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

Deliverable	D3.4
Title	Final specification of the Smart AP solutions
Date	30/04/2018
Milestone	MS7
Authors	Jose Saldana (UNIZAR, Editor), Rubén Munilla (UNIZAR), Jose Ruiz-Mas (UNIZAR), Julián Fernández-Navajas (UNIZAR), Alessandro Raschella (LJMU), José Almodóvar (TNO), Salim Eryigit (AirTies)
Reviewers	Michael Mackay (LJMU), Faycal Bouhafs (LJMU), Qi Shi (LJMU)
Approved by	General Assembly
Dissemination	Public

Abstract

This deliverable presents the final version of the specification for the mechanisms included in the Wi-5 Access Points (APs), which have been developed within WP3 of the Wi-5 project. Coordinated by a controller, these APs are able to run the Smart Access Point Solutions including resource management algorithms such as dynamic channel allocation, load balancing and power control. The seamless handover is also an important functionality to support this and the integration with the coordination entities of the Wi-5 architecture (*i.e.,* the Wi-5 controller) and the interface with performance monitoring mechanisms are also defined. The document also includes a series of simulations aimed at studying the possibility of performing a centrally controlled coordination of the frame aggregation functionalities available in 802.11n and 802.11ac.

The main section of this deliverable (section 4) is devoted to explaining the final version of the functionalities enabling all the Wi-5 features, with detailed information about their implementation, and the advances with respect to previous versions reported in Deliverables D3.2 and D3.3. These functions rely on the monitoring mechanisms defined in Deliverable D3.1. This section includes *a)* The framework used for the implementation based on the use of Light Virtual APs (LVAPs). *b)* The horizontal handover scheme, integrating multi-channel APs with the LVAPs approach, which includes extensive tests of the handover latency illustrating that they can really be seamless. *c)* Different applications including Channel Assignment, Mobility Management (in a reactive and a proactive way), and Load Balancing based on Received Signal (RSSI), Fittingness Factor and also considering the services being run in the terminals.

Another section (section 5) details the results of a battery of measurements of the delays incurred by the system.

Finally, a simulation environment is used in order to test different ways of performing a coordinated control of the frame aggregation mechanisms of 802.11.

A Conclusions section surveys the work that has been carried out. The most innovative aspects are: *a)* The development of a method able to proactively manage the mobility of the users, also combining this with load balancing in real time. *b)* The proposal of central coordination for frame aggregation, which can provide a significant improvement in efficiency while still respecting the real-time requirements.

Wi-5 Consortium

Liverpool John Moores University	2 Rodney Street Egerton Court Liverpool, L3 5UK United Kingdom
Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek	Schoemakerstraat 97 Delft 2628 VK Netherlands
Universidad de Zaragoza	Calle Pedro Cerbuna 12 Zaragoza 50009 Spain
AirTies Kablosuz İletişim San ve Dış. Tic. A.Ş	Gulbahar Mah Avnidilligil Sok 5 5A 7A 7B 9A 11A 11B Mecidiyekoy Instanbul 34394 Turkey
PrimeTel PLC	The Maritime Center, 141 Omonia Avenue, 3045 Limassol, Cyprus

Table of Contents

Wi-5 Consortium 2
List of Figures 5
List of Tables 8
Glossary 9
Executive Summary 11
1 Introduction 12
1.1 Background to project 12
1.2 Scope and structure of the deliverable 13
1.3 Relationship to other deliverables 13
2 Literature Review 15
2.1 Architectural proposals for Software-Defined Wireless Networks (SDWNs) 15
2.2 Use of Virtual Wi-Fi APs in SDWNs 16
2.3 Frameworks Providing Fast Handovers 17
2.4 Resource Management in Wi-Fi WLANs 17
2.5 Packet/Frame Grouping at Different Layers 18
2.6 Weaknesses of the previous work 19
3 Global view of the implemented architecture 20
3.1 Global view of Smart Access Point Solutions 20
3.2 Global view of Cooperative Functionalities 21
3.2.1 Radio Resource Management algorithm 22
3.2.2 Access Point Selection Algorithm 22
3.2.3 Vertical Handover 23
3.2.4 Integration of Monitoring Tools and Radio Configuration with Cooperative Functionalities 23
3.3 Most innovative aspects of WP3 24
4 Specification of Wi-5 Smart AP Solutions 25
4.1 Scheme of the implementation 25
4.1.1 Detailed explanation of the implementation 25
4.1.2 Monitoring functionalities 26
4.2 Channel Assignment 30
4.2.1 Parameters that can be tuned for Channel Assignment 34
4.2.2 Applications for test purposes 34
4.3 Mobility Management and Load Balancing 35
4.3.1 Reactive Mobility Management 35

4.3.2 Proactive Mobility Management 36
4.3.3 Evaluation and comparison between reactive and proactive schemes 37
4.3.4 Integration of load balancing with mobility management 46
4.3.5 Use of the Fittingness Factor parameter in the Smart AP Selection 48
4.3.6 Integration of service detection 50
5 Measurements of System Delays 53
5.1.1 Delays incurred when collecting monitoring information 53
5.1.2 Delays when dispatching configuration changes to the equipment 61
5.1.3 Delays of the applications combining monitoring and configuration changes 64
6 Packet Grouping 66
6.1.1 Layer 2 aggregation mechanisms 66
6.1.2 Frame Aggregation in Central Controlled 802.11 WLANs: the Latency vs. Throughput Trade-off 68
6.1.3 Tests and Results 69
7 Conclusions 83
References 85
Annex 1. Estimating the *distance* in dBs between the Wi-5 Access Points 89
Annex 2. Parameters of SmartAPSelection in poolfile 92
Annex 3. New mechanism for storing information in the AP 93

List of Figures

Figure 1: SDN Layer Architecture according to RFC 7426 15
Figure 2: Scheme of a Wi-Fi network using LVAPs 16
Figure 3: Scheme of the Smart AP functionality implementation 21
Figure 4: Radio configuration procedures 23
Figure 5: Elements of the implementation 26
Figure 6: Monitoring for a) proactive; b) reactive applications 27
Figure 7: Monitoring: output of the application using passive monitoring for obtaining detailed information of the status of an AP 28
Figure 8: Monitoring: output of the application using passive monitoring for obtaining detailed information of the STAs connected to an AP 28
Figure 9: Monitoring: output of the application using passive monitoring for obtaining detailed information of the wireless environment 29
Figure 10: Active monitoring scheme 30
Figure 11: Monitoring: output an application using active monitoring for calculating the path loss matrix between 4 APs 30
Figure 12: Monitoring: *distance in dBs* matrix between 6 APs 31
Figure 13: Monitoring: *Internal Interference Impact* matrix between 6 APs 31
Figure 14: Monitoring: *External Interference Impact* matrix using 6 APs 31
Figure 15: Operation scheme of the Channel Assignment application 32
Figure 16: Results of the Channel Assignment application 33
Figure 17: Scheme of the operation of the *ChannelAssignment* application 33
Figure 18: Messages sent to the APs by the Channel Assignment application 33
Figure 19: Scheme of the proactive mobility management 36
Figure 20: Scenarios for the tests: a) AirTies test house: b) AirTies test robot 38
Figure 21: Scenarios for the tests: Unizar lab 39
Figure 22: Network scheme used in the tests 39
Figure 23: Comparison between Wi-Fi handovers: a) state-of-the-art; b) reactive handover; c) proactive handover 41
Figure 24: Test setup for measuring the handover delay 44
Figure 25: Box plot of the handover delay, using the same hardware (Intel PRO/Wireless 4965) and different values of inter-beacon time 45
Figure 26: Box plot of the handover delay, using a constant inter-beacon time of 20 ms and different hardware 45
Figure 27: Box plot of the handover delay, using a constant inter-beacon time of 100 ms and different hardware 46
Figure 28: Operation scheme of the SmartAPSelection application when performing RSSI-based load balancing 47

Figure 29: Screenshot of the SmartAPSelection application in BALANCER mode 48
Figure 30: Operation scheme of the SmartAPSelection application when including the FF 49
Figure 31: Screenshot of the SmartAPSelection application in FF mode 49
Figure 32: Scheme of the detection and its integration within the Wi-5 framework 50
Figure 33: Operation scheme of the SmartAPSelection application when performing Detector mode 51
Figure 34: Initial status of the STAs 51
Figure 35: One STA has been moved to the VIP AP 52
Figure 36: A second STA has been moved to the VIP AP 52
Figure 37 : Scheme of the Click Modular Router elements used by the Wi-5 Agents 54
Figure 38 : Measurement environment, including the switches/hubs, the traffic generator, the controller, the DHCP server and the traffic sniffer that captures the traffic 54
Figure 39 : Traffic capture of statistics monitoring traffic 55
Figure 40 : Content of the packets including statistics information, exchanged between the Wi-5 controller and a Wi-5 agent 56
Figure 41 : Capture of the packets including statistics information, exchanged between the Wi-5 controller and a Wi-5 agent 57
Figure 42 : Traffic capture of statistics the control traffic exchange when the *distance in dBs* application is running 58
Figure 43 : Capture of the packets exchanged between the Wi-5 controller and a Wi-5 agent during the running of the *distance in dBs* application 59
Figure 44 : Traffic capture of statistics the control traffic exchange when the scanning application is running 60
Figure 45 : Capture of the packets including monitoring information of the environment, exchanged between the Wi-5 controller and a Wi-5 agent 61
Figure 46: Meaning of the parameters of the *ChannelAssignment* application 63
Figure 47: Delays of the scanning using a virtual machine for the controller 64
Figure 48: Delays of the scanning using a Raspberry Pi for the controller 65
Figure 49: Structure of a normal 802.11 frame 67
Figure 50: Frame aggregation schemes in 802.11n: a) A-MSDU; b) A-MPDU 67
Figure 51: Structure of an A-MPDU 802.11 frame with many subframes 68
Figure 52: Efficiency improvement in 802.11 when using aggregation schemes 68
Figure 53: Test Scenario with 1 AP 70
Figure 54: A single AP shared by 2N of users. a) total throughput of the *N* TCP users; b) average one-way latency experienced by the N VoIP users; c) packet loss rate 71
Figure 55: A single AP shared by 10 users. Delay histogram of VoIP packets: a) No aggregation; b) AMPDU aggregation 72
Figure 56: A single AP shared by a number of users, using EDCA priorities: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) packet loss rate 73

Figure 57: A single AP shared by 10 users, using EDCA priorities. Delay histogram of VoIP packets: a) No aggregation; b) AMPDU aggregation 74

Figure 58: A single AP shared by a number of users, using a limit of 8 kB and 16 kB for the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users ; c) packet loss rate 75

Figure 59: A single AP shared by a 10 of users. Delay histogram of VoIP packets using a limit of a) 8 kB and b) 16 kB for the size of the AMPDU. 76

Figure 60: Test Scenario with 16 APs 77

Figure 61: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) Total throughput of the TCP users; b) Average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS not used 78

Figure 62: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. No EDCA priorities are used. RTS/CTS not used........ 79

