interfaces they include: AP_2 is in channel B, but a central controller tells it to set its auxiliary interface in channel A, so it can hear the STA approaching and prepare the handoff subsequently.
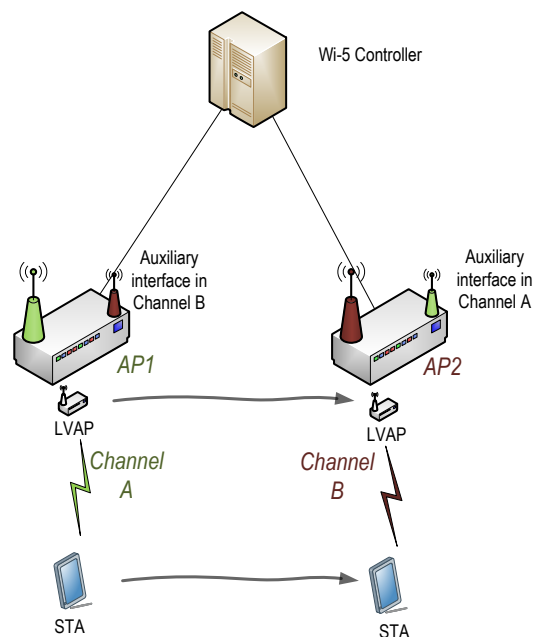
Figure 2: Scheme of two Wi-5 APs in different channels using auxiliary interfaces

This has required a modification of the scheme of the AP, and some improvements in the software tools running in it. In addition, the handoff scheme has been improved, including a significant implication of the monitoring tools. This process is described in detail in deliverable D3.3.

Improvement of the communication between the controller and the APs: The separation between the Data Plane and the Control Plane allows us to separate the data of the users from the control traffic

required for communication between the APs and the central controller. It can be said that we have now fully integrated the concept of SDWN within the solution initially proposed in [12].

Integration of service detection tools in the WLAN: Another field where Wi-5 presents innovative results is the integration of the service detection tools within the global scheme. This allows the central controller to know the services that are running on each of the APs, with a special focus on those with real-time constraints. This paves the way to the development of novel resource management algorithms, able to take into consideration the nature of the flows being served by each AP. As an example, the controller may try to avoid the coexistence of a real-time flow with a number of video downloads in the same AP, so it may make smarter load balancing decisions if this information is considered as an input to the algorithm.

3   Wi-5 Performance Monitoring Mechanisms

3.1   Role of Performance Monitoring Mechanisms in the Wi-5 architecture

The design of the Wi-5 architecture relies on the separation of the control and data planes in Wi-Fi APs, based on an SDN approach. This approach is being followed in order to have a single point where all the control operations can be integrated and coordinated. The most important entity is the Wi-5 controller, having a global view of the network, and being able to run different algorithms for optimising the network. Therefore, certain functionalities (e.g. load balancing, channel selection) run as applications on top of the controller (see Figure 3), which include new functionalities in order to interface with the Wi-5 tools.
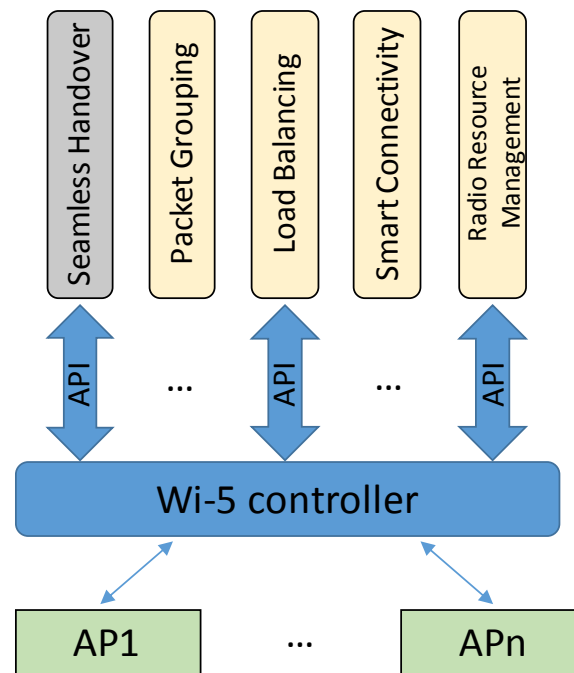
Figure 3: Scheme of the Wi-5 architecture design

In order to make it possible for the controller to manage all the APs, new functionalities are included in each of them, i.e. their internal switch is an OpenFlow switch, and a Wi-5 agent is added, in order to interact with the Wi-5 controller.

One of the aims of the Wi-5 Project is to make it possible for a set of APs to support real-time applications (e.g. VoIP, online games) with quality. This includes resource management algorithms that take into account the nature of each flow, and its coexistence with other services. Therefore, it is necessary to include a functionality able to detect the real-time flows and to report their existence to the controller.

In addition, seamless handovers between APs are not only required for supporting user mobility, but also for the optimisation of radio resources. The main reason for the decision of using a solution based on LVAPs, is that it is suitable for supporting load balancing, dynamic channel and power configuration and other interesting features in multi-channel WLANs. There is an additional advantage: in regular Wi-Fi, the handover is usually triggered by the user device, so it is not controlled by the network and it may require 1 or 2 seconds to perform. In contrast, the SDWN solution allows the network to control

the mobility and to select the best moment for the handover, which is very fast, in the order of 20-100 milliseconds [13].

Figure 4 shows the scheme of the Wi-5 monitoring entities, indicating the place where they are implemented, and how they interact between them. As can be seen here, monitoring functionalities are running in each Wi-5 AP, but a centralised approach is required in some cases (e.g. for detecting services with real-time constraints), so a monitoring module is also included in the Wi-5 controller. This is important because the monitoring data is also fundamental to making network control decisions. For example, decisions about resource management (channel selection, power control, load balancing, and packet grouping) are made by the central controller, but the configuration will take place at the AP. The existence of services with real-time constraints also affects the frame aggregation: it should be taken into account that aggregation introduces new delays (required to gather a number of packets that will travel together) that can affect the user experience [14] [15].
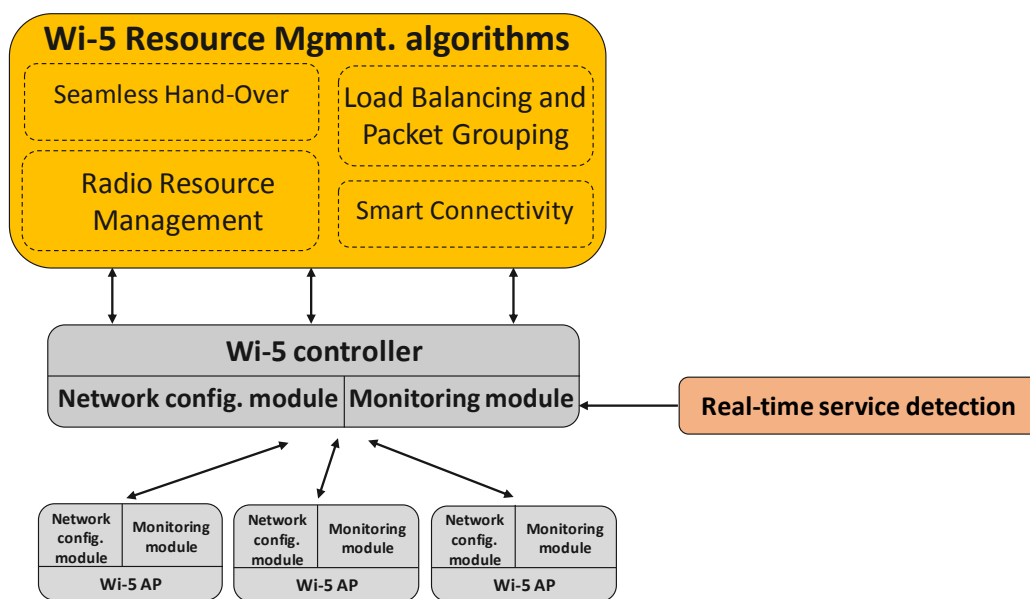
Figure 4: Scheme of the Wi-5 entities and the place where monitoring is performed

The role of the monitoring tools is to measure the interference level and the channel load seen at each available channel in the relevant frequency bands (i.e. 2.4 and 5 GHz). In addition, the number of clients associated with an AP and the amount of data passing through each AP will be considered. Finally, automatic detection tools, which can be based on machine-learning [11], have been integrated in order to detect and identify different kinds of real-time services (mainly VoIP and online games).

Figure 5 illustrates the flows that are exchanged between the different elements of the Wi-5 architecture. As can be seen here, the monitoring module on each AP communicates the measured parameters to the controller, which also receives information about the flows with real-time constraints (e.g. VoIP, games) that are present in the network. This information is useful for the algorithms running in the decision module, which produces output that are implemented by the network configuration module. It can be said that a timely and accurate monitoring information is required for the success of the whole system.
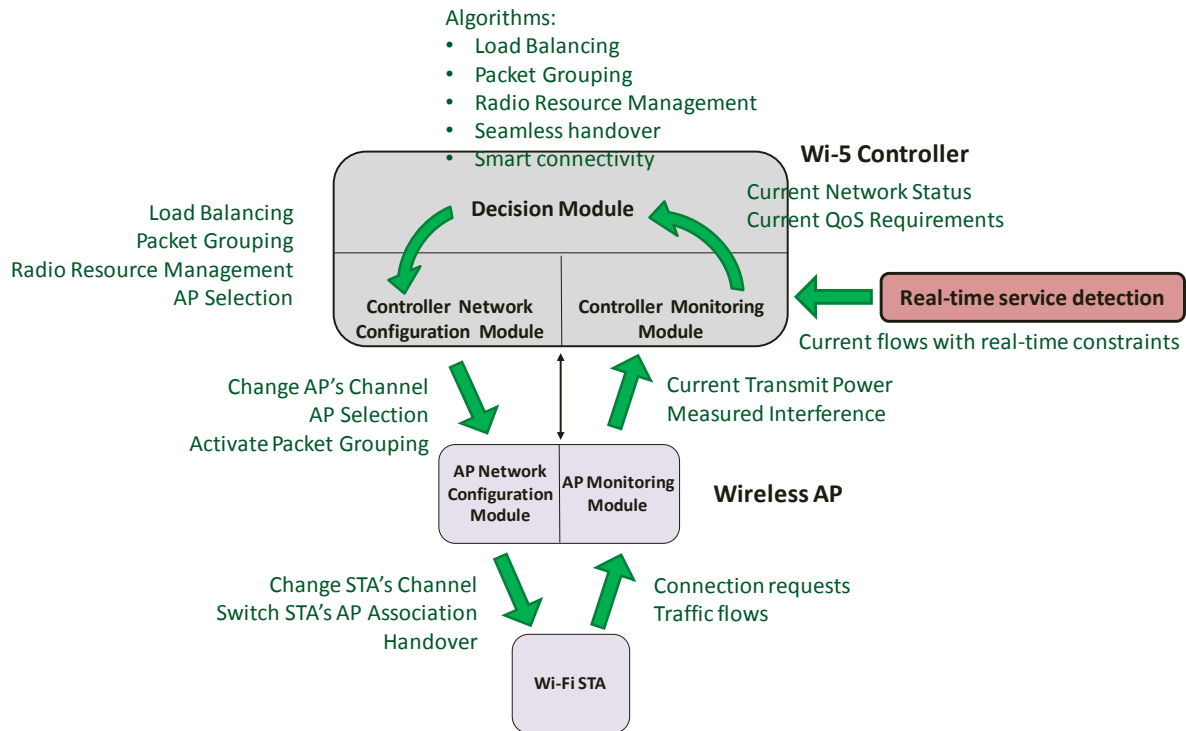
Figure 5: Information exchange between Wi-5 entities

The information gathered by the performance monitoring elements can then be combined and used to perform the resource management features and the packet grouping:

- Radio Resource Management. This includes mechanisms allowing optimised performance of the Wi-5 algorithms for: (i) Dynamic Channel Allocation, looking for an optimal distribution of the channels of the APs, in order to reduce the interference; (ii) Load Balancing, looking for an optimal distribution of the users between the APs; (iii) Power Control, trying to keep the signal level as low as possible, in order to save energy and to reduce the interference.

- Packet grouping, considering smart scheduling policies enabling a better use of the aggregation policies already included in 802.11n and subsequent versions [16]. This can alleviate the airtime inefficiency caused by the STAs requesting the shared channel before being able to send actual data. For legacy devices not including aggregation at Layer 2, multiplexing packets at Layer 3 can also be a solution [17], [18].

In the remainder of this section, we will first present a basic description of the Wi-5 implementation. Next, the functionalities added for monitoring the STAs connected to an AP will be explained, along with the monitoring tools that run during a handover between APs in different channels. Next, the method for scanning for other APs in the neighbourhood will be reported. The section finishes with a subsection about the automatic detection of real-time services.

3.2   Basic Description of the Wi-5 implementation

In this subsection we give an overview of the implementation being developed in Wi-5 and show how it is extended to support monitoring. As discussed above, Wi-5 is an LVAP-based wireless LAN solution, built as an extension of the work started in [12]. It includes a central controller, and a set of agents (i.e. the APs). A more detailed explanation can be found in deliverable D3.3.

The system runs on commodity OpenWrt APs. OpenWrt[2] is a Linux distribution for embedded devices, which provides a fully writable filesystem with package management. More than 1000 Wi-Fi devices are currently supported, many of them being low-cost APs, typical of SOHO (Small Office / Home Office) wireless scenarios.

OpenFlow is used to control the internal switch of each wireless AP in a network. For that aim, *Open Vswitch* [3] is installed, thus making the internal switch of each AP behave as an OpenFlow switch. The controller runs a Floodlight OpenFlow Controller[4] in order to manage all the switches of the APs. The resource management algorithms run as applications on top of this controller.

In addition, *Click Modular Router* [19] runs in the AP, adding a specific software module called `Odin`, which interacts with the controller when required. A virtual Linux *tap* [5] interface called `ap` is added to the internal switch of the controller (called `br0`). The wireless network interface has to run in the *monitor* mode, so its name is `mon0`.

A patch has to be added to the 802.11 driver (e.g. Atheros *ath9k*) of the AP in order to acknowledge frames with different destination MAC addresses in a single real interface.

3.2.1   Original framework and limitations

A scheme of the entities and interactions proposed in the original framework [12] (called Odin) is shown in Figure 6.
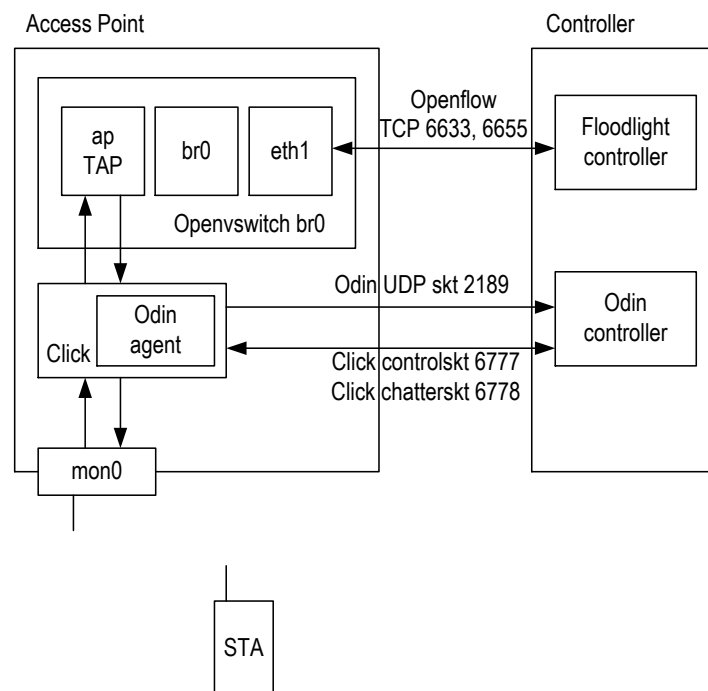
Figure 6: Entities and communications in the original framework

---

[2] OpenWrt, Wireless Freedom, https://openwrt.org/

[3] Open v Switch, http://openvswitch.org/

[4] Floodlight SDN Controller: http://www.projectfloodlight.org/floodlight/

[5] TAP enables layer-2 frame reception and transmission for user space programs in Linux.

As can be seen from Figure 6, the controller and the AP exchange signalling information which is used for monitoring and the resource management. This includes:

- OpenFlow: uses TCP ports 6633 and 6655 to control the internal switch of the AP, as it is an OpenFlow switch using *Open Vswitch*.
- *Click control socket* and *Click chatter socket*: uses TCP ports 6777 and 6778.
- Odin agent: uses UDP port 2189 in order to transmit monitoring messages to the controller.

Although the work in [9] represents a good starting point, achieving Wi-5 objectives using this framework presents several limitation:

- First, the original framework assumes that all the APs are using the same channel, as this makes it possible for an AP to hear all clients in the vicinity, even if they are associated to other APs. However, this represents a severe limitation, as it restricts the controller from implementing a centralised frequency planning for the different APs.
- Second, in the original framework the data and the control traffic share the same connection, i.e. the traffic of OpenFlow and the Click control socket use the same network interface as the data traffic.
- Finally, the available applications and algorithms in the original framework are implemented at proof-of-concept level and quite simple. For instance, the load balancing only consisted of a round robin algorithm that does not take into account any signal power parameters.

Therefore, we concluded that it was potentially a good option, but many features should be added in order to build monitoring solutions enabling resource management in real scenarios. Different functionalities are being added in order to make it capable of supporting the requirements of Wi-5. The new code is being shared in the Wi-5 GitHub repositories[6], including the Wi-5 controller (written in Java) and the Wi-5 agent (written in C++).

3.2.2 Modifications required for Monitoring the Wireless Environment

When developing our monitoring approach, the addition of an extra interface in the wireless router being used for development was deemed necessary, as shown in Figure 7. A TP-Link TL-WN722N[7] USB wireless card has been added in order to simultaneously allow both the monitoring of the wireless environment (STAs and APs in the neighbourhood) and the service to the connected STAs. It should be noted that the additional cost of this new interface is low (about 10-15€).

---

[6] Wi-5 GitHub code repository, https://github.com/Wi5

[7] The datasheet of this device is available at http://www.tplink.com/resources/document/datasheet/TL-WN722N_ds.zip [Accessed Nov 2016]

(a) (b)
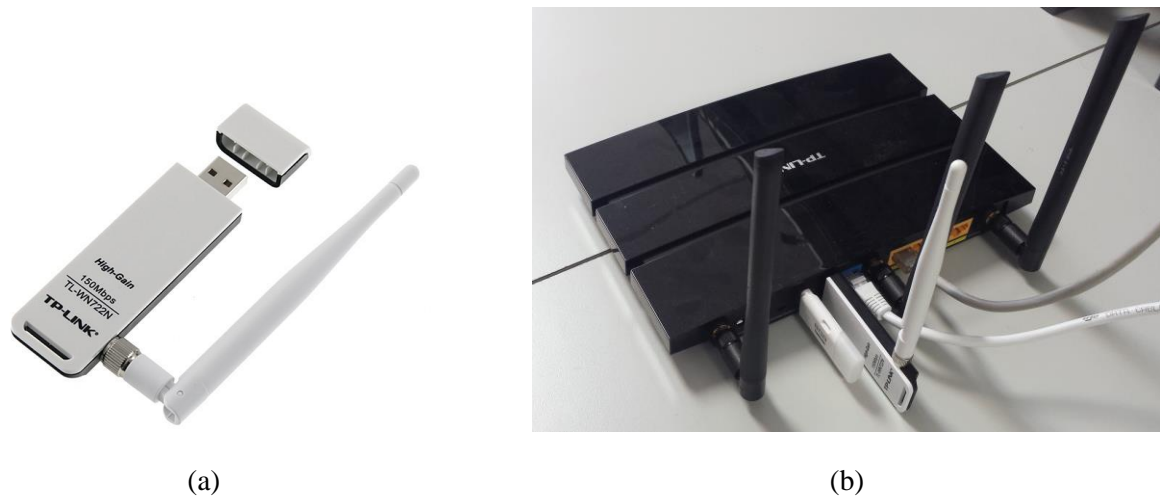
Figure 7: An additional wireless network interface (a) added to the Wi-5 AP (b)

As explained before, the main reason for this is that we are adding the possibility of each AP operating in a different wireless channel to the original framework [12]. This feature, which provides obvious benefits (e.g. a frequency planning is now possible), introduces an additional complexity in the system: in the original framework, as all the APs operated in the same channel, they were able to detect STAs associated to other APs in the neighbourhood in order to perform handoffs when required. If different channels are employed, a neighbour AP will not detect an STA in another channel.

One possibility for this was to define scanning periods in which the AP would move to another channel in order to monitor the environment [5]. However, this is not desirable as during these intervals the associated STAs would not be able to send and receive frames. Therefore, the introduction of an additional interface allows us to simultaneously serve the STAs associated in the AP channel, and to monitor the environment in other channels.

This also has some implications in the scheme of the entities and communications in *Odin*. As shown in Figure 8, two network interfaces in the monitor mode, namely `mon0` (main interface) and `mon1` (auxiliary interface), are now present, and all their traffic is monitored by the Odin agent. Therefore, it can report different events depending on the characteristics of the wireless frames. Note that the original scheme shown in Figure 6 has been modified.