Figure 63: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. No EDCA priorities are used. RTS/CTS is activated 80

Figure 64: A network including 9 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS is disabled 81

Figure 65: A network including 4 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS is disabled 82

Figure 66: Scheme of the operation of *ShowMatrixOfDistancedBs* application 89

List of Tables

Table 1. Handover delays for cases with different numbers of STAs [ms] 42
Table 2. Effect of α parameter: Average number of handovers per tour 43
Table 3. Effect of hysteresis time: Average number of handovers per tour... 43
Table 4. Parameters of the environment scanning application’s traffic 61
Table 5: Handover delay: percentage (Acc) of times that handover time can reach a time value. 62
Table 6. Parameters of the ns3 simulations of frame aggregation 69

Glossary

AP	Access Point
CAIDA	Center for Applied Internet Data Analysis
CRC	Cyclic Redundancy Check
CSA	Channel Switch Announcement
DFS	Dynamic Frequency Selection
DHCP	Dynamic Host Configuration Protocol
D-ITG	Distributed Internet Traffic Generator
DOI	Digital Object Identifier
DS	Differentiated Services
DSCP	Differentiated Service CodePoint
EDCA	Enhanced Distributed Channel Access
FCS	Frame Check Sequence
FF	Fittingness Factor
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IRTF	Internet Research Task Force
ISO	International Organization for Standardization
LTE	Long Term Evolution
LVAP	Light Vırtual Access Point
MAC	Media Access Control
MMORPG	Massively Multiplayer Online Role Playing Game
MPDU	MAC Protocol Data Unit aggregation
MSDU	MAC Service Data Unit aggregation
NFV	Network Function Virtualisation
ONF	Open Networking Foundation
PHY	Physical Layer
PLCP	Physical Layer Convergence Protocol

QoE	Quality of Experience
QoS	Quality of Service
RAT	Radio Access Technology
RF	Radio Frequency
RFC	Request For Comments
RRM	Radio Resource Management
RTP	Real Time Protocol
RTS/CTS	Ready To Send / Clear To Send
SDN	Software-Defined Network
SDNRG	Software-Defined Networking Research Group
SDWN	Software-Defined Wireless Network
SOHO	Small Office / Home Office
SSID	Service Set Identifier
STA	Wi-Fi Station
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
WMM	Wi-Fi Multimedia

Executive Summary

This deliverable presents the final version of the specification for the mechanisms included in the Wi-5 Access Points (APs), which make them able to run the Smart Access Point Solutions including resource management algorithms as dynamic channel allocation, load balancing and power control. The seamless handover is explained in detail and it also includes a series of simulations aimed at studying the possibility of performing centrally controlled coordination of the frame aggregation functionalities available in 802.11n and 802.11ac. The integration with the coordination entities of the Wi-5 architecture (*i.e.,* the Wi-5 controller) and the interface with performance monitoring mechanisms are also defined here.

After the Introduction, the Literature Review and a global view of the implemented Wi-5 architecture are provided in order to understand the context. Then, the next section of this deliverable (section 4) is devoted to explaining the functionalities enabling all the Wi-5 features, with detailed information about their implementation and the advances with respect to previous versions reported in Deliverables D3.2 and D3.3. These functions rely on the monitoring mechanisms defined in Deliverable D3.1. This section includes:

- The framework being used for the implementation of the Wi-5 functionalities, based on the use of Light Virtual APs (LVAPs).
- The horizontal handover scheme, integrating multi-channel APs with the LVAPs approach. This includes extensive tests of the handover latency, illustrating that they can really be *seamless, i.e.,* not disrupting the ongoing connections.
- Different applications including Channel Assignment, Mobility Management (in a reactive and a proactive way) and Load Balancing based on RSSI (Received Signal Strength Indicator), FF (Fittingness Factor) and also considering the services being run by the users.

Another section (section 5) details the results of a battery of measurements of the delays incurred by the system.

Finally, a simulation environment is used in order to test different ways of performing a coordinated control of the frame aggregation mechanisms of 802.11. These results are reported in section 6.

Two innovative aspects can be remarked: firstly, the development of a method able to proactively manage the mobility of the users, also combining this with load balancing in real time. The obtained results show that running these two functionalities at the same time is possible, even using a single-board computer as a controller. This avoids the so-called *"sticky client"* problem, which happens when a client remains connected to the original AP it associated with, rather than moving to a nearby one that would be a better choice for it. And second, the proposal of a central coordination of frame aggregation, which can provide a significant improvement in efficiency, respecting the real-time requirements at the same time.

1 Introduction

1.1 Background to project

During the last few years we have witnessed an important increase in the use of wireless devices (*e.g.,* smartphones, laptops and tablets), and the subsequent deployment of extensive wireless networks in areas such as business centres, airports, malls, campuses or even entire European cities. 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 to the IEEE 802.11 standard in order to enhance its performance.

This situation has boosted the emergence of solutions that include a set of coordinated Wi-Fi Access Points (APs from now), usually known as "Enterprise Wi-Fi." However, these solutions are usually proprietary, closed and costly, which in most of the cases makes them infeasible for many organisations. Some of these solutions also lack the required flexibility, scalability, and dynamism in order to optimise the utilization of Wi-Fi networks and to alleviate spectrum congestion. In this context, the scientific community is looking for proposals for inter-AP coordination, enabling advanced features such as load balancing, frequency planning or power control, while making use of low-cost hardware and open software.

At the same time, real-time services with tight latency constraints are becoming ubiquitous. For example, VoIP (Voice over Internet Protocol) 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 desktops, but are also run on tablets or even smartphones. These real-time services share the same connection with "traditional" applications, such as e-mail and Web browsing, but have different requirements (tight latency constraints) in order to meet the Quality of Experience (QoE) demands.

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), and smartphone and tablet users walk while using these real-time services. The clients (stations - STAs from now on) that connect to these wireless networks implement their own algorithms for selecting the AP to associate with. These mechanisms try to select the best candidate among the different Wi-Fi APs they can *see* in the wireless environment. For this task, they rely on the available information, *i.e.,* the power level of the beacons received from each AP in the neighbourhood. As a result, these decisions are only based on local knowledge of each STA. Hence, they are totally uncoordinated and result in situations like *e.g.,* the so-called "sticky client problem," a term that is used to describe a STA that remains connected to an AP that it knows, even if the distance has been significantly increased due to the movement of the user [1]. This results in a rate reduction that also harms the rest of the STAs connected to the AP [2]. Another problem is that the normal 802.11 handover has a delay of several hundreds of milliseconds [3], which may be unacceptable for applications with real-time constraints.

The *What to do With the Wi-Fi Wild West* (Wi-5) project has combined 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 have been integrated at different layers of the protocol stack with the aim of meeting a demanding set of goals such as seamless handover, reduced congestion, increased throughput and energy efficiency. The project has developed a variety of different solutions, which have been made available in academic publications, in addition to other dissemination channels for industrial exploitation and standardisation.

1.2 Scope and structure of the deliverable

This deliverable presents the final version of the specification of the mechanisms included in the Wi-5 APs to perform different optimisations of the radio environment. It also includes a study of the policies to be employed when using packet grouping between the AP and the end device, as defined in 802.11n and 802.11ac. The integration with the coordination entities of the Wi-5 architecture (the Wi-5 controller), and the interface with performance monitoring mechanisms are also defined.

Therefore, this report summarises the work carried out regarding the Smart Access Point Solutions during the third period of the Wi-5 project (January 2017-April 2018). This includes the radio configuration capabilities allowing the use of the resource management algorithms, namely dynamic channel allocation, load balancing and reactive and proactive mobility management. The mechanisms for including packet/frame grouping are also included. These functionalities consider a scenario where all the APs are managed by a controller, which has a global view of the network.

Beyond the Introduction, the rest of this document is structured as follows. An updated Literature Review is provided in section 2, related to the topics covered here. Next, a global view of the Wi-5 architecture is presented in section 3, including the smart AP solutions described here and the cooperative functionalities developed as part of WP4. Section 4 is devoted to explaining the functionalities enabling all the Wi-5 Smart AP Solutions, with additional information about their implementation. New tests with the horizontal handover scheme, integrating multi-channel APs with the Light Virtual APs (LVAPs) are presented, comparing the performance of the reactive mobility management, already presented in Deliverable D3.3, with a new proactive scheme, able to perform different functions (load balancing and mobility management) within a single application.

Section 5 reports the results of a battery of tests that have been carried out in order to measure the delays incurred by the system, *i.e.,* the delays required to gather the monitoring information and the delays when dispatching configuration changes to the equipment.

In section 6, the simulation software that has been developed for measuring the savings of the centrally managed frame aggregation mechanism is presented and used to perform an extensive battery of tests. The document ends with the Conclusions in section 7.

1.3 Relationship to other deliverables

The five use cases and their requirements were reported in Deliverable D2.3, including the different scenarios, the applications and the services considered. The functionalities included in the present document will be in charge of enabling the accomplishment of these requirements.

The final version of the Wi-5 architecture is presented in Deliverable D2.5 “Final Wi-5 architecture,” which provides a global view of the whole set of solutions deployed, including the Smart AP Solutions explained in this deliverable. The architecture is described according to the ISO-IEC-IEEE 42010 standard, and the requirements are presented in the context of the business and stakeholders' requirements.

Some features regarding Smart AP Solutions were already reported in the previous versions of this deliverable (Deliverable D3.2 “Specification of Smart AP Solutions version 1”, and Deliverable D3.3

"Specification of Smart AP Solutions version 2"), so they will not be included again here. For example, the reader can find in Deliverable D3.2 the studies carried out with Simplemux, or in Deliverable D3.3 the study of the effects of packet aggregation on subjective quality. In some other cases, when the functionalities have experienced a significant update, new versions of some parts of Deliverables D3.2 and D3.3 are included here.

The Wi-5 monitoring functionalities were reported in Deliverable D3.1 "Definition of the performance monitoring mechanisms", which not only included the features able to monitor the wireless environment, but also the tools for automatically detecting real-time services.

The Cooperative Functionalities, being deployed in WP4, are explained in detail in Deliverable D4.3 "Final Specification of Cooperative Functionalities". 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.

The document Deliverable D5.1 "Testbed description and definition of the tests" included a description of the platforms for the evaluation tests, which will be carried out using the tools presented in the current document. Finally, Deliverables D5.2 and D5.3 include the "Integration results" and the "Evaluation results" respectively.

2 Literature Review

This section updates the literature review provided in the deliverables D3.1, D3.2 and D3.3, including new developments found in the state of the art that address the considered features and functionalities.

2.1 Architectural proposals for Software-Defined Wireless Networks (SDWNs)

The SDN (Software-Defined Networking) concept aims to separate the network control and data plane, allowing an abstraction of the underlying technology for applications and services. It is becoming increasingly popular, since it provides a coordinated programmable interface for network managers. In combination with NFV (Network Function Virtualisation), it is becoming a concept considered not only in wired networks, but also in wireless environments [4].