When the Wi-5 agent has to communicate with the Wi-5 controller, it sends packets to the *tap* interface called `ap`, which is controlled by *Open Vswitch* (i.e. the internal switch of the AP works as an OpenFlow switch). A separation between the data and control planes has been implemented, which is defined from the OpenFlow controller: the data going to the Internet (and also the DHCP requests and responses) goes through `eth1.2`, while the control traffic uses the other interface (`eth1.1`). This includes the OpenFlow packets, in addition to the two flows of the Odin protocol: a UDP socket in port 2819, and one TCP socket in port 6777. Finally, DHCP packets are also injected to the control plane, as the Wi-5 controller has to know the IP addresses assigned to each of the STAs. This allows it to define the OpenFlow rules of each of the clients.
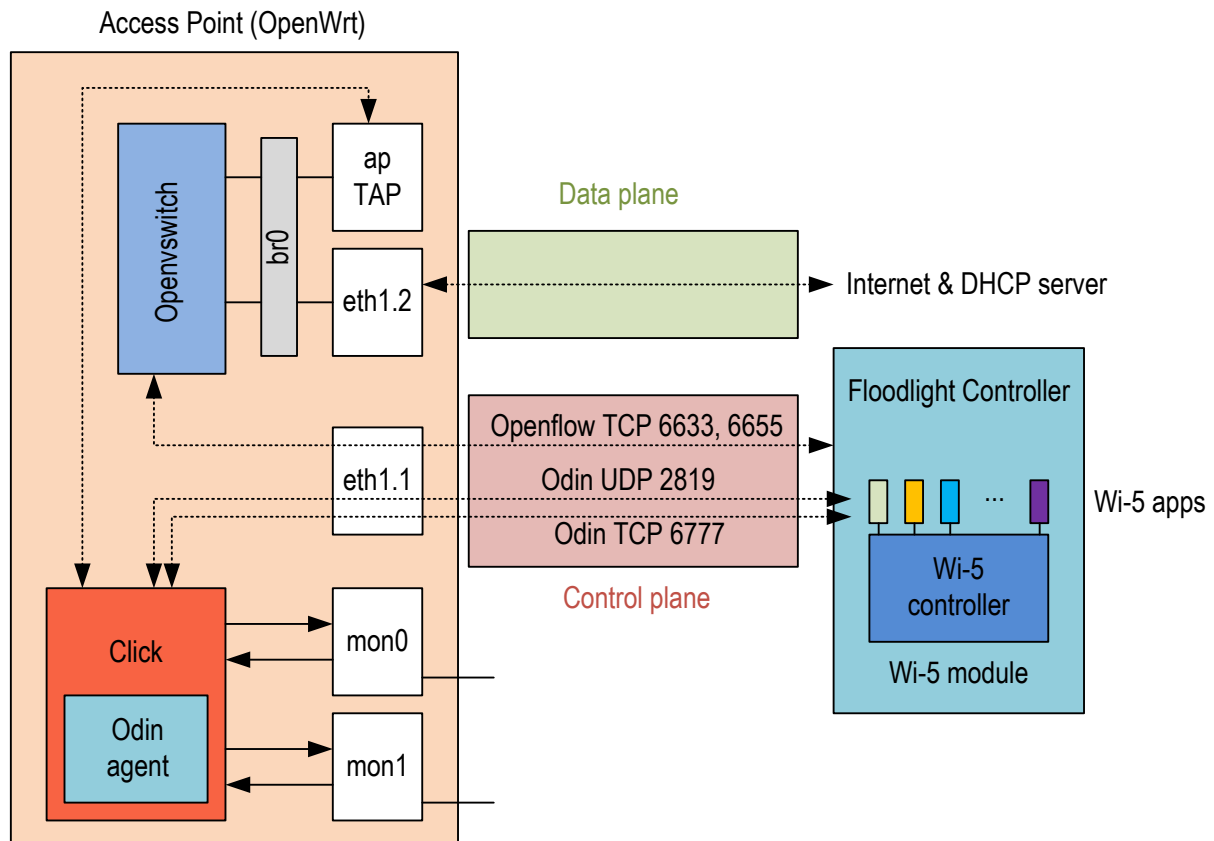
Figure 8: Modification of the scheme when adding an auxiliary wireless network interface

The scheme of the AP is also slightly modified in order to integrate the auxiliary wireless interface in a correct way, as shown in Figure 9. The wired interface `eth1.1` is now not connected to the `br0` internal Linux bridge, as it is the control interface used to communicate between the agent and the controller.
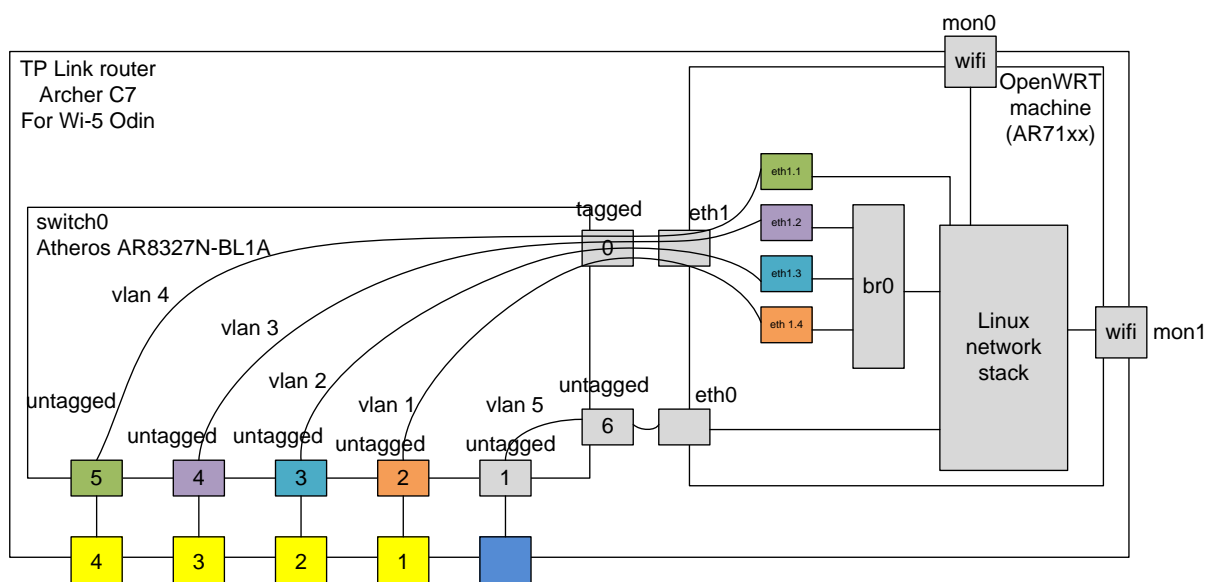
Figure 9: Modification of the AP for the addition of an auxiliary wireless network interface

Finally, it should be noted that the addition of this interface does not come with a performance penalty. As shown in Figure 10, obtained with the `top` Linux system monitoring tool[8], during normal operation of an AP with two interfaces, the CPU is only at 9%, which is only slightly above the figure obtained using a single interface.
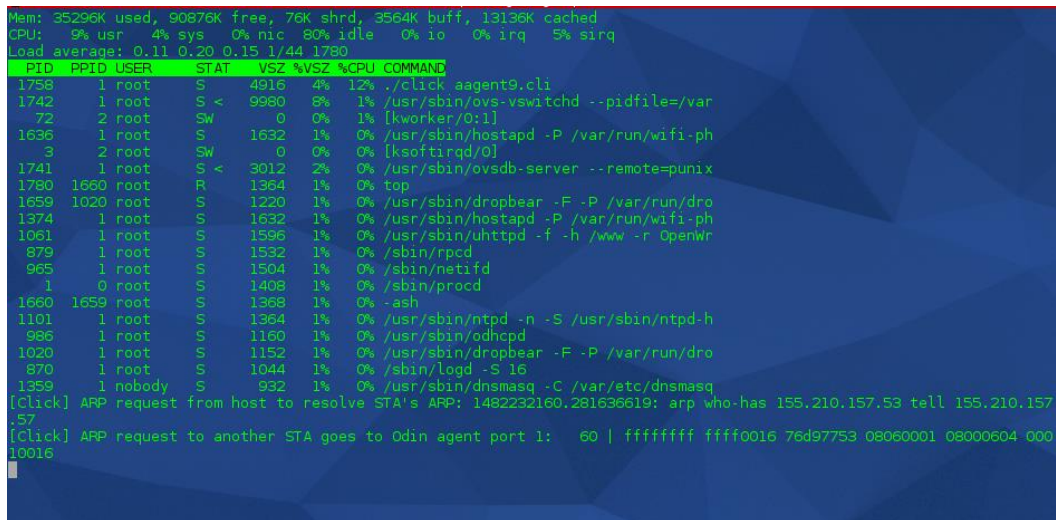
Figure 10: CPU performance while running Wi5 agent with an auxiliary interface

3.3 Monitoring the STAs connected to an AP

The Wi-5 framework includes an SDN controller able to manage a number of agents (i.e. a process running inside an 802.11 AP) which manage connectivity for STAs. In order to make informed radio resource management decisions, the controller must have updated and detailed information about the status of the radio link between each of the APs and the STAs associated to it.

Therefore, each AP has to capture different statistics for each of the STAs associated to it. The process is as follows (see Figure 11): the Wi-5 agent (running in the AP) receives and sends frames to/from the associated STA. It builds, updates, and stores different statistics using information in each of the frames sent and received. The Wi-5 controller can then periodically request these statistics from the agent, using a time period it defines according to the needs of the resource management algorithms. Once the statistics have been sent to the controller, the agent resets their values and the gathering of the information starts again.

---

[8] See top(1) - Linux man page, https://linux.die.net/man/1/top, [Accessed Dec 2016].
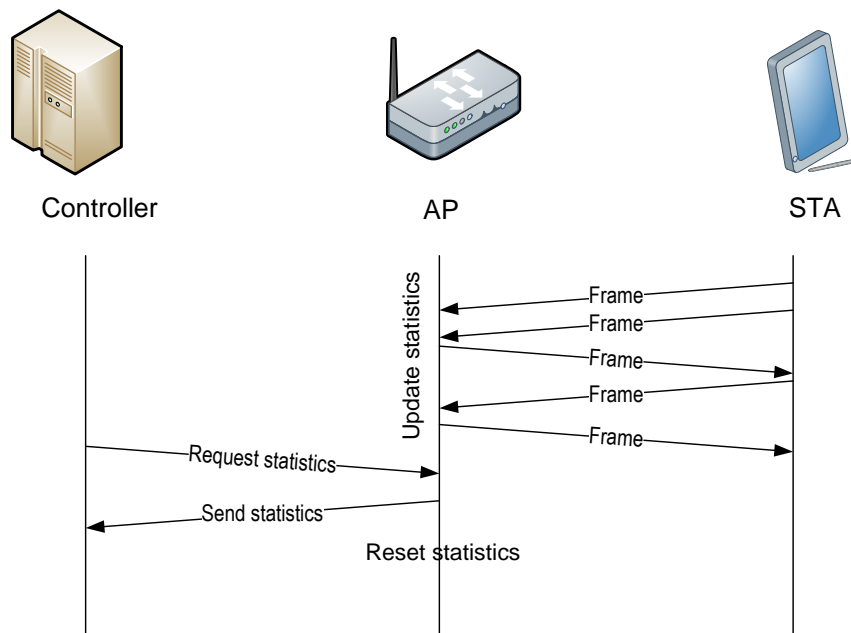
Figure 11: Scheme of the elements involved in the radio statistics process

A subsection with specific information about the statistics has been added to the wiki in the Wi-5 GitHub repository [9].

3.3.1   Statistics in the Wi-5 agent

The Wi-5 agent runs as an additional element of Click Modular Router, running inside the AP. The agent is programmed to manage all the packets sent and received to/from the AP. For the purpose of building the monitoring statistics, new functionalities have been included, and some others have been extended, with the aim of:

- Gathering information of both the uplink and the downlink.
- Getting more detailed information about the frames.
- Performing some processing in the AP, in order to send averaged statistics to the controller.

In addition to the gathering of information for the controller, periodical reports have been included in the agent in order to monitor the status of each of the APs and the associated STAs. These reports can be shown by the standard output of the device that has started the process (this can be the screen of a remote machine connected via SSH (Secure Shell) to the AP, and can be used for debugging purposes).

In Figure 12, an example of the periodical report of the information gathered by an AP, which has a single associated STA, is shown:

```
[Odinagent.cc] ##### Periodic report. Number of stations associated: 1
[Odinagent.cc]        Station -> BSSID: 00:1B:B3:88:66:C0
[Odinagent.cc]                -> IP addr: 192.168.101.200
[Odinagent.cc]           Downlink (transmission)
[Odinagent.cc]                -> last_packet_rate: 24 (12000 kbps)
[Odinagent.cc]                -> last_packet_noise: 0
[Odinagent.cc]                -> last_packet_signal: 281 (25 dBm)
[Odinagent.cc]                -> last_packet_length: 328 bytes
```

---

[9] See "STA Statistics measured by the Agents and sent to the Controller," https://github.com/Wi5/odin-wi5/wiki/STA-Statistics-measured-by-the-Agents-and-sent-to-the-Controller

```
[Odinagent.cc]                    -> total packets: 51
[Odinagent.cc]                    -> avg_rate: 12000.000000 Kbps
[Odinagent.cc]                    -> avg_signal: 25.000000 dBm
[Odinagent.cc]                    -> avg_len_pkt: 903.078431 bytes
[Odinagent.cc]                    -> air_time: 30.704667 ms
[Odinagent.cc]                    -> first heard: 1472547227.230445427 sec
[Odinagent.cc]                    -> last heard: 1472547227.736631414 sec
[Odinagent.cc]                    -> interval heard: 0.506185987 sec
[Odinagent.cc]
[Odinagent.cc]            Uplink (reception)
[Odinagent.cc]                    -> last_packet_rate: 2 (1000 kbps)
[Odinagent.cc]                    -> last_packet_noise: 0
[Odinagent.cc]                    -> last_packet_signal: 220 (-36 dBm)
[Odinagent.cc]                    -> last_packet_length: 24 bytes
[Odinagent.cc]                    -> total packets: 46
[Odinagent.cc]                    -> avg_rate: 49391.304348 Kbps
[Odinagent.cc]                    -> avg_signal: -35.908758 dBm
[Odinagent.cc]                    -> avg_len_pkt: 176.521739 bytes
[Odinagent.cc]                    -> air_time: 1.956741 ms
[Odinagent.cc]                    -> first heard: 1472547226.593717712 sec
[Odinagent.cc]                    -> last heard: 1472547227.959109068 sec
[Odinagent.cc]                    -> interval heard: 1.365391356 sec
```

Figure 12: Periodic report created by an agent with an associated STA

As it can be seen, this *periodic report* first shows the Odin LVAP, i.e. a MAC address specifically created for each STA (in this example, the real MAC of the device is `40:F3:08:88:66:C0`, but the Odin controller has modified the first three bytes and maintained the others, resulting in a BSSID `00:1B:B3:88:66:C0`). The AP is using this MAC for communicating with this STA, which has been assigned (via DHCP) an IP address `192.168.1.200` in this case.

Next, the different parameters about the radio link are included in the report. This is done for both the downlink (AP to STA) and the uplink (STA to AP). Although the statistics correspond to the measurements of the link between the AP and the STA, it should be noted that the measured parameters are always those measured at the wireless interface of the AP.

The values of these parameters are provided by Radiotap [20], which is the *de-facto* standard for injection and reception of 802.11 frames in Linux, FreeBSD, openBSD, and netBSD (Windows also supports it with *airpcap)*. It permits the gathering of radio parameters of the received frames from the driver to user space; and for transmission, it allows user space applications to specify the transmission parameters. It is based on a number of fields specified on a bitmask *presence* field in the header. Therefore, if a bit is set, you can expect to find the parameter in the corresponding place.

As an example, Figure 13 shows a Radiotap header captured with Wireshark. Each of the bits of the `Present flags` field is set if the corresponding field is present. The fields `Data Rate`, `Channel frequency`, and `SSI Signal` report the values of these parameters. More detailed information can be found in the Radiotap web page[10].

---

[10] Fields in Radiotap, http://www.radiotap.org/defined-fields, [Accessed Dec 2016]

```
            Header pad: 0
            Header length: 32
        ∨ Present flags
            ∨ Present flags word: 0x0000482f
                .... .... .... .... .... .... .... ...1 = TSFT: Present
                .... .... .... .... .... .... .... ..1. = Flags: Present
                .... .... .... .... .... .... .... .1.. = Rate: Present
                .... .... .... .... .... .... .... 1... = Channel: Present
                .... .... .... .... .... .... ...0 .... = FHSS: Absent
                .... .... .... .... .... .... ..1. .... = dBm Antenna Signal: Present
                .... .... .... .... .... .... .0.. .... = dBm Antenna Noise: Absent
                .... .... .... .... .... .... 0... .... = Lock Quality: Absent
                .... .... .... .... .... ...0 .... .... = TX Attenuation: Absent
                .... .... .... .... .... ..0. .... .... = dB TX Attenuation: Absent
                .... .... .... .... .... .0.. .... .... = dBm TX Power: Absent
                .... .... .... .... .... 1... .... .... = Antenna: Present
                .... .... .... .... ...0 .... .... .... = dB Antenna Signal: Absent
                .... .... .... .... ..0. .... .... .... = dB Antenna Noise: Absent
                .... .... .... .... .1.. .... .... .... = RX flags: Present
                .... .... .... .0.. .... .... .... .... = Channel+: Absent
                .... .... .... 0... .... .... .... .... = MCS information: Absent
                .... .... ...0 .... .... .... .... .... = A-MPDU Status: Absent
                .... .... ..0. .... .... .... .... .... = VHT information: Absent
                ...0 0000 00.. .... .... .... .... .... = Reserved: 0x00
                ..0. .... .... .... .... .... .... .... = Radiotap NS next: False
                .0.. .... .... .... .... .... .... .... = Vendor NS next: False
                0... .... .... .... .... .... .... .... = Ext: Absent
        MAC timestamp: 1123643543634
    > Flags: 0x10
        Data Rate: 1.0 Mb/s
        Channel frequency: 2432 [BG 5]
    > Channel flags: 0x00a0, Complementary Code Keying (CCK), 2 GHz spectrum
        SSI Signal: -46 dBm
        Antenna: 1
    > RX flags: 0x0000
```

Figure 13: Example of a Radiotap header

It should be noted that not all the drivers provide all the parameters. This depends on the driver and the underlying hardware. For example, in our case the driver did not provide the value of the `Antenna noise`[11], i.e. the RF noise power measured at the antenna.

The code updating the statistics is mainly included in the functions `OdinAgent::update_rx_stats` and `OdinAgent::update_tx_stats`, of the Wi-5 agent[12], which are called every time a packet is sent/received by the agent.

Here are the parameters measured and reported:

- Parameters of the last packet (`rate, noise level, signal level` and `length`). They are mainly useful for debugging purposes, as these individual parameters are not sent to the controller because they are not significant (the averaged values are more useful).
  - `rate` corresponds to the Radiotap `Rate` field.
  - `noise level` corresponds to the Radiotap `Antenna noise` field.
  - `signal level` corresponds to the Radiotap `Antenna Signal` field.
  - `length` corresponds to the length of the packet, (including the MAC, IP and Transport headers)[13].
- `total packets` is the number of packets sent/received during the interval. It is calculated as a counter in the agent.

---

[11] For further information: http://www.radiotap.org/defined-fields/Antenna%20noise
[12] The code is available in the Wi-5 GitHub repository: https://github.com/Wi5/odin-wi5-agent/blob/master/src/odinagent.cc
[13] More details are given in the class description: http://www.read.cs.ucla.edu/click/doxygen/class_packet.html, accessed Nov 2016.

- `avg_rate` is the average rate of the frames sent/received during the interval. It is calculated incrementally, using the previous average value and the rate of the current packet $N+1$ (note that Radiotap counts it as multiples of 500 kbps[14]):

$$average\_rate_{N+1} = average\_rate_N + \frac{rate_N * 500 - average\_rate_N}{N+1}$$

- `avg_signal` is the value of the power level of the frames sent/received during the interval. In order to calculate the value of the signal, the value is first converted from dBm to mW, averaged, and converted to dBm again.

$$average\_signal\_mW_{N+1} = 10^{average\_signal\_dBm/10}$$

$$average\_signal\_mW_{N+1} = average\_signal\_mW_N + \frac{signal\_mW_N - average\_signal\_mW_N}{N+1}$$

$$average\_signal\_dBm_{N+1} = 10 * \log_{10}(average\_signal\_mW_{N+1})$$

- `avg_length` of the packets sent / received during the interval. It is calculated incrementally:

$$average\_length_{N+1} = average\_length_N + \frac{length_N - average\_length_N}{N+1}$$

- `Air time` gives an idea of the air time these packets have consumed, i.e. the quotient of the length (in bits, `stat._len_pkt`) by the rate (in bits per second, `stat._rate*500`). The parameter is calculated for each of the packets, and it is cumulative:

$$air\_time_{N+1} = air\_time_N + \frac{8 * length_{N+1}}{rate_{N+1} * 500}$$

- `First heard` and `last heard` are the initial and final timestamps of the interval. The function `stat._time_first_packet.assign_now` is used for getting the current timestamp.