The Software-Defined Networking Research Group (SDNRG) of the Internet Research Task Force (IRTF) defined the layers and architecture terminology for SDN systems [5]. Even though the document did not pretend to standardize any layer or interface, it provided a reference of the approaches that can be taken when defining SDN architectures. The Open Networking Foundation (ONF) proposed another architecture [6], which is more service-oriented, whereas the one proposed by the IRTF has a more functional view [7]. We have followed the IRTF approach in our work.

The architecture proposed by the IRTF (see Figure 1) consists of: an *Application Plane* where applications that define network behaviour reside; a *Network Services Abstraction Layer* that provides access from applications of the control, management; the *Control Plane* is responsible for making decisions on how packets should be forwarded by one or more network devices and pushing such decisions down to the network devices for execution; the *Management Plane* is in charge of monitoring, configuring, and maintaining network devices, *e.g.,* making decisions regarding the state of a network device; and the *Device and resource Abstraction Layer* abstracts the resources of the device's forwarding and operational planes to the control and management planes.

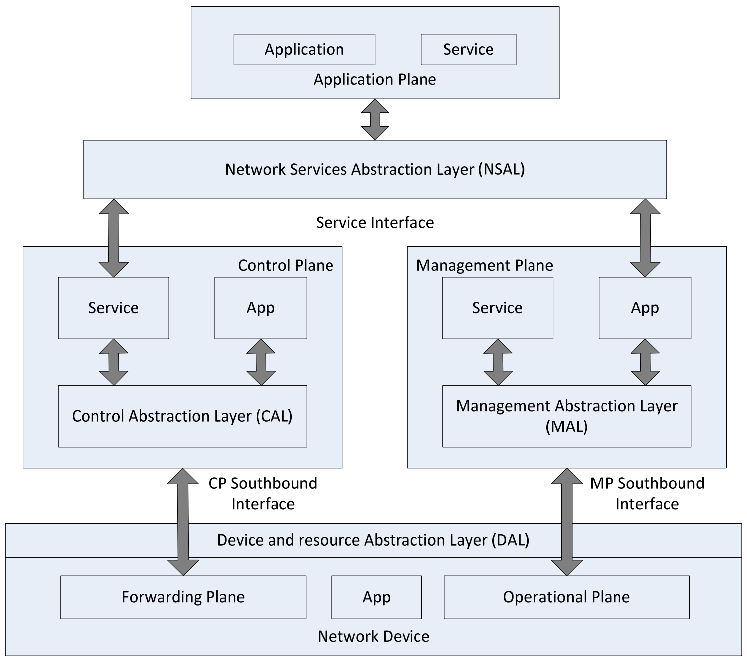


Figure 1: SDN Layer Architecture according to RFC 7426

Different high-level languages have been proposed for SDNs [8], [9], but they are mainly designed for wired networks based on the OpenFlow protocol [10] for data plane control. However, the work in [11] surveyed a set of abstractions for wireless networks, as OpenFlow does not itself capture all the issues appearing in wireless scenarios. For example, the "flow" abstraction proposed by OpenFlow does not reflect the stochastic nature of wireless links (which are very different from an Ethernet connection), so it does not suffice for network management in wireless environments.

2.2 Use of Virtual Wi-Fi APs in SDWNs

SDWN architectures need to integrate the capacity to manage the specific issues that appear in wireless scenarios. For this aim, the *Light Virtual Access Point* (LVAP) abstraction was first proposed in [12] as a way for providing flexibility in wireless networks. When a STA connects for the first time, a central controller creates an LVAP for it and assigns it to a physical AP (the most suitable one for the STA). An LVAP is just a tuple of 4 fields: a MAC (Media Access Control) address and an SSID (Service Set Identifier) to be used to communicate with the STA, the real MAC of the STA and its IP address. Therefore, a physical AP can host a number of LVAPs, and it will use a different one for communicating with each client, *i.e.,* the same AP can send frames with different source MAC addresses, depending on the destination STA.

As illustrated in Figure 2, the controller can move the LVAP from one physical AP to another, according to the movement of the STA (the LVAP *travels* with its STA). As a consequence, the STA will always *see* the same AP (it is always receiving frames from the same MAC and SSID), so it will not resort to its algorithms for finding other APs. Therefore, centralised control of the assignment of STAs to the APs can be performed. It should be noted that this does not require any modification on the STA, which runs standard IEEE 802.11.

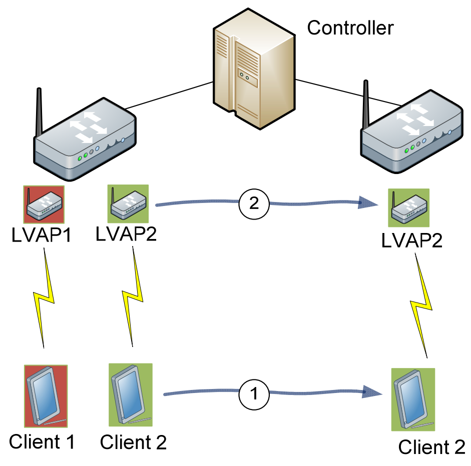


Figure 2: Scheme of a Wi-Fi network using LVAPs

A distributed solution using LVAPs was introduced in [13], where a protocol for the direct exchange of information between APs was also proposed. One limitation of this proposal is that, due to the absence of a central controller, each AP has to build a list of neighbouring APs by itself.

An LVAP-based solution including a central controller and a set of low-cost *OpenWrt*[1] APs was presented in [14], using two Southbound protocols that facilitate communication between the controller and the APs: OpenFlow, in charge of controlling the internal switch of the AP, and a new protocol

[1] OpenWrt, Wireless freedom, https://openwrt.org/

called Odin, that takes care of the wireless part. The central manager is the SDN controller that combines both functionalities, and is in charge of creating the LVAP for each STA that sends an association request. The controller also hosts a number of algorithms for radio resource management that run as applications on top of it.

More research has recently been published which also uses the concept of LVAP, and can provide fast handovers: in [15] a scheme called BIGAP is presented, which is based on the use of a mechanism below the MAC layer for handover, exploiting the Dynamic Frequency Selection (DFS) capability in 802.11. In a similar way to the solutions developed by Wi-5, it does not require any modifications to the STAs. However, it requires the support of 802.11n [16] or 802.11ac [17], including the IEEE 802.11h [18] amendment. In addition, it requires the existence of a sufficiently large number of available RF channels in order to allow the possibility of different channels being assigned to co-located APs. This is feasible in the 5GHz band (there are 25 channels with Dynamic Frequency Selection, DFS), but it may not be in the 2.4 GHz band.

Finally, it should be noted that the literature related to monitoring the wireless environment and service detection is summarised in Deliverable D3.1 and is not included here.

2.3 Frameworks Providing Fast Handovers

The authors of [19] proposed a scheme based on cognitive management of WLANs, integrating an SDN-based distribution system, which allows checking the resource availability in order to select the best AP. It also includes a QoS-aware handovers and proactive forwarding table updates in an SDN-based distribution system. The scheme is based on 802.11r, and a modification of the STA is required. In [20] a channel scanning scheme for 802.11 WLANs was also presented. Here, each AP was equipped with two wireless network interfaces, one of which is dedicated to broadcast beacons in other channels, thus avoiding the need for STAs to perform active scans when they need a handover.

Some other frameworks that jointly consider SDWN and a virtual abstraction of the underlying network are CloudMAC [21], Anyfi [22] and Odin [14]. In CloudMAC and Anyfi all the wireless frames are passed to an entity called the *Access Controller,* whereas in Odin they are translated into Ethernet frames by the AP, as usually happens in Wi-Fi networks in which the APs are connected to each other, and to the Internet, by a wired network. However, Odin presents two main limitations for fast handovers: firstly, it assumes that all the APs operate in the same channel, which makes it impossible to perform channel planning. Secondly, there is a scalability problem, as broadcast beacons cannot be employed: the AP has to send unicast beacons with a specific MAC to each STA. These two limitations were alleviated in our previous study [23], where a new handover mechanism was studied, including the possibility of using different channels. In addition, the beacon rate was reduced, except during a short period after the handover, thus improving the scalability of the system.

2.4 Resource Management in Wi-Fi WLANs

Radio resource management (RRM) approaches aim to design appropriate strategies and algorithms for configuring wireless transmission parameters in order to efficiently utilise limited radio resources. In the context of Wi-Fi networks, the RRM solutions in Wi-5 focus on three main issues: smart channel selection, dynamic transmit power control and load balancing. A literature review of these functionalities was provided in previous deliverables (D3.2, D3.3). In the present section we survey some advances that have been significant for the research carried out.

As far as load balancing is concerned, in Wi-Fi networks STAs independently select the APs to connect to. Therefore, the traffic in WLAN networks can be unevenly distributed, which leads to inefficient use of the available resources. Load balancing techniques try to solve this problem by better distributing the traffic load in the network. In [24] a survey of both network- and wireless-station-based solutions was presented. In [25] a combined metric consisting of the number of stations associated, and mean and instantaneous Received Signal Strength Indicator (RSSI from now) for each of the clients associated with this AP is used. A similar AP-assisted approach was proposed by Cisco [26], to give associated client information about the load through beacons. Both these solutions require modifications on the client side. Murty *et al.* [27], and Chandra and Bahl [28], achieve fair load balancing between APs through a centralised management approach that aggregates AP workload, which requires changes in the wireless network infrastructure.

As one of the requisites considered in Wi-5 is that it should work with unmodified STAs, we have followed an approach in which the controller tries to evenly distribute the STAs between the APs.

2.5 Packet/Frame Grouping at Different Layers

As reported in [29], which used a traffic trace at a backbone link from the *CAIDA Anonymized Internet Traces* 2015 Dataset [30], a high number of small packets are present in the public Internet: 33.4% of the packets were 60 bytes or smaller, and 41.4% overall were 200 bytes or smaller. The efficiency in wireless networks is very low for small packets. As explained in [31], when UDP and TCP packets of different sizes are sent following the 802.11 standard, a significantly low efficiency was observed when small packets (some tens of bytes) were sent, since the time required for medium access is in the same order of magnitude as the time required for the actual transmission of the data. In Wi-Fi networks, frame aggregation was added as an option to 802.11n, and it is compulsory in 802.11ac.

Two frame aggregation mechanisms are considered in the standard: AMSDU (Aggregated MAC Service Data Unit aggregation) and AMPDU (Aggregated MAC Protocol Data Unit aggregation). In the former, a single FCS (Frame Check Sequence) is included, which protects the integrity of the whole multi-frame. Therefore, if some bits are corrupted all the information has to be retransmitted. In contrast, in AMPDU each sub-frame carries its own FCS which permits individual retransmissions of the corrupted sub-frames. Therefore, the latter method has become more successful: in fact, in 802.11ac every frame must have an AMPDU format, and very big frames (up to 1 MB) can be sent.

Frame aggregation can provide significant improvements in terms of efficiency: in one of the first works on the topic [31], simulations showed a significant efficiency increase, and a better performance of AMPDU, which was emphasized when the packet-error rate was high. In [32], an optimisation method was proposed to dynamically adjust the number of AMPDUs according to the sub-frame size, the maximum aggregation level allowed, and the instantaneous channel bit-error-rate. However, as a counterpart of the throughput increase, when a STA gains access to the medium and transmits a long frame including a number of sub-frames, the rest of the STAs connected to the AP have to wait until the end of the transmission. This delay can be significant, especially if the STA is far from the AP and it is transmitting at a low rate. This results in delay, and also in jitter, as some frames will wait more time than others. All in all, a trade-off appears: frame aggregation is positive for bandwidth-demanding services but it can negatively affect services with real-time constraints.

2.6 Weaknesses of the previous work

A number of gaps have been identified in the existing literature, which have been addressed by the Wi-5 project. We will next summarise them.

First, many of the existing works using SDWN solutions to provide fast handovers for 802.11 WLANs (see sections 2.2 and 2.3), were only conceived for scenarios in which all the APs operate in the same channel. We considered that this was a severe limitation, which made it impossible to perform channel planning. In addition, a scalability problem appeared, as broadcast beacons could not be employed: the AP has to send unicast beacons with a specific MAC to each STA.

Second, regarding resource management (section 2.4), the use of reactive handoff schemes had some limitations in terms of scalability, especially if a number of STA move together. In addition, the existing solutions were not able to provide a joint set of functionalities, *i.e.,* the solutions providing a good mobility management were not considering load balancing at the same time. As a result, contradictory decisions could be taken by the different functionalities, resulting in undesired behaviours as *e.g.,* a "ping-pong" effect of a STA moving between two APs.

Third, although many works have studied the benefits of frame aggregation in 802.11 (see section 2.5), they were not focused on the trade-off between the improved throughput and the additional latency. Different applications with different requirements may coexist in the same WLAN: while some of them are aimed at maximising the throughput, others have a very different objective: minimising the added delay in order to meet the real-time constraints of the service.

3 Global view of the implemented architecture

3.1 Global view of Smart Access Point Solutions

This subsection provides a brief summary of the developed solutions that will be presented in more detail in section 4. The Wi-5 Smart AP Solutions are a set of improvements developed in WP3 considering a centralised cooperation between APs.

The functionalities can be summarised as:

- Monitoring, including identification of flows with special requirements (*e.g.,* real-time constraints) and scanning of the radio environment in order to obtain accurate information about the interference level on each channel, the power being sent by the STA, etc. They will be explained in detail in Section 4.1.2.
- Radio Resource Management actuation. This includes mechanisms to optimise performance of the algorithms being deployed in the Wi-5 architecture as *a)* Dynamic Channel Allocation, looking for an optimal distribution of the channels of the APs, in order to reduce the interference; *b)* Load Balancing, looking for an optimal distribution of the users between the APs; *c)* other functions, *e.g.,* power Control, are also possible, although they have not been implemented in the testbed.
- Seamless Handover is an essential functionality as it makes it possible to use other functionalities without reducing the quality experienced by the users: for example, Load Balancing implies that certain STAs will be moved from one AP to another based on smart AP selection; or the management of the users' mobility also requires a seamless handover scheme.
- Frame aggregation, considering smart scheduling policies enabling a better use of the aggregation features already included in 802.11n and subsequent versions. This alleviates the airtime inefficiency caused by the STAs requesting the shared channel before being able to send actual data. For legacy devices not implementing aggregation at Layer 2, multiplexing packets at Layer 3 can also be a solution (see D3.3).

A proof-of-concept implementation of the Wi-5 architecture has been built using open-source software and low-cost commercial equipment. Its main aim is to show that advanced functionalities for coordination in a wireless scenario can be provided. The following design objectives have been accomplished:

- The solution has to work with low-cost APs, *i.e.,* the ones usually deployed by operators in the households of their customers.
- The use of different channels in the APs should be possible.
- It should not require any change in the user terminal.
- The users can move at walking speed.
- The support of services with real-time constraints (*e.g.,* VoIP, online gaming) should be possible.

Figure 3 shows the proposed scheme, which consists of two main elements: the Wi-5 controller, and a set of Wi-5 APs. Several applications can run on top of the controller, which in turn manages a set of Wi-Fi APs by means of the Southbound protocols. The APs are equipped with functionalities that help to optimise the utilisation of the network resources. This includes the monitoring tools, which provide timely information about the wireless spectrum, allowing a set of resource management algorithms to optimise the QoE experienced by the users.

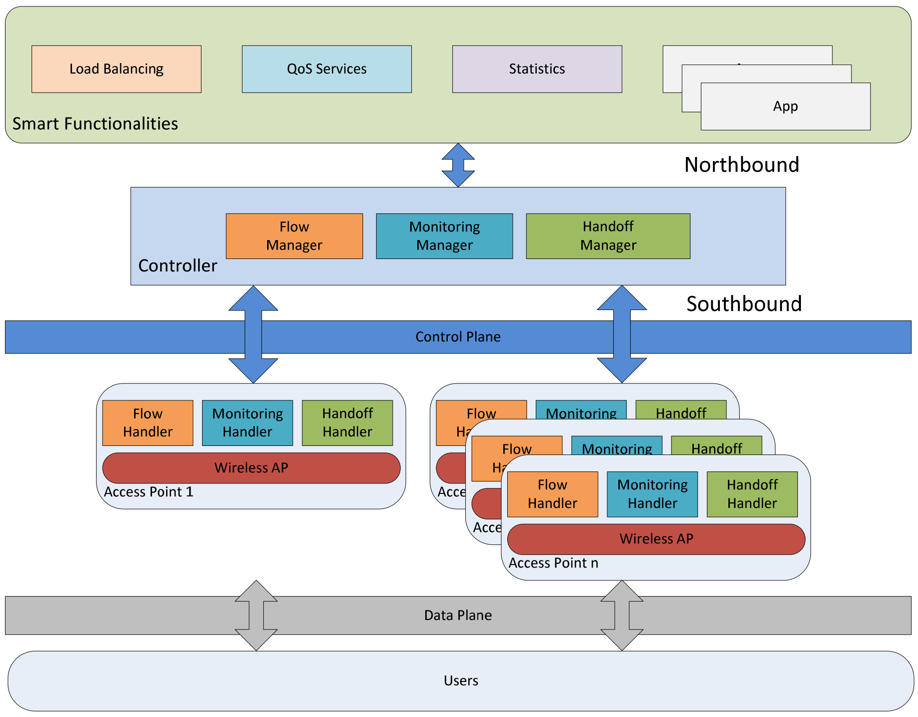


Figure 3: Scheme of the Smart AP functionality implementation

In order to allow the controller to manage the mobility, and to select the best moment for the handover of a STA between APs, a separation between control and data planes has been implemented via SDN. This enables an abstraction of the underlying technology for applications and services. The key feature of any SDN architecture is the *Control Plane*, responsible for defining and enforcing spectrum management and utilisation policies. It provides an Application Programming Interface (API) to the smart functionalities running on top of it. Each of the elements of the controller has a corresponding one in the managed APs:

- the *flow manager* and the *flow handler* are in charge of the flows associated to the users;
- the *monitoring manager* and *monitoring handler* continuously gather relevant information about the wireless environment (scanning) such as *e.g.,* rate, frame size, power, interference levels, air time consumption, etc.;
- the *handoff manager* and the *handoff handler* are in charge of generating the required actions to re-allocate user terminals. They allow the handover between APs (even in different channels) without affecting IP and upper levels. For that aim, they coordinate the generation of Channel Switch Announcement (CSA) beacons when required.

As a consequence of the functionalities provided by the *Control Plane,* enhanced capabilities are provided by the *Data Plane* to the STAs as *e.g.,* seamless handovers in which the IP address of the STA is maintained.

3.2 Global view of Cooperative Functionalities

This section presents a summary of the final version of the algorithms designed to efficiently exploit the use of the radio resources, reducing the interference impact between APs and providing optimised connectivity for each user/flow that is served by an AP in the considered scenarios. A more detailed explanation of the developed algorithms has been included in Deliverable D4.3 *"Final specification of*

Cooperative Access Points Functionalities" [33]. In more detail, several improvements have been developed in the final version of the specification with the aim of achieving the following purposes:

- Defining a RRM algorithm to address interference in Wi-Fi networks by combining both channel assignment and transmit power adjustment techniques. The proposed approach aims to improve the application flow QoS, while at the same time considering the effect of a configuration on the rest of the network.
- Defining a Smart AP selection algorithm that will assist users/flows in selecting the most suitable AP according to the application running on the station and its QoS requirements. This algorithm could also be extended to achieve Horizontal Handover by using QoS metrics, such as the RSSI and the Fittingness Factor (FF) explained in detail in deliverables D4.2 and D4.3, to reflect the wireless user's mobility.
- Defining a Radio Access Technology (RAT) selection strategy that extends the AP selection algorithm towards the Vertical Handover between Wi-Fi and 3G/4G mobile networks.

3.2.1 Radio Resource Management algorithm

The RRM algorithm presented in D4.3 is composed of channel assignment and transmit power adjustment strategies. The channel assignment process is based on an objective function which reduces the magnitude of the interference impact in the whole system. In detail, this strategy allows the Wi-5 controller to find an optimised channel distribution, in terms of interference for the different APs, in a network based on (i) the Wi-Fi system properties (*e.g.,* IEEE 802.11's standard channel characteristics); (ii) the logical network topology (the AP distribution throughout the network); and (iii) the desired resource management criteria (the assigned channels, interference-related QoS, or handover requirements). At the same time, the power adjustment process provides the capability of setting the transmission power of the APs such that the QoS requirements of the flows are met, and the interference level in the network is maintained close to its optimal value defined through the channel assignment process.

3.2.2 Access Point Selection Algorithm

The AP Selection algorithm is an extention of the solution presented in deliverable D4.2 and it is based on a potential game, which allows an efficient distribution of Wi-Fi users among the APs in a network. Potential games [34] are a tool that allows us to perform a distributed optimisation of resource allocation through the convergence to a pure Nash Equilibrium (NE), which is always guaranteed [35]. The main drawback of this tool is the complexity resulting from its implementation on large distributed scenarios such as in enterprise Wi-Fi networks; because players usually require overall information about the remaining players of the network, making the solution not scalable. On the other hand, our strategy of using SDN as a management platform allows us to store all the required information on the centralised SDN controller, so the game can be played at this central control entity. The controller selects the best AP for each application flow required by a Wi-Fi user through a potential game based on the FF concept, which represents the suitability in terms of available QoS between an AP and a certain flow. The inclusion of a potential game in the SDN-based controller together with the proposed suitability concept allows us to achieve improved performance in terms of users' satisfaction compared to the state of the art. With respect to our previous works presented in deliverables D4.1 and D4.2, we can now guarantee an optimal and efficient reallocation of the APs to the flows connected to the network when needed.

3.2.3 Vertical Handover

This algorithm allows Wi-5 to handle Vertical Handovers where wireless users could be moved from a WLAN to a 4G network. Specifically, we propose a novel strategy that matches the most suitable RAT for a certain user based on QoS requirements for his/her ongoing application. Such a match will allow smart use of the limited spectrum resources guaranteeing the users' QoS demands in the most efficient way. The SDN controller provides all the monitoring information needed for our RAT selection and implements a Vertical Handover strategy to allow a complete and efficient integration between LTE and Wi-Fi technologies.