- `Interval heard` is the duration of the interval when the statistics have been gathered.

Different periods can be defined in the agent:

- A time interval after which the statistics will appear on screen.
- A timer interval after which the stats of old STAs (those no longer active) will be removed.

The controller can send a statistics request message to an agent at any point, and in this case the agent will respond sending the Tx/Rx statistics of all the STAs associated to it. New handlers (`handler_txstat` and `handler_rxstat`) have been included in the agent for managing these requests.

---

[14] Note that the rate is given by the Radiotap driver as an integer, which has to be multiplied by 500 kbps (e.g. "1" means 500 kpbs, "12" means 6 Mbps, etc.).

3.3.2 Statistics in the Wi-5 controller

The Wi-5 controller runs on top of the Floodlight OpenFlow controller and the radio management algorithms run as Java applications on top of it. The new functionalities are being included in the Wi-5 GitHub repository[15].

As said before, radio statistics are sent from the agents to the controller under its request to be used by the radio resource management algorithms. As a proof of concept, an application called ShowStatistics.java has been built in the controller[16], which shows the results of the statistics gathered from the APs. Figure 14 presents the output of this application, when it is gathering statistics from two APs (namely 192.168.101.9 and 192.168.101.10). The first AP has an STA associated (192.168.1.200).

```
[ShowStatistics] 1/192.168.101.9
        Uplink station MAC: 40:F3:08:88:66:C0 IP: 192.168.101.200
                num packets: 14
                avg rate: 23714.2857143 kbps
                avg signal: -34.4715803134 dBm
                avg length: 49.7142857143 bytes
                air time: 1.61066666667 ms
                init time: 1472547242.307101602 sec
                end time: 1472547251.279112125 sec

        Downlink station MAC: 40:F3:08:88:66:C0 IP: 192.168.101.200
                num packets: 18
                avg rate: 12000 kbps
                avg signal: 25 dBm
                avg length: 86.8333333333 bytes
                air time: 1.042 ms
                init time: 1472547242.361167266 sec
                end time: 1472547251.709062530 sec

[ShowStatistics] Agent: /192.168.101.10
[ShowStatistics] Last ping heard from agent /192.168.101.10 1472547344335
```

Figure 14: Report of radio statistics in the controller

As it can be observed, the parameters previously gathered by the agent are now shown in the controller: the number of packets sent or received during the period; the average rate, the signal level and length; the *air time* usage; and the initial and final timestamps of the interval.

As every agent sends regular *keep alive* messages to the controller (every second by default), the application also shows the timestamp of the last ping heard from the agent.

<p align="center">***</p>

New statistics can now be easily added to the framework if required, following the same steps just presented. We have decided to focus on those outlined above, as they are required by the radio management algorithms under development in WP4.

3.4 Monitoring the STAs during the handoff

As explained before, one of the moments where monitoring is required is the process of a handoff between two APs in different channels. STAs associated to other Wi-5 APs may move, so they have to

---

[15] See the Odin Wi-5 Controller repository in https://github.com/Wi5/odin-wi5-controller
[16] See https://github.com/Wi5/odin-wi5-controller/blob/development/src/main/java/net/floodlightcontroller/odin/applications/ShowStatistics.java

be detected when they approach, taking into account that we are considering users that walk while using the service. We are proposing a network-initiated handover between Wi-5 APs, so the capability of detecting a new user is necessary in order to trigger the procedure in a timely manner.

The use of Multichannel Light Virtual APs was first proposed in [21], allowing an STA to change the AP and the channel at the same time. The solution was based on the communication between the involved APs. The Wi-5 solutions will follow this approach but, instead of using an AP-AP protocol, the handoff will be governed by the controller.

Therefore, a trade-off appears: on the one hand, scanning in other channels is necessary, but on the other hand, the AP would not be able to manage its own clients while scanning [5]. As explained in Subsection 3.3.2, this trade-off has been solved by the addition of an auxiliary 802.11 interface, which is only used for scanning purposes, relieving the main interface from this task.

As illustrated in Figure 15, we are considering a client moving from AP_1 to AP_2. Logically, it can be expected that adjacent APs are in different (and non-overlapping) Wi-Fi channels. Therefore, AP_2 will not receive frames from a client associated with AP_1. Since an event is required to trigger the handover, it is necessary to establish scanning capabilities in AP_2 in order to look for Wi-5 users in the vicinity. If a user is detected, it can be reported to the controller which, according to its load balancing algorithm, will make a decision about handing off the client or not.
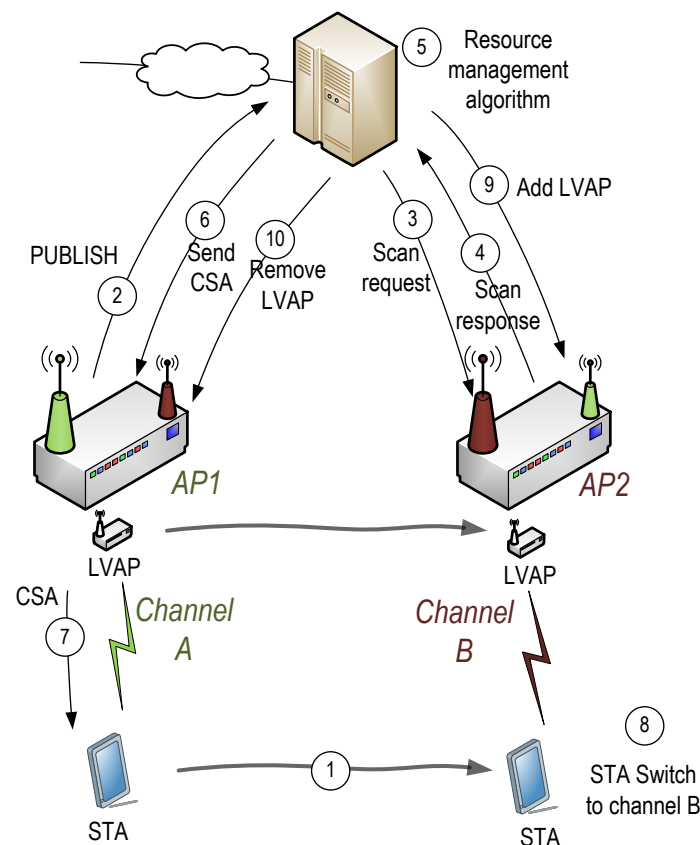

Figure 15: Need for scanning in other channels to trigger hand-offs

The full handoff process is explained in detail in Deliverable D3.3. However, it can be seen from the above that some monitoring functionalities have to run in order to make it possible:

Wi-5: What to do With the Wi-Fi Wild West

- In step (2), a *PUBLISH* message is sent to the controller when a predefined parameter reaches a threshold. The Radiotap parameters explained in Subsection 3.3.1 are used for this purpose: for example, when the power received (`signal`) from a frame sent by a STA is below a certain level (e.g. -100dBm), this fact can be reported to the controller.
- In step (3), the controller sends a *Scan Request* to an AP, requesting it to scan for a certain STA in a certain channel. After a scanning period, a *Scan Response* is returned to the controller. The function `scanClientFromAgent`[17] of the controller is used for that, and a new handler called `scan_client`[18] has also been added in the code of the agent.

With this information, the controller can make a decision, in step (5), about the best AP where the STA should be moved. It should be taken into account that more than a single AP can be considered as a destination for the STA. The next steps of the handoff do not require more monitoring information, but they are based on the accuracy of the information gathered by the auxiliary interfaces.

3.5 Monitoring other Wi-5 APs in the neighbourhood

Wi-5 APs also incorporate scanning features in order to gather information of the occupancy of the radio channels in their neighbourhood. This information is then forwarded to the controller, to be used as an input for the radio resource management algorithms. The interference caused by the different APs in the network (both under and outside the Wi-5 administrative domain) has to be detected. Those in the same Wi-5 administrative domain can then be properly allocated to minimise the interference by using the radio resource management algorithms.

New code has been added in the Wi-5 agent in order to support these functions[19]. These features rely on the auxiliary wireless interface, to avoid connection disruption. The scheme is illustrated in Figure 16, which we will use for the explanation: the controller tells AP_2 (operating in channel 1) to send beacons using its auxiliary interface in channel 6, with the SSID '`odin_init`' during 200ms. At the same time, the controller tells AP_1 and AP_3 (both operating in channel 11) to scan for these beacons using their auxiliary interfaces, also in channel 6, and send a report to the controller. Finally, a "map" of the "distance" in dBs between the AP_2 and the two other APs can be built in the controller. We next explain the developed functionalities in more detail.

---

[17] See https://github.com/Wi5/odin-wi5-controller/blob/MobilityManager/src/main/java/net/floodlightcontroller/odin/master/IOdinMasterToApplicationInterface.java
[18] See https://github.com/Wi5/odin-wi5-agent/blob/MobilityManager/src/odinagent.cc
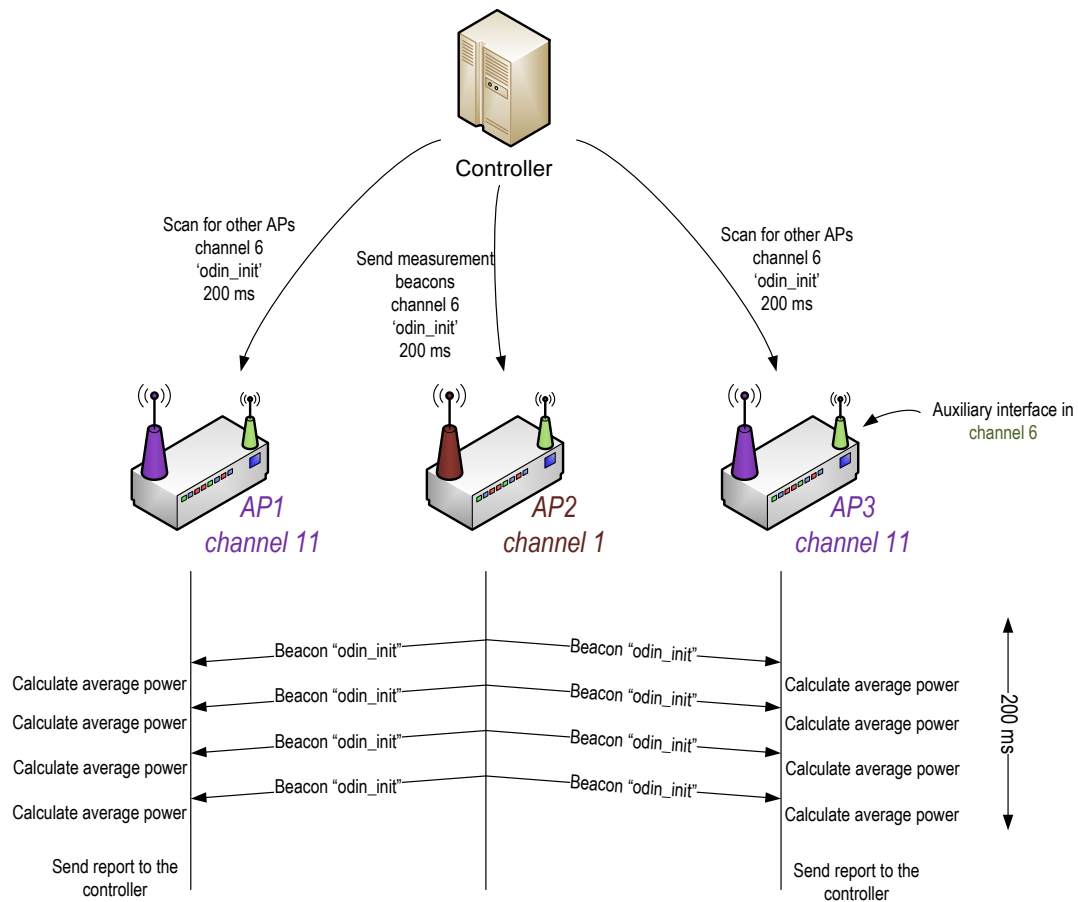[19] See https://github.com/Wi5/odin-wi5-agent/tree/APScanning