3.2.4 Integration of Monitoring Tools and Radio Configuration with Cooperative Functionalities

The functional blocks defined in the Wi-5 architecture which will implement RRM, AP selection and vertical handover algorithms rely on the monitoring tools included in the Wi-5 APs and in the Wi-5 controller, and explained in detail in section 4.1.2.

These tools allow the correct deployment of the algorithms developed in the context of WP4 in a real-time environment. Specifically, the role of these tools is to provide information on: (i) the power level sensed by each AP at the available channels in the considered frequency bands; (ii) the number of users/flows associated to each AP; and (iii) automatic identification of different kinds of user services in terms of bit rate requirements (*e.g.,* a combined PHY and MAC layer process at the APs). These monitoring mechanisms are essential during the decision-making process at the Wi-5 controller. For instance, monitoring the power levels can support the channel assignment process implemented in the RRM algorithm during the initialisation of the Wi-Fi network, or during a possible reassignment of the channels due to a change of the network status. Moreover, information on the power levels supports the AP selection procedure when a new user/flow tries to join the network. Therefore, the results of these algorithms may trigger the reconfiguration of a certain AP, providing the Wi-5 agent in the allocated AP with the appropriate radio configuration parameters through the southbound API, as illustrated in Figure 4.

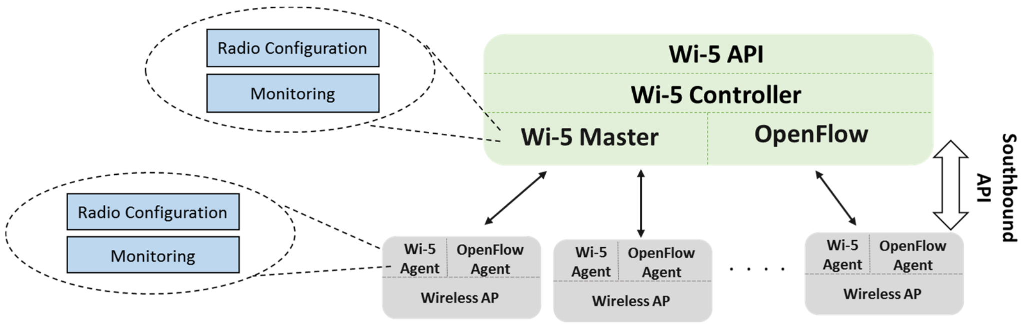


Figure 4: Radio configuration procedures

3.3 Most innovative aspects of WP3

This subsection summarises the most innovative aspects addressed by WP3 of Wi-5. Different gaps in the existing work were identified (see section 2.6), which have been addressed as part of the Wi-5 objectives.

First, the horizontal handover scheme has been improved to integrate multi-channel APs with the LVAPs approach. The new approach has been extensively tested, including extensive measurements of the handover latency, illustrating that they can really be seamless, *i.e.,* not disrupting the ongoing connections, even if the APs are operating in different channels. This handover mechanism is integrated with a set of active and passive monitoring tools and other functionalities, resulting in a solution that is able to provide smart functionalities using low-cost commercial APs in an open source framework.

Regarding resource management, different applications have been developed for the Wi-5 controller, including Channel Assignment, Mobility Management (in a reactive and a proactive way) and Load Balancing based on RSSI, Fittingness Factor and also considering the services being run by the users. This avoids the problem of the "sticky client," which remains connected to the original AP it associated with, rather than moving to a nearby one that would be a better choice for it.

Two different test scenarios (a lab environment in UNIZAR and a test house in AirTies) have been used to compare proactive and reactive handover mechanisms in realistic conditions, and the advantages of the proactive approach have been highlighted. The inter-channel handover delay has been measured with three different devices, using five values for the inter-beacon time, proving that fast and seamless handovers are possible in the scenario, even with low cost off-the-self equipment.

These resource management functionalities have been addressed together, so a joint and coordinated approach is taken when making the decisions caused by mobility management and those derived from load balancing. This avoids the problem of having contradictory decisions taken by different functionalities.

Regarding packet aggregation, current solutions did not consider the kind of service in order to limit the maximum additional delay. Different queues are used according to the service, with different priorities, but they do not consider any mechanism to limit the additional delay caused by aggregation. As a result, the savings in terms of bandwidth and efficiency may come with the counterpart of a reduction of the subjective quality perceived by the users. A simulation environment has been created, where three solutions, based on prioritisation, limiting the maximum AMPDU size and using a central coordination of the aggregation have been studied and discussed. To the best of our knowledge, this was the first study proposing a centralised coordination and control of 802.11 frame aggregation functionalities. This allowed a joint approach, providing a low latency for applications with real-time constraints, and a maximum throughput to other applications as *e.g.,* file downloads.

4 Specification of Wi-5 Smart AP Solutions

The Wi-5 architecture includes a series of improvements for AP cooperation (be it intra or inter-operator). Therefore, a number of functionalities and procedures have been defined in WP3 in order to enable the different considered optimisations. This section details the final design and definition of these features. These functionalities are exploited by the cooperative functionalities detailed in D4.3.

4.1 Scheme of the implementation

4.1.1 Detailed explanation of the implementation

In this subsection, we will provide a more detailed view of the final implementation. For that aim, the open source framework presented in the previous deliverables (D3.1, D3.2 and D3.3) has been taken as a basis for evolving the proof-of-concept implementation[2]. As a summary, the implementation was able to perform seamless handovers between Wi-Fi APs in different channels [13], leveraging on the use of LVAPs. This is a summary of the new functionalities that have been added:

- Different scanning mechanisms, based on the use of a second (auxiliary) wireless interface, have been included. This interface (a low cost Wi-Fi dongle) is able to measure different network parameters (*e.g.,* signal, noise, rate, etc.) of each STA.
- Different handover mechanisms have been implemented as individual applications (smart functionalities). In addition to the original reactive handover, a proactive one has been implemented (they will be compared in next sections). Different metrics can be used to trigger these handovers [36].
- New applications have been built, which run in the controller, and are able to perform an optimal channel assignment, and to combine mobility management with load balancing. Integrating the detection of certain services of interest is also possible, allowing a special management of the corresponding flows.

The implementation is illustrated in Figure 5. Following the SDN approach, *control* and *data* planes have been separated. *OpenvSwitch*[3] is installed in the Wi-5 APs, making their internal switch behave as an OpenFlow switch managed by Floodlight Controller[4]. In addition, *Click Modular Router* [37], with a specific module (Wi-5 agent [14]) is run in the AP, allowing the interaction with the controller to directly manage the traffic. More details about the setup implementation, *e.g.,* the ports used, the kind of Linux interfaces, can be found in Deliverable D3.3.

[2] All the software components of the proposed solution are available at https://github.com/Wi5
[3] Open vSwitch, http://openvswitch.org
[4] Floodlight SDN Controller, http://www.projectfloodlight.org/floodlight

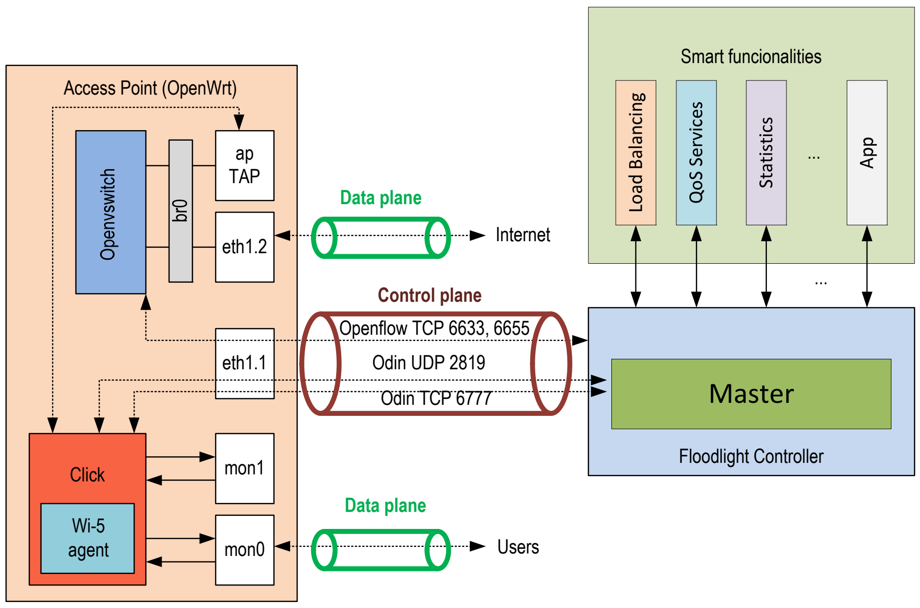


Figure 5: Elements of the implementation

4.1.2 Monitoring functionalities

4.1.2.1 Internal and external monitoring

In order to observe the correct performance of the smart functionalities, a set of monitoring tools has been included in the APs and the controller. The internal wireless interface of the AP is set to monitor mode (`mon0`), so it captures all the frames in its channel (we will use the term *internal* monitoring for this). These frames go to the Wi-5 agent, which treats them according to their nature:

- Data frames targeted to one of the LVAPs hosted by the physical AP are sent to a Linux TAP interface called `ap`.
- Control frames (*e.g.,* association requests) are forwarded to the controller using the wired connection (Odin protocol [14]).
- Other frames in the same channel, but not corresponding to the AP, are just taken into account for monitoring purposes, and discarded.

The auxiliary wireless interface (`mon1`) is used for off-channel monitoring (*i.e.,* in other channels, called *external* monitoring), following the requests of the controller.

4.1.2.2 Passive monitoring

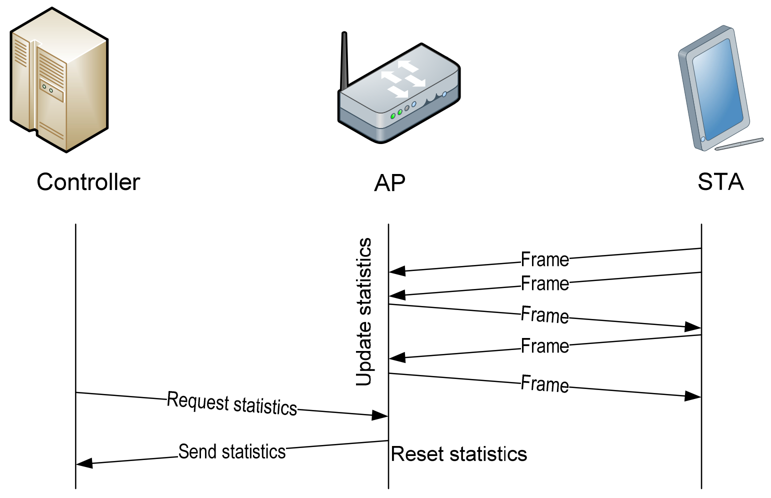
The monitoring process obtains all the information of each frame received by the AP, available in the Radiotap[5] header (for instance: *rate*, *noise*, *signal*, *channel* and *packet length*). Once properly averaged, this information permits functionalities such as:

- Measuring airtime usage by each of the STAs associated to each AP (using the main interface), and the airtime consumed by non-Wi-5 APs and STAs (using the auxiliary interface).
- Performing a smart AP allocation when a new STA tries to join the network.
- Triggering reactive functionalities: *e.g.,* a reconfiguration of a certain AP's channel, etc.

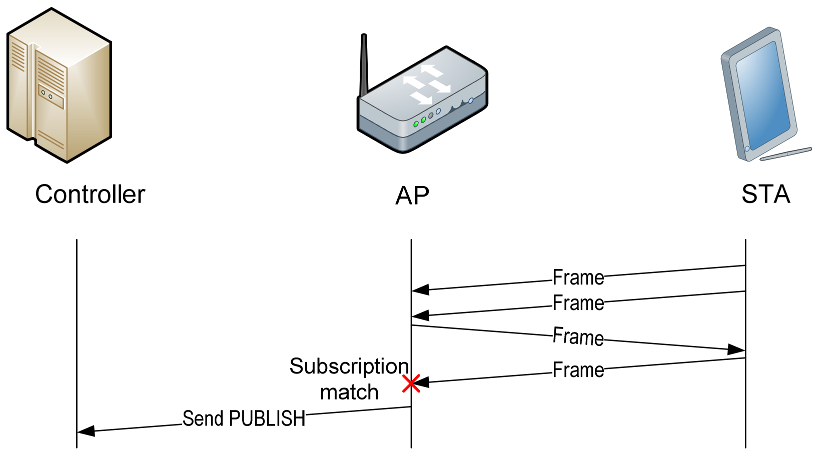
[5] Radiotap, http://www.radiotap.org

To support the proactive functionalities in the controller (Figure 6a), every smart application can cycle between sleeping and performing some activity. When the controller needs some information, it requests the corresponding statistics. The AP sends them and resets all the counters.

In the reactive model (Figure 6b), the smart functionalities use a content-based publish-subscribe mechanism to be aware of certain events. For that aim, applications register subscriptions with the controller*,* which subsequently registers them in the APs. When an AP receives a frame, it performs a check to see if any of the collected statistics corresponding to the frame matches any of the subscriptions that have been registered with it. If there is a match, the AP sends a message to the *controller*, indicating the source address of the frame that triggered the notification. The *controller* then forwards the event to the corresponding application(s).



(a)



(b)

Figure 6: Monitoring for a) proactive; b) reactive applications

As examples of passive monitoring, we present a set of captures obtained with the application *DemoStatistics*[6]. It presents a menu to the user, who can get different statistics from the APs and the

[6] Wi-5 *DemoStatistics* application, see https://github.com/Wi5/odin-wi5/wiki/Application-DemoStatistics

connected STAs. In Figure 7, we present the output of an application able to capture the status of the AP; in Figure 8, the internal statistics (using the wireless interface of the AP `mon0`) are shown.

```
[DemoStatistics] =======Internal statistics=======
[DemoStatistics] Agent [0]: /192.168.1.13
[DemoStatistics]  Txpower: 10 dBm
[DemoStatistics]  Channel: 1
[DemoStatistics]  Last heard: 389 ms ago
[DemoStatistics]
[DemoStatistics] Agent [1]: /192.168.1.15
[DemoStatistics]  Txpower: 10 dBm
[DemoStatistics]  Channel: 6
[DemoStatistics]  Last heard: 1893 ms ago
[DemoStatistics]
[DemoStatistics] Agent [2]: /192.168.1.14
[DemoStatistics]  Txpower: 10 dBm
[DemoStatistics]  Channel: 6
[DemoStatistics]  Last heard: 142 ms ago
```

Figure 7: Monitoring: output of the application using passive monitoring for obtaining detailed information of the status of an AP

```
[DemoStatistics] =================================
[DemoStatistics] <<<<<<<<< Rx statistics >>>>>>>>>
  Uplink station  MAC: 30:07:4D:9E:8E:C8
IP: 192.168.2.215
          num packets: 4175
          avg rate: 51261.7964072 kbps
          avg signal: -30.5969219499 dBm
          avg length: 295.96239521 bytes
          air time: 213.734666667 ms
          init time: 1519733522.191784021 sec
          end time: 1519733574.866460499 sec
[DemoStatistics] <<<<<<<<< Tx statistics >>>>>>>>>
  Downlink station MAC: 30:07:4D:9E:8E:C8
IP: 192.168.2.215
          num packets: 3849
          avg rate: 54000 kbps
          avg signal: 10 dBm
          avg length: 702.74564822 bytes
          air time: 400.721185185 ms
          init time: 1519733525.415637841 sec
          end time: 1519733574.344232353 sec
```

Figure 8: Monitoring: output of the application using passive monitoring for obtaining detailed information of the STAs connected to an AP

Finally, in Figure 9, we present an example of the external statistics (obtained with the auxiliary interface `mon1`) of the different STAs connected to all the APs in the system. All these statistics are available in the controller, which can use them as an input for its cooperative algorithms. It distinguishes between four different kinds of device:

- Wi-5 LVAPs.
- Wi-5 STAs.
- Non-Wi-5 APs in the neighbourhood.
- STAs associated to non-Wi-5 APs.

In order to distinguish between the APs and the STAs, we use the *DS status* flag of the *Frame Control Field* of the 802.11 header. By using it, we can know if the MAC addresses corresponds to a STA or an AP. We can also use the information we have about the associated devices (the database of associated

STAs available in the controller), to know if a STA is associated to a Wi-5 AP, or if it is an AP that does not belong to the Wi-5 network (using the information about LVAPs available in the controller).

```
[DemoStatistics]
  Select agent [0-2]: 0
  Select channel to scan [1-11]: 2
  Select time to scan (msec): 2000
[DemoStatistics] <<<<< Scanning in channel 2 >>>>>>
  Station MAC: 76:A0:26:C1:E1:AA
          num packets: 1
          avg rate: 1000 kbps
          avg signal: -70 dBm
          avg length: 126 bytes
          air time: 1.008 ms
          init time: 1519733747.339689262 sec
          end time: 1519733747.339826638 sec
          AP of client: unknown
          Channel of AP: unknown
          Code: non-Wi-5 STA

  Station MAC: DC:EF:09:E6:9C:DB
          num packets: 16
          avg rate: 1000 kbps
          avg signal: -54.1879014765 dBm
          avg length: 162 bytes
          air time: 20.736 ms
          init time: 1519733746.736829563 sec
          end time: 1519733748.477789793 sec
          AP of client: unknown
          Channel of AP: unknown
          Code: non-Wi-5 STA
```

Figure 9: Monitoring: output of the application using passive monitoring for obtaining detailed information of the wireless environment

4.1.2.3 Active monitoring

In some cases, it is necessary to resort to active monitoring, *i.e.,* sending some special frames just for monitoring purposes. As an example, in order to perform an optimal channel assignment, it would be convenient to have accurate information about the path-loss between Wi-5 APs. The objective is to build a matrix reflecting the *"distance in dBs"* between each pair of Wi-5 APs. Note that the *distance in dBs* is a metric introduced to estimate the location of the APs, which is one of the main input of the channel assignment algorithm designed in the context of WP4. Further details on this metric will be given in Annex 1. For that aim, beacons with a special identifier are sent from one of the APs, while the rest of the APs are listening for them. As the transmit power is known, the path-loss between the transmitter and other receiver APs can be obtained. In order to form the complete path-loss matrix, each one of the APs sends beacons which are heard by the others. This information can then be fed to an algorithm that calculates an optimal channel allocation. The algorithm is explained in detail in Deliverables D4.2 and D4.3.

In Figure 10, we show the scheme of the active monitoring application. It should be noted that this operation can be done without disrupting ongoing connections, as it is performed using an auxiliary interface.

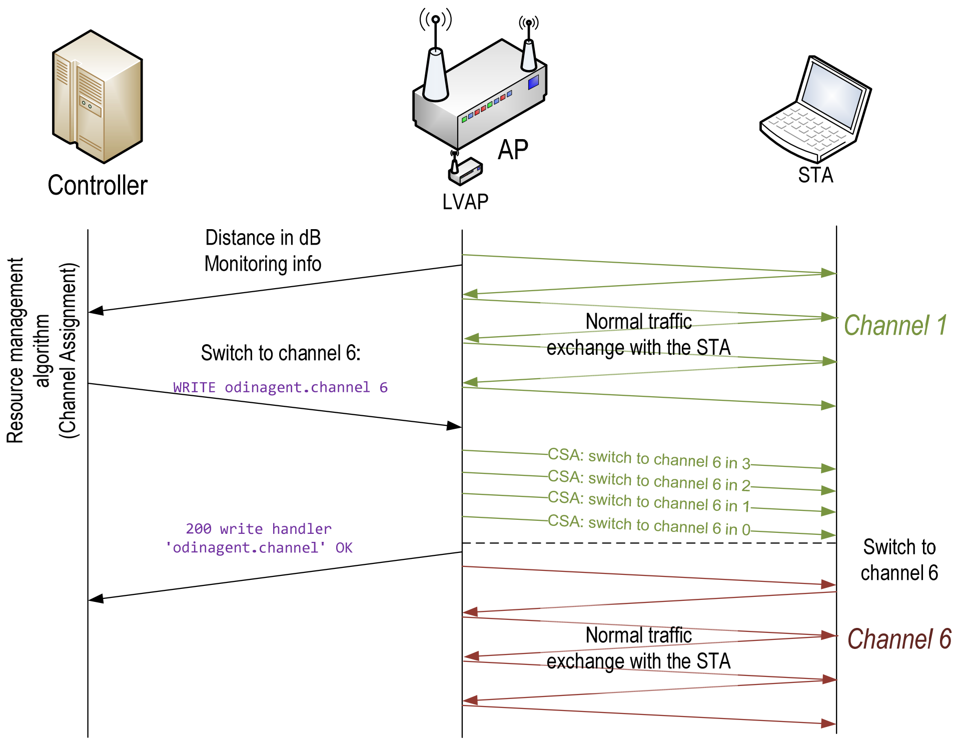

Controller
AP
LVAP
STA
Resource management algorithm (Channel Assignment)
Distance in dB
Monitoring info
Switch to channel 6:
WRITE odinagent.channel 6
Normal traffic exchange with the STA
Channel 1
CSA: switch to channel 6 in 3
CSA: switch to channel 6 in 2
CSA: switch to channel 6 in 1
CSA: switch to channel 6 in 0
200 write handler 'odinagent.channel' OK
Switch to channel 6
Normal traffic exchange with the STA
Channel 6

Figure 10: Active monitoring scheme

In Figure 11 we provide the output of the active monitoring application, *i.e.,* the "path-loss matrix" built after sending beacons between all the APs.

```
[ChannelAssignment] === MATRIX OF PATHLOSS (dB) ===
192.168.1.13  --------  67.95 dB  65.52 dB  53.97 dB
192.168.1.15  64.68 dB  --------  60 dB     52.14 dB
192.168.1.14  60 dB     50 dB     --------  40 dB
192.168.1.10  54.31 dB  60 dB     41.02 dB  --------
[ChannelAssignment] ================================
Optimization terminated.
```

Figure 11: Monitoring: output an application using active monitoring for calculating the path loss matrix between 4 APs

4.2 Channel Assignment

Using as an input the result of the application for obtaining the *"distance in dB"* presented above, an integrated application able to perform optimal channel assignments in the APs has been developed. The *ChannelAssignment* solution[7] is an example of an application where all the elements are combined. It uses different inputs:

- Active monitoring tools provide a matrix of *"distance in dBs"* between Wi-5 APs. Figure 12 shows an example obtained with 6 APs.