Figure 16: Scheme of the monitoring of other APs

3.5.1   Flags for coordination

In order to avoid an AP from performing different monitoring tasks at the same time, a set of flags have been defined. Three flags have been created, so if one of them is active, then the other tasks cannot be performed at the same time:

- `_active_client_scanning`: the scanning for a client (explained in 3.4) is active.
- `_active_AP_scanning`: the function for scanning for APs is active.
- `_active_mesurement_beacon`: the auxiliary interface is being used for sending measurement beacons (see next subsection).

3.5.2   Sending measurement beacons

This new functionality makes the agent send a series of beacons during a time interval (see Figure 16), with a predefined SSID (e.g. '`init_odin`'). These beacons are just for making other APs aware of the existence of this other AP, and to measure the power received.

Two new complementary WRITE handlers have also been added in the agent:

- `handler_send_mesurement_beacon`. The parameters are the `channel`, the `SSID` and a `time_interval.` When the corresponding agent receives this message from the controller, it:
  o Puts the auxiliary interface `mon1` in the selected `channel`
  o Sends beacons with the `SSID` during `time_interval` seconds

Wi-5: What to do With the Wi-Fi Wild West

- `handler_scan_APs`: the controller tells the agent to scan for beacons in a `channel`, with a specific `SSID` during a time `interval`. If the `SSID` is '*', it will hear all the SSIDs. Note that this allows us to also measure the interference from non-Wi-5 APs in the neighbourhood.

By the use of these two handlers, the attenuation in dB from one agent to the other can be measured, as the power with which the beacons are sent is known.

3.5.3   Scanning for other APs

This primitive just listens to the beacons generated by the previous functionality (see Figure 16). One new READ handler has been included in the code of the agent in order to report the result of the scanning:

- `handler_scan_APs`: the controller tells the agent to send back the result of the scanning for beacons in a `channel`, with a specific `SSID`.
- It sends back to the controller a table including the `BSSID`, the `SSID` and the `average power` of all the beacon frames heard:

```
BSSID                         SSID            Average power
04:20:AC:37:FA:2B             odin_init       48
60:e3:27:1d:32:b7             odin_init       36
c4:6e:1f:8c:dd:d2             odin_init       56
```

The average power is updated with the power level with which each beacon that has been received. It is measured in the same units given by Radiotap.

3.5.4   Collecting the statistics between the APs

These monitoring primitives can be combined in order to collect information of the interference from the agents. The next example shows how the controller could build a matrix including the "distance" in dBs between the Wi-5 APs. This structure will consist of an N x N array per channel (where N is the number of APs).

```
for ch = all channels
  for i = 1 to N
    // the AP 'i' starts sending beacons with SSID 'odin_init' in channel 'ch'
    send_measurement_beacons ( i, ch, 200ms, odin_init)

    // at the same time, the rest of the APs listen to those beacons in channel 'ch'
    for j = i + 1 to N
        scan_for_other_APs ( j, ch, 200ms, odin_init)
    end
  end
end
```

3.5.5   How to estimate the overlapping level between channels

In 802.11, adjacent channels have some overlap, and these tools can also be used to measure the extent to which this occurs. In the next example, we assume a scenario with only 2 APs. We want an estimation of how much power of frames generated by AP_2 in channel 7 is captured by AP_1 in channel 6.

```
// AP2 sends beacons in channel 7
send_measurement_beacons (2, 7, 200ms, odin_init)

// hear the beacons in AP1 in channel 6
scan_for_other_APs (1, 6, 200ms, odin_init)
```

So this would give us an estimation of the overlapping between channels 6 and 7.

3.6 Detecting the User Service

One of the aims of the Wi-5 architecture is to provide good quality to real-time services with very tight latency constraints. A first step for this is to detect these traffic flows and the APs where they are present. This can be an additional input to the controller algorithms, which can then make better decisions using this information. As an example, if a user running a real-time service is under the coverage of two APs, the controller can decide not to assign it to an AP where a number of users are downloading large files, as the delay and jitter would be higher.

Different options can be considered for detecting the service: in [11], a solution called *Diffuse* (DIstributed Firewall and Flow-shaper Using Statistical Evidence) was proposed. It is designed to run inside an AP, e.g. those based on OpenWrt, but it can be executed elsewhere (e.g. in a Linux machine). In order to implement the prioritisation of traffic and to establish different queues, tagging IP packets is based on a statistical analysis made using data mining software [22].

Other tools can be used for this aim, e.g. *Bro IDS* (Intrusion Detection System)[20], which can be executed in the same network elements that route traffic, but can also be attached to the network in a mirror link as an independent monitoring tool.

Therefore, there are two places where traffic detection can be performed: it can be done in a distributed manner, i.e. running on each AP which would in turn send a report to the controller every time a real-time traffic is detected; or a centralised approach can also be taken, using the router, or even a specific traffic-detection machine. The advantage of the first option is that it is distributed, but it should be taken into account that the APs have a limited processing capacity. If the centralised option is taken, a more capable machine can be used (e.g. the router or a machine next to it), and traffic detection can be made there. Therefore, the approach we have followed is the latter (see Figure 17), where a specific machine is in charge of the detection of real-time flows.

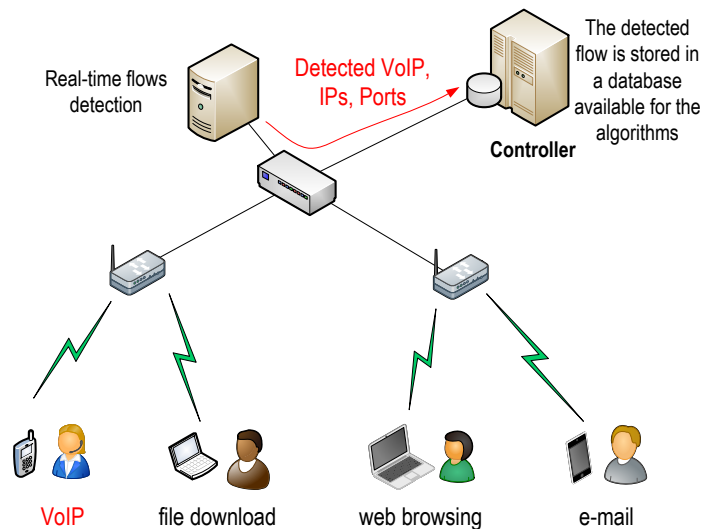[20] The Bro Network Security Monitor, https://www.bro.org/

Figure 17: Scheme for the traffic detection functionality

3.6.1   Integration of the service detection

As explained in the previous subsection, different tools for detecting traffic flows are available. They can be based on statistical features of the traffic flows, integrate machine learning, etc. In our case, the most important question is how to integrate these detection tools into the Wi-5 framework.

The initial idea is to detect real-time traffic flows, i.e. those with certain delay requirements as in VoIP or online games. The information about a real-time flow can be useful for the radio resource management algorithms, in order to make better decisions about load balancing for example.

The final objective is to let the Wi-5 controller have information about the flows that are currently present in the network. Although the primary objective of the service detection was for real-time flows, the integration has been done in a generic way such that it not only reports the real-time flows, but different flow categories can be defined, so each of them may have associated its own requirements (in terms of e.g. delay or bandwidth).

The integration has to be made in a way that permits the use of any classification tool, and its seamless integration within the Wi-5 framework. If this objective is achieved, different classification tools can be easily integrated. Therefore, we have made a design that decouples these two functionalities:

- *Classifier*: is in charge of traffic classification and flow detection.
- Wi-5 *flow report*: Reporting of the detected flows to the controller.

The scheme is shown in Figure 18, where the *classifier* has been included into one of the routers managing all the traffic to and from the end users. This element analyses all the traffic traversing it, and includes a classification tool (e.g. *Diffuse*, *Bro IDS*, etc.) that is configured to duplicate the traffic of interest and send it to the Wi-5 *flow report* machine.

We have decided to use traffic duplication because it can be easily integrated with any classification tool. For example, the `-j TEE` option of the `iptables` Linux tool can be used to duplicate the traffic[21].