[7] Wi-5 *ChannelAssignment* applications, see https://github.com/Wi5/odin-wi5/wiki/Applications-for-Channel-Assignment

```
[ChannelAssignment] === MATRIX OF PATHLOSS (dB) ===
[ChannelAssignment]     1 scans
192.168.1.13 ----------   70.0 dB          66.289 dB         70.0 dB           70.0 dB           69.118 dB
192.168.1.15 66.225 dB    ----------       60.108 dB         50.0 dB           30.861 dB         40.0 dB
192.168.1.14 63.735 dB    60.111 dB        ----------        60.0 dB           60.0 dB           70.0 dB
192.168.1.1  66.289 dB    50.0 dB          60.0 dB           ----------        40.0 dB           40.0 dB
192.168.1.3  70.0 dB      30.098 dB        60.199 dB         40.0 dB           ----------        50.0 dB
192.168.1.2  60.0 dB      40.0 dB          64.317 dB         40.0 dB           50.0 dB           ----------
[ChannelAssignment] ================================
```

Figure 12: Monitoring: *distance in dBs* matrix between 6 APs

- With these values, the controller retrieves the transmission power of each AP and calculates the *Interference Impact* that the channel assignment algorithm aims to optimise. This metric, which we will call *Internal Interference Impact*, gives an estimation of the interference caused in a Wi-5 AP by other Wi-5 APs in the neighbourhood. It can be observed that the result is also a 6x6 array (see Figure 13).

```
[ChannelAssignment] ================================
[ChannelAssignment] = INTERNAL INTERFERENCE IMPACT =
[ 0,00 -61,87 -58,16 -61,87 -61,87 -60,99 ]
[ -58,10 0,00 -51,98 -41,87 -22,73 -31,87 ]
[ -55,61 -51,98 0,00 -51,87 -51,87 -61,87 ]
[ -58,16 -41,87 -51,87 0,00 -31,87 -31,87 ]
[ -61,87 -21,97 -52,07 -31,87 0,00 -41,87 ]
[ -51,87 -31,87 -56,19 -31,87 -41,87 0,00 ]
[ChannelAssignment] ================================
[ChannelAssignment] Interference Impact: -1423.4840487309427
```

Figure 13: Monitoring: *Internal Interference Impact* matrix between 6 APs

- The *External Interference Impact* is a metric that allows to include in the optimisation problem, already defined in deliverables D4.2 and D4.3, the possible effect of non-Wi-5 equipment in the environment, e.g., APs which are not managed by the Wi-5 controller. In this case (see Figure 14), the number of columns is 6, corresponding to the Wi-5 APs, and the number of rows is 11, *i.e.,* the number of Wi-Fi channels in use.

```
[ChannelAssignment] ================================
[ChannelAssignment] = EXTERNAL INTERFERENCE IMPACT =
[ -29,26 -28,78 -25,85 -25,78 -25,59 -25,40 ]
[ -29,26 -28,62 -26,04 -25,86 -25,66 -25,61 ]
[ -64,59 -39,35 -29,38 -29,34 -28,97 -28,93 ]
[ -43,24 -38,29 -38,20 -37,83 -35,90 -34,40 ]
[ -37,40 -33,93 -33,75 -33,50 -32,54 -31,68 ]
[ -35,98 -29,50 -29,40 -29,28 -28,39 -28,31 ]
[ -41,97 -36,09 -35,44 -34,91 -33,80 -33,57 ]
[ -55,33 -39,23 -39,11 -38,28 -36,10 -34,44 ]
[ -51,00 -38,99 -38,84 -33,55 -31,27 -30,22 ]
[ -29,38 -28,97 -28,84 -28,78 -28,50 -28,39 ]
[ -45,71 -36,23 -36,05 -33,84 -32,01 -31,55 ]
[ChannelAssignment] ================================
```

Figure 14: Monitoring: *External Interference Impact* matrix using 6 APs

Note that the inclusion of the *External Interference Impact* is motivated by the detection of external sources of interference during the field trial stage in which the algorithm was being tested. The operation of the Channel Assignment application is detailed in Figure 15.

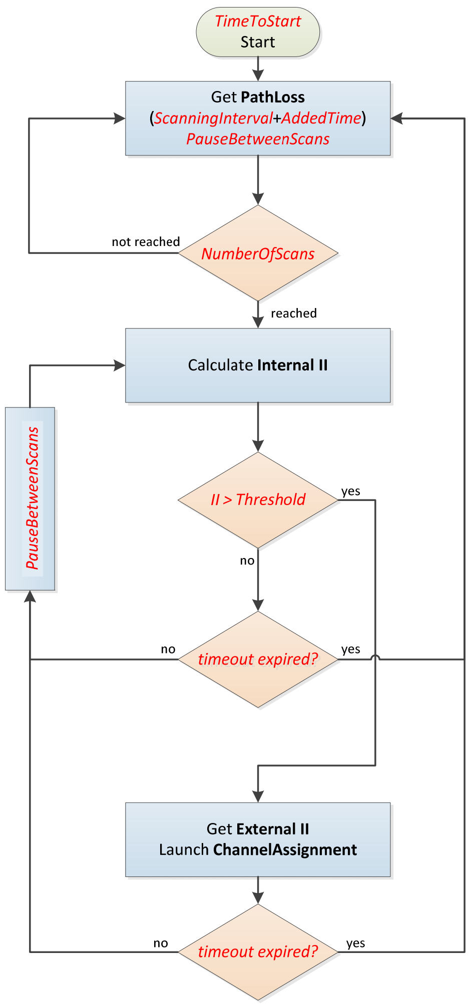


Figure 15: Operation scheme of the Channel Assignment application

After obtaining the *distance in dBs* (*i.e.,* pathloss) matrix between APs, the controller calculates the *Internal Interference Impact.* If a *Threshold* value is not reached, it checks if *Timeout* has expired. In this case, the controller orders the APs to rescan so as to get the matrix again; otherwise it waits *PauseBetweenScans* before recalculating the *Internal Interference Impact*. After the pause between scans, if the application is in *manual* mode, the user can manually modify the channel assignment. This option was included in order to test the correct behaviour of the channel assignment algorithm.

If the *Threshold* is reached, the application instructs the APs to scan with their auxiliary interfaces, to obtain the *External Interference Impact.* Once updated, it calls the channel assignment algorithms developed in WP4 and detailed in Deliverables D4.2 and D4.3. These algorithms optimise the channels using the information about *Internal* and *External Interference Impact*. Finally, it implements the decision of the algorithm and checks *Timeout* in order to rescan or wait.

The Matlab code implementing the channel assignment algorithm[8] has been exported as a Java library using the Matlab compiler. A free set of libraries called Matlab Compiler Runtime (MCR)[9] has to be installed in the machine hosting the controller, in order to run the channel assignment application. As

[8] Wi-5 *ChannelAssignment* application, see https://github.com/Wi5/Wi-5-Channel-Assignment
[9] Matlab Compiler Runtime (MCR) is defined as *"a standalone set of shared libraries that enables the execution of compiled MATLAB applications or components on computers that do not have MATLAB installed."* See https://uk.mathworks.com/products/compiler/mcr.html, accessed Sep 2017.

the code is previously compiled, it can run fast and does not require a full Matlab installation. The compiled package has been shared in the GitHub repository. Detailed information about the process required to build and run the code integrating Java and Matlab has been added in the GitHub wiki, also including detailed documentation[10].

As a result of the algorithm, the controller gets a series of channels that it has to assign to each of the APs (see Figure 16).

```
[ChannelAssignment] =======CHANNEL ASSIGNMENTS=======
1     6     1    11    11     1
[ChannelAssignment] ================================
```

Figure 16: Results of the Channel Assignment application

In order to implement this decision, it has to perform some changes in the APs. The scheme of the operation for switching the channel of an AP with its associated STAs is illustrated in Figure 17. We also show here some parts of the log information shown by the controller.

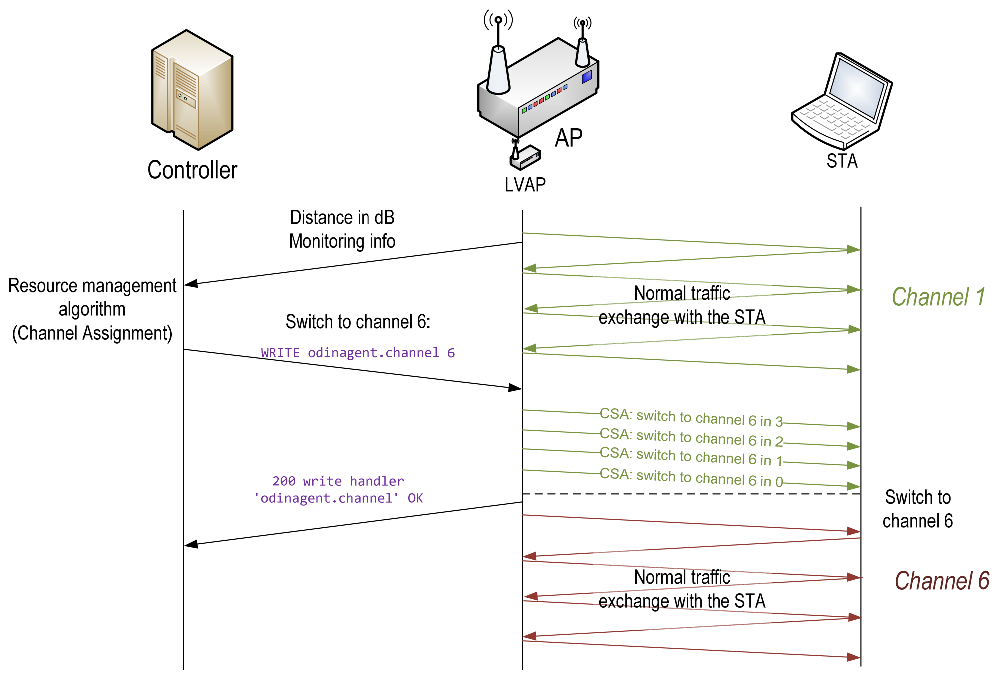

Controller
AP
LVAP
STA
Distance in dB
Monitoring info
Resource management algorithm (Channel Assignment)
Switch to channel 6:
WRITE odinagent.channel 6
Normal traffic exchange with the STA
Channel 1
CSA: switch to channel 6 in 3
CSA: switch to channel 6 in 2
CSA: switch to channel 6 in 1
CSA: switch to channel 6 in 0
200 write handler 'odinagent.channel' OK
Switch to channel 6
Normal traffic exchange with the STA
Channel 6

Figure 17: Scheme of the operation of the *ChannelAssignment* application

A message `WRITE odinagent.channel` is sent by the controller to each AP (see Figure 18).

```
[ChannelAssignment] ================================
[ChannelAssignment] Setting AP /192.168.1.13 to channel: 1
[ChannelAssignment] Setting AP /192.168.1.15 to channel: 11
[ChannelAssignment] Setting AP /192.168.1.14 to channel: 6
[ChannelAssignment] Setting AP /192.168.1.10 to channel: 11
[ChannelAssignment] ================================
```

Figure 18: Messages sent to the APs by the Channel Assignment application

[10] Wi-5 applications for channel assignment, see https://github.com/Wi5/odin-wi5/wiki/Applications-for-Channel-Assignment

When the AP receives the message, it also has to instruct all the associated STAs to switch to the new channel. Beacons including the Channel Switch Announcement element are used for this. A *countdown* is implemented so the STA switches to the new channel when it reaches zero.