---

[21] For example, this simple command works for a Linux kernel 3, duplicating all the packets received from 192.168.200.3 to 192.168.0.4, where the *flow report* machine would be listening:
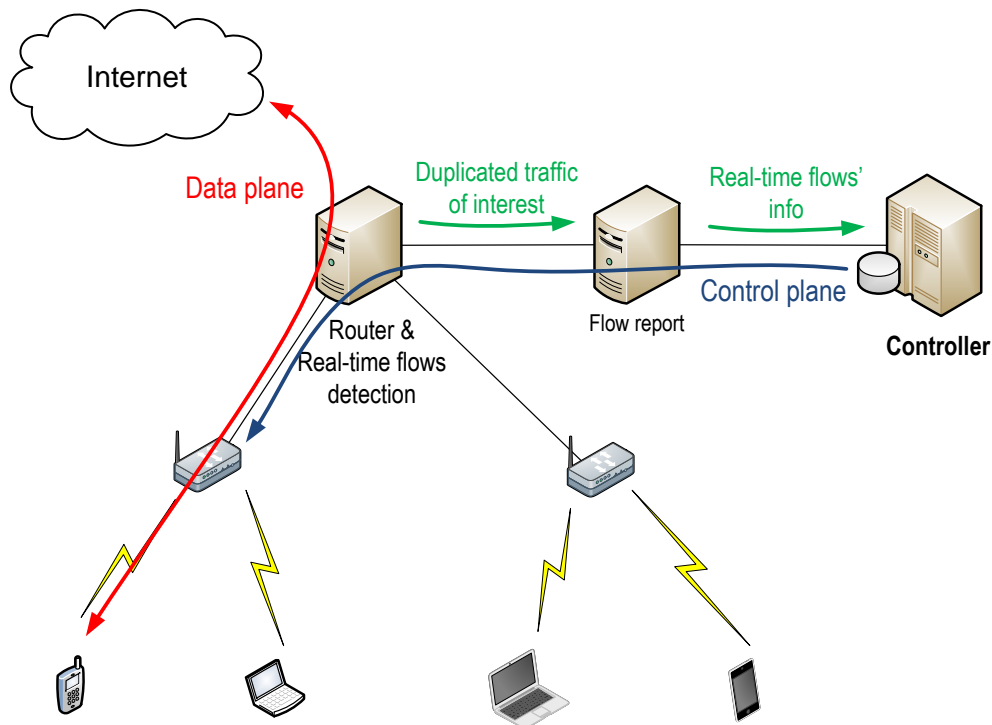
Figure 18: Scheme of the detection and its integration within the Wi-5 framework

Flow Report machine

The *flow report* machine receives a series of duplicated traffic packets corresponding to those traffics detected by the *classifier*. Every packet that enters to this interface will be analysed and included into a *flow*, defined by a 5-tuple, including the following fields:

- *IP source address*
- *IP destination address*
- *TCP/UDP source port[22]*
- *TCP/UDP destination port*
- *Protocol*

Taking into account that Wi-5 agents are implemented as Click Modular Router elements, we have decided to use the same tool for implementing the *flow report* machine. A new Click element (*detection_agent*) has been defined, with one input and one output port (see Figure 19):

- *In-port-0*: Any Ethernet encapsulated frame.
- *Out-port-0*: Used exclusively to talk to a Socket UDP, to be used to communicate with the Wi-5 controller.

The scheme simply considers as input the packets received from the host, classifies it, then builds the reports and sends them to the *odinsocket,* which is in charge of sending the information to the Wi-5 controller.

```
~$ iptables -t mangle -A PREROUTING -s 192.168.200.3 -j TEE --gateway 192.168.0.4
```
[22] Only TCP and UDP packets are considered. Packets belonging to other protocols are currently discarded.
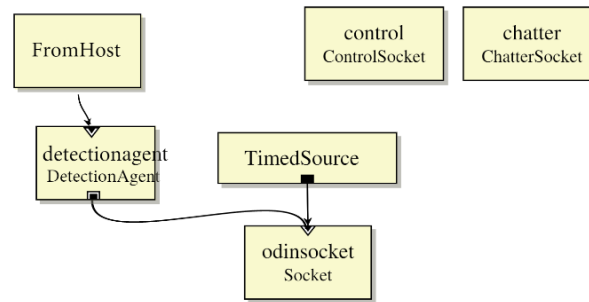
Figure 19: Scheme of the Click element implementing the *flow report* machine

Therefore, information about each of the flows will periodically be sent to the Wi-5 controller. For that aim, different time periods that can be defined by the user:

- Interval for periodic report by the standard output, e.g. the screen of a device used for debugging (`RESCHEDULE_INTERVAL_GENERAL`).
- Interval for sending information to the controller (`RESCHEDULE_INTERVAL_FLOWS`).
- Interval for removing old flows: Flows which have not been updated in the last `THRESHOLD_REMOVE_FLOWS` are removed.
- Interval after which a FLOW message can be sent again to the Odin controller (`THRESHOLD_FLOWS_SENT`).

3.6.2 Implementation of the service detection

The system for service detection described in this subsection has now been implemented and tested. The source code has been publicly shared in GitHub[23]. The next files have been provided:

- A README file providing the information about the tool, how to compile it, and its configuration.
- A Python script for generating the configuration file of Click Modular Router.
- Two files (`detection_agent.cc` and `detection_agent.hh`) implementing the detection agent (i.e. the *flow report* machine) as a Click Modular Router element.
- The scheme of the Click detection agent.

The created agent sends messages to the controller starting with the string "`Flow `". They are similar to the PUBLISH messages used by the normal agents.

Test of the integrated system
In order to test the integrated system the following lab setup was created, which consists of a *sender* (*miniPC2*), a *flow report* agent running in *miniPC3*, and a Wi-5 controller:

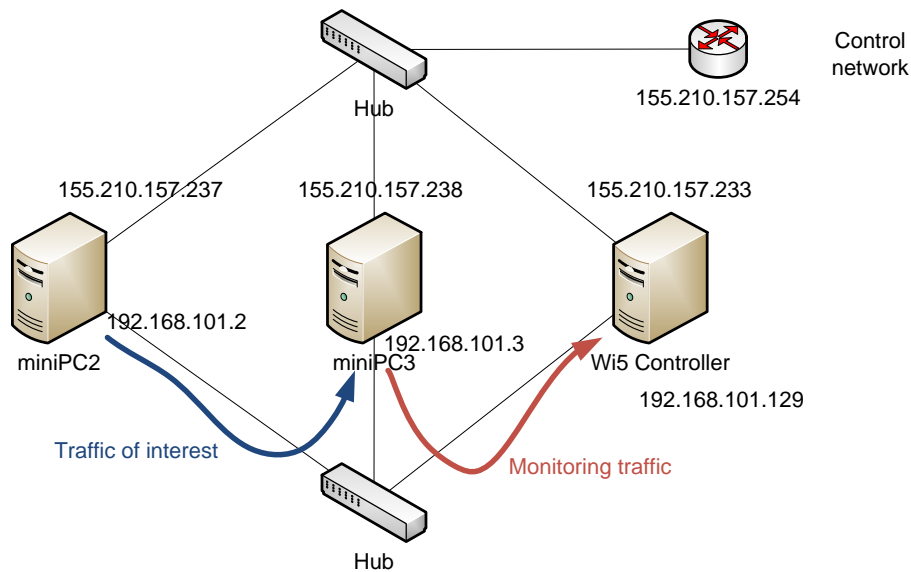[23] See this repository https://github.com/Wi5/odin-wi5-flow-detection

Figure 20: Scheme of the setup for testing the service detector

The sender (*miniPC2*, IP address 192.168.101.2) creates a UDP flow (traffic of interest) with destination *miniPC3* (IP address 192.168.101.3). For that aim, this command is used in *miniPC2*:

```
$echo "This is my data" > /dev/udp/192.168.101.3/3000
```

The agent file (it is a script readable by Click Modular Router) can be created with the next command, where the IP of the controller, the machine, and the port to be used for the messages can be specified:

```
$python detection_agent-click-file-gen.py 192.168.101.129 2819 192.168.101.4 2 12 >
../detection.click
```

And the agent is run with this command:

```
./click/userlevel/click detection.click
```

As soon as a TCP or UDP packet arrives from another computer, the *flow report* agent running in *miniPC3* starts detecting.
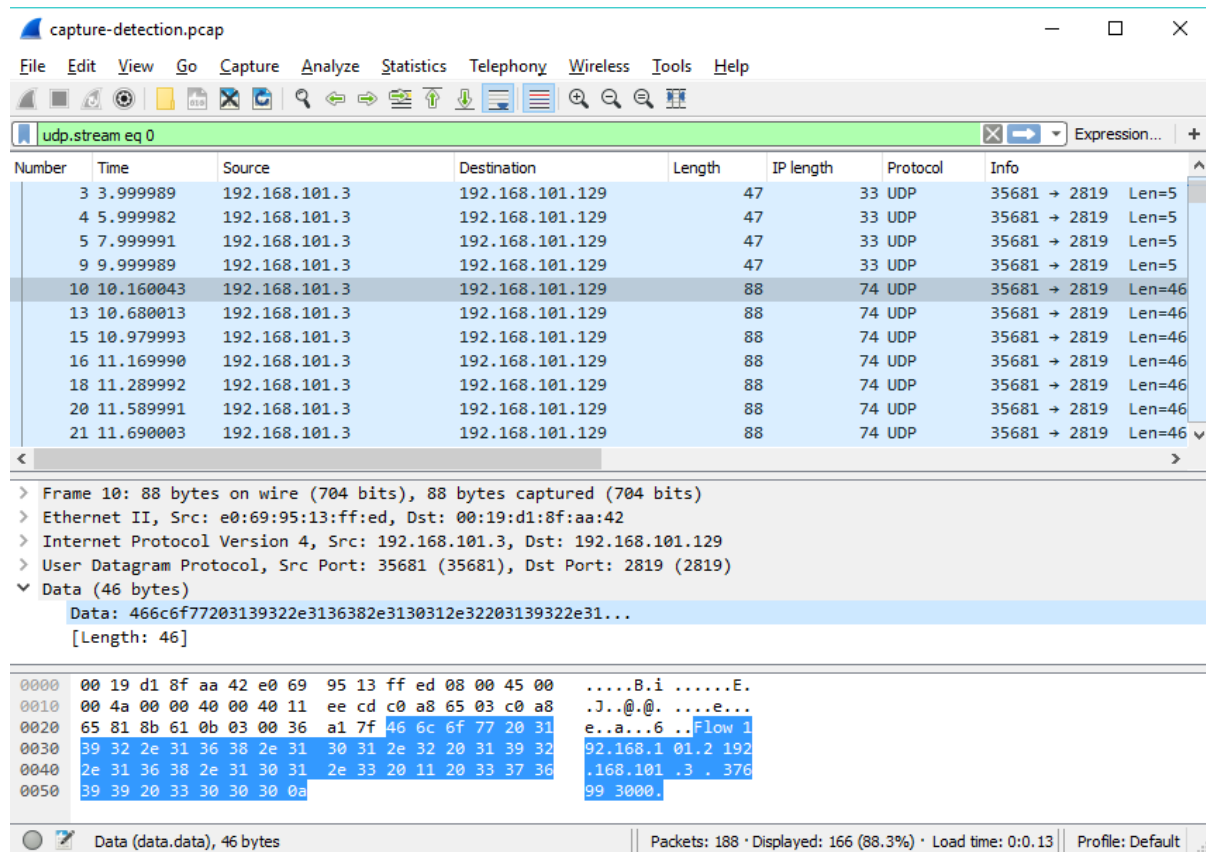
Wi-5: What to do With the Wi-Fi Wild West

Figure 21: Scheme of the Click element implementing the *flow report* machine

This is the payload of these messages, which includes the word `Flow`, plus the 5-tuple with the IP addresses, the *protocol*[24] field and the source and destination ports:

```
Flow 192.168.101.2 192.168.101.3 . 37699 3000
Flow 192.168.101.2 192.168.101.3 . 44170 3000
Flow 192.168.101.2 192.168.101.3 . 52483 3000
Flow 192.168.101.2 192.168.101.3 . 37699 3000
Flow 192.168.101.2 192.168.101.3 . 36399 3000
```

Periodic reports are also generated by the screen for debugging purposes. This is an example of a periodic report including the information of 4 different flows:

```
#################################################################
[DetectionAgent.cc] ##### Periodic report. Number of flows: 4
[DetectionAgent.cc]Flow: 1
[DetectionAgent.cc]     -> Source IP: 192.168.101.2
[DetectionAgent.cc]     -> Destination IP: 192.168.101.3
[DetectionAgent.cc]     -> Protocol: 17
[DetectionAgent.cc]     -> Source Port: 55992
[DetectionAgent.cc]     -> Destination Port: 3000
[DetectionAgent.cc]     -> last sent: 1479224716.755008897 sec
[DetectionAgent.cc]     -> last heard: 1479224707.696236085 sec
[DetectionAgent.cc]Flow: 2
[DetectionAgent.cc]     -> Source IP: 192.168.101.2
[DetectionAgent.cc]     -> Destination IP: 192.168.101.3
[DetectionAgent.cc]     -> Protocol: 17
```

---

[24] Note that the value of the *protocol* field cannot be observed, and it is shown as a dot. The cause is that it corresponds to UDP, number 17 decimal and 11 hexadecimal, which has no representation in ASCII.

```
[DetectionAgent.cc]       -> Source Port: 51794
[DetectionAgent.cc]       -> Destination Port: 3000
[DetectionAgent.cc]       -> last sent: 1479224716.195016870 sec
[DetectionAgent.cc]       -> last heard: 1479224710.160633586 sec
[DetectionAgent.cc]Flow: 3
[DetectionAgent.cc]       -> Source IP: 192.168.101.2
[DetectionAgent.cc]       -> Destination IP: 192.168.101.3
[DetectionAgent.cc]       -> Protocol: 17
[DetectionAgent.cc]       -> Source Port: 41530
[DetectionAgent.cc]       -> Destination Port: 3000
[DetectionAgent.cc]       -> last sent: 1479224716.675013732 sec
[DetectionAgent.cc]       -> last heard: 1479224712.656139175 sec
[DetectionAgent.cc]Flow: 4
[DetectionAgent.cc]       -> Source IP: 192.168.101.2
[DetectionAgent.cc]       -> Destination IP: 192.168.101.3
[DetectionAgent.cc]       -> Protocol: 17
[DetectionAgent.cc]       -> Source Port: 41333
[DetectionAgent.cc]       -> Destination Port: 3000
[DetectionAgent.cc]       -> last sent: 1479224716.365020750 sec
[DetectionAgent.cc]       -> last heard: 1479224715.360148596 sec
###############################################################
```

And real-time information is also sent to the screen every time a new flow message is sent to the controller:

```
[DetectionAgent.cc]   flow  message sent: Flow 192.168.101.2 192.168.101.3  49001 3000
[DetectionAgent.cc]   flow  message sent: Flow 192.168.101.2 192.168.101.3  39199 3000
[DetectionAgent.cc]   flow  message sent: Flow 192.168.101.2 192.168.101.3  49001 3000
```

4   Conclusions

This document has presented the monitoring mechanisms being used by the Wi-5 APs and controller, which are necessary for monitoring the wireless environment and the services being run by the users. The resource management mechanisms developed within the Wi-5 architecture depend on having an accurate and reliable stream of information about this, in order to run properly.

A summary of the scope and objectives of the Wi-5 monitoring elements has been provided, considering the role that they play in the global Wi-5 scheme, and their logical inclusion in both the controller and the agents. Next, the relationship with other deliverables has been detailed, especially related to Deliverable 3.3. A literature review has been provided, including the mechanisms for monitoring the wireless environment in SDWN, and the topic of the automatic detection of real-time services upon which we rely. Finally, the most innovative aspects developed in Wi-5, as far as performance monitoring is concerned, have been summarised: namely the use of a secondary wireless interface on each AP, bringing the possibility of scanning in different channels without service disruption; the improvement of communication between the controller and the APs by the separation of the data and control planes, and the integration of service detection tools in the WLAN, which allow for more fine grained management of the traffic.

The main section of the document presented the performance monitoring mechanisms themselves. First, their role in the Wi-5 architecture has been explained in more detail, taking into account the approach that is being followed, in which the internal switches of the APs (Wi-5 *agents)* are now controlled by an SDN controller, in which the resource management algorithms run.

The implementation has been described, including the different software and hardware elements that are employed for each element: OpenWrt, Open Vswitch and Click Modular Router in the agents, and Floodlight controller in the Wi-5 controller. The limitations of the original framework upon which the platform is based (*Odin*) have been detailed, with a special focus on the monitoring part. The main limitations are the use of a single channel by all the APs, the use of the same connection for data and control traffic, and the limitation of the available applications.

The modifications implemented for having the information of the wireless environment that is required by the applications have been detailed: the addition of an auxiliary interface, and the modification of the internal network scheme of the agents.

Finally, the different monitoring features now available in the Wi-5 platform have been detailed:

- The gathering of the information about the STAs connected to each of the Wi-5 APs. This information is collected by the APs, and forwarded to the controller, which can have a global vision of the status of the network. The information includes the number of packets, rate, power level, etc.
- The monitoring information that is required in order to perform fast handoffs between APs, which also include a channel switch of the STA. The importance of this information for performing a handoff is high, especially in some key moments of the process.
- The primitives included in the controller and the agents to monitor the interference caused by other APs in the neighbourhood. This includes the sending of special beacons and the detection of its power level. This information is then forwarded to the controller, which can build different "interference maps" between APs. They can also be used for measuring interference levels between APs in adjacent channels.

- The integration of the automatic detection of real-time flows in the Wi-5 scheme. A centralised approach has been followed, in which the detector is integrated within the router, and a *flow report* machine is in charge of taking the packets, grouping them in flows and sending this information to the controller. It has been done in a generic way, so it allows different detection tools to be employed.

All the developed functionalities are being shared in a public software repository (in GitHub), which also allows for input from other interested developers and users.

References

[1] S. Chieochan, E. Hossain, J. Diamond, "Channel assignment schemes for infrastructure-based 802.11 WLANs: a survey," IEEE Communications Surveys and Tutorials 2010; 12(1): 124–136.

[2] D. Qiao, S. Choi, K.G. Shin, "Interference Analysis and Transmit Power Control in IEEE 802.11a/h Wireless LANs," IEEE/ACM Transactions on Networking, vol. 15, no.5, pp. 1007-1020, Oct. 2007.

[3] L.-H. Yen, T.-T. Yeh, K.-H. Chi, "Load Balancing in IEEE 802.11 Networks", IEEE Internet Computing, Jan-Feb 2009, pp. 56-64.

[4] H. Wu, K. Tan, Y. Zhang, Q. Zhang, "Proactive Scan: Fast Handoff with Smart Triggers for 802.11 Wireless LAN," in INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pp.749-757, May 2007.

[5] S. Lee, M. Kim, S. Kang, K. Lee, I. Jung, "Smart scanning for mobile devices in WLANs," Communications (ICC), IEEE International Conference on. IEEE, 2012.

[6] J. But, G. Armitage, L. Stewart, "Outsourcing automated QoS control of home routers for a better online game experience," IEEE Commun. Mag., vol. 46, no. 12, pp. 64–70, Dec. 2008.

[7] J. Frank, "Machine learning and intrusion detection: Current and future directions," in Proc. 17th Nat. Comput. Security Conf., Oct. 1994, pp. 22–33.

[8] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in Proc. PAM, Apr. 2004, pp. 205–214.

[9] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in Proc. 4th ACM SIGCOMM IMC, Oct. 2004, pp. 135–148.

[10] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in Proc. IEEE 30th LCN, Nov. 2005, pp. 250–257.

[11] T. T. T. Nguyen, G. Armitage, P. Branch, S. Zander, "Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic," in Networking, IEEE/ACM Transactions on, vol.20, no.6, pp.1880-1894, Dec. 2012.

[12] J. Schulz-Zander, L. Suresh, N. Sarrar, A. Feldmann, T. Hühn, R. Merz, "Programmatic orchestration of wifi networks," in USENIX Annual Technical Conference (USENIX ATC 14), pp. 347-358, Jun 2014.

[13] L. Sequeira, J. L. de la Cruz, J. Ruiz-Mas, J. Saldana, J. Fernandez-Navajas, J. L. Almodovar, "Building a SDN Enterprise WLAN Based On Virtual APs," IEEE Communications Letters, Published online. Oct. 2016, doi 10.1109/LCOMM.2016.2623602.

[14] M. Suznjevic, J. Saldana, M. Matijasevic, and M. Vuga, "Impact of Simplemux Traffic Optimisation on MMORPG QoE," presented at the PQS 2016, 5th ISCA/DEGA Workshop on Perceptual Quality of Systems, Berlin, Germany, 2016.

[15] J. Saldana, "The effect of multiplexing delay on MMORPG TCP traffic flows," in 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), 2014, pp. 245–250.

[16] B. Ginzburg, A. Kesselman, "Performance analysis of A-MPDU and A-MSDU aggregation in IEEE 802.11n," Sarnoff Symposium, 2007 IEEE, vol., no., pp.1,5, April 30 2007-May 2 2007.

[17] J. Saldana, I. Forcen, J. Fernandez-Navajas, J. Ruiz-Mas, "Improving Network Efficiency with Simplemux," IEEE CIT 2015, International Conference on Computer and Information Technology, pp. 446-453, 26-28 October 2015, Liverpool, UK.

[18] J. Saldana, "Simplemux. A generic multiplexing protocol," draft-saldana-tsvwg-simplemux-05 (Jul. 2016). Available at http://datatracker.ietf.org/doc/draft-saldana-tsvwg-simplemux/

[19] E. Kohler, R. Morris, B. Chen, J. Jannotti, M.F. Kaashoek, "The Click modular router," ACM Transactions on Computer Systems (TOCS), 18(3), 263-297, 2000.

[20] Radiotap, the de facto standard for 802.11 frame injection and reception. http://www.radiotap.org/ [accessed Sept 2016]

[21] M.E. Berezin, F. Rousseau, A. Duda, "Multichannel Virtual Access Points for Seamless Handoffs in IEEE 802.11 Wireless Networks," in Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd , pp.1-5, May 2011.

[22] S. R. Garner, "Weka: The waikato environment for knowledge analysis," Proceedings of the New Zealand computer science research students conference. 1995.