When the AP has finished the channel switch (including the switch of the associated STAs), it sends a `200 write handler 'odinagent.channel' OK` message to the Wi-5 controller.

4.2.1 Parameters that can be tuned for Channel Assignment

These are the parameters that the system user can set for the application, using `poolfile`:

- `TimeToStart` (sec): The application will wait some seconds before starting. It gives time to the user to start the agents. This is run only one time.
- `PauseBetweenScans` (sec): The time between the scans that are run before the algorithm.
- `ScanningInterval` (sec): Time used for collecting the monitoring information.
- `AddedTime` (sec): An extra time after scanning (a margin). Recommended value: 1 second.
- `NumberOfScans`: The number of times the wireless environment is scanned before running the algorithm.
- `Timeout` (sec): The time the application uses for deciding when to perform the algorithm.
- `Channel`: Channel used for the measurement of the distance (sending and hearing for special beacons).
- `Method`: Three values can be chosen:
  - 1: Wi-5 algorithm. It will use the algorithm for optimal channel selection.
  - 2: Random channel selection.
  - 3: Least Congested Channel (LCC).
- `Threshold`: Internal Interference Impact should be less than this value in order to scan External Interference Impact and trigger the algorithm. Recommended value: `-50`.
- `Mode`: Two values can be chosen:
  - `auto:` the application runs without user prompt.
  - `manual:` it allows the user to change manually the channels of each AP to test if the algorithm restores the setup to optimal channel selection.
- `Filename`: Logging file where all the application info will be saved. Optional parameter.

The delays incurred by this application are studied in detail in Section 5.1.2.2.

4.2.2 Applications for test purposes

Three more applications related to channel assignment have been developed for test purposes:

- *ChannelLoop*[11]: This application switches an AP through channels 1 to 11. The selected AP is the first one appearing in `poolfile`.
- *ChannelPrompt*[12]. This application allows the user to introduce a channel number for each AP using the keyboard.

[11] Wi-5 *ChannelLoop* application, see https://github.com/Wi5/odin-wi5-controller/blob/master/src/main/java/net/floodlightcontroller/odin/ applications/ChannelLoop.java
[12] Wi-5 *ChennelPrompt* application, see https://github.com/Wi5/odin-wi5-controller/blob/master/src/main/java/net/floodlightcontroller/odin/ applications/ChannelPrompt.java

- Initial version[13]. It is only based on the "distance in dBs" between APs. It takes monitoring info as an input, runs an algorithm and implements the resulting channel assignment in the APs, moving the AP itself and its associated STAs.

4.3 Mobility Management and Load Balancing

One of the advantages provided by the Wi-5 architecture is that there is a central element (the Wi-5 controller) with a global and updated view of the network and its status. As explained in Section 4.1.2, the information monitored by the APs makes it possible for the controller to know which STA is connected to which AP, the power level with which each STA is *seen* by each AP, the traffic generated and received by each STA and its airtime consumption.

There are different problems that have to be simultaneously addressed in the considered scenarios: first, one of the requirements of the system is that the users may move at walking speed, *i.e.,* they will sometimes need to handover between APs. In addition, it should be possible to run smart algorithms for load balancing: for example, if a STA can connect to two different APs (it is in the middle), the controller should connect it to the least congested one, or to one where its quality requirements can be granted.

In this section we will present and compare the different solutions that have been developed to address these problems: Mobility Management and Load Balancing to provide a good quality to the users. It should be noted that both functionalities have to be addressed in a coordinated way, so as to avoid inconsistencies between their respective decisions (*e.g.,* a load balancer might decide to move a STA to a less loaded AP, but the mobility manager could move it back, trying to associate it with the closest AP). In subsequent sections, we will see how both functionalities can be integrated in a coordinated way.

The seamless handover is a crucial part of the proposed solution. Not only is it used when the user walks and changes its AP, but it is also necessary when a load balancing algorithm is run: it can decide to move a STA from one AP to another, requiring a seamless handover.

The comparison between these two handover schemes has been submitted to IEEE Access. On 9th April 2018 we received a first response, encouraging us to include some improvements and resubmit the article [38].

4.3.1 Reactive Mobility Management

A reactive seamless handover was proposed and measured in Deliverable D3.3, and also in [23]. This smart functionality [14] has been improved by the inclusion of new conditions for the trigger: a subscription is set in the AP, so it sends a message to the controller if a number of consecutive frames (5 by default) are under a power threshold (*e.g.,* -56 dBm), meaning that the STA is moving away from the AP. There is a time to reset the trigger (1 s by default), and a hysteresis time (15 s by default) to avoid the ping-pong effect. As an example, the condition for sending the message to the controller could be: *"5 packets below -56 dBm have been received from a STA in the last 1 s, and a previous handover of this STA has not happened in the last 15 s."*

[13] Wi-5 Channel assignment initial version application, see https://github.com/Wi5/odin-wi5-controller/blob/master/src/main/java/net/floodlightcontroller/odin/applications/ChannelAssignment.java
[14] Wi-5 *MobilityManager*, see https://github.com/Wi5/odin-wi5/wiki/Application-MobilityManager

When the controller receives the message, it sends a *Scan Request* to the APs in the neighbourhood. For a short period of time (1 s by default), all neighbouring APs switch their auxiliary interfaces to the channel of the origin AP and listen to packets originated by the STA. The APs that are able to hear packets of the STA, send a *Scan Response* message to the controller, which then selects the best suited AP and moves the LVAP to it.

The handover is performed as explained in Deliverable D3.3: it requires the use of beacons instructing the STA to move to the channel of the new AP. There is no re-association nor IP address modification, so ongoing communications are only interrupted briefly due to the channel switching.

4.3.2 Proactive Mobility Management

In the last subsection we have explained the mechanism for performing reactive handovers that are triggered by the origin AP, *i.e.,* the one to which the STA is associated. However, another option is to utilise a proactive smart handover functionality in the controller, following a *scan - gather results - make assignment decision* loop. If this loop is fast enough, it will perform a sort of fast load balancing, which can at the same time be in charge of mobility management, *i.e.,* always assigning the STA to the best suited AP.

An application called *SmartAPSelection*[15] has been developed, which is able to perform a proactive management of the mobility. A detailed explaining of its parameters is provided in Annex 2.

As shown in Figure 19, the controller asks all the APs to scan for STAs they can hear, using their auxiliary interfaces for a short period of time (200 ms by default on each channel). After that, it is able to build a matrix of the STAs that can be heard by each AP, and their respective RSSI values.

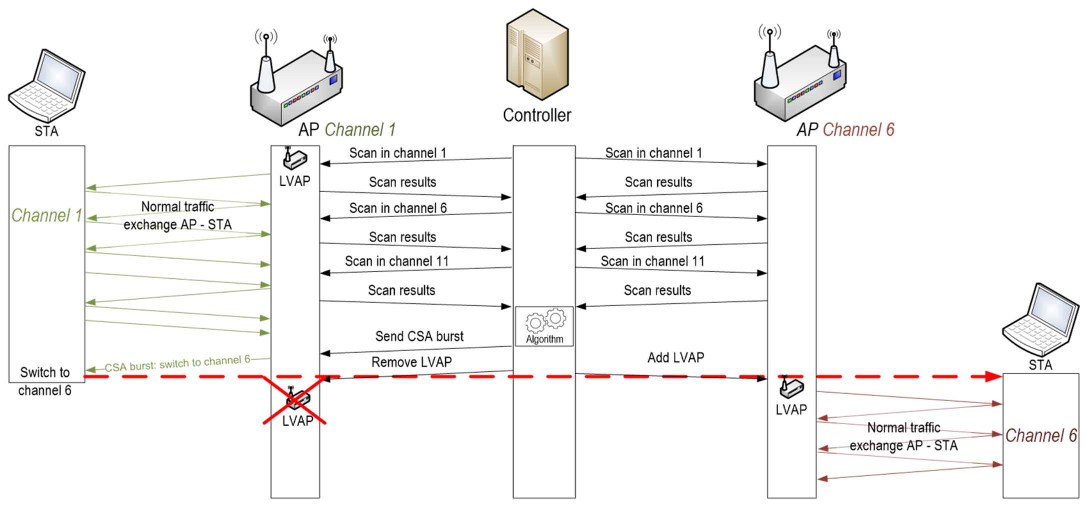


Figure 19: Scheme of the proactive mobility management

In order to keep track of some historical data to be used for these decisions, a *weighted* RSSI (*w*RSSI) is defined. Its value is updated using the average power level measured in the last iteration. $RSSI_{i,j}$ stands for the average value of the RSSI of STA_i at AP_j, measured in milliwatts.

This is the matrix with the *weighted* values of RSSI (1):

[15] Wi-5 *SmartAPSelection,* see https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection

$$\begin{pmatrix} wRSSI_{1,1} & wRSSI_{1,2} & wRSSI_{1,3} & ... & wRSSI_{1,k} \\ wRSSI_{2,1} & wRSSI_{2,2} & wRSSI_{2,3} & ... & wRSSI_{2,k} \\ wRSSI_{3,1} & wRSSI_{3,2} & wRSSI_{3,3} & ... & wRSSI_{3,k} \\ ... & ... & ... & ... & ... \\ wRSSI_{m,1} & wRSSI_{m,2} & wRSSI_{m,3} & ... & wRSSI_{m,k} \end{pmatrix} \quad (1)$$

On each iteration, the new value is calculated this way (2):

$$w\mathrm{RSSI}_{i,j}^{N+1} = \alpha \bullet \mathrm{RSSI}_{i,j}^{N} + (1 - \alpha) \bullet w\mathrm{RSSI}_{i,j}^{N} \quad (2)$$

Where $\mathrm{RSSI}_{i,j}^{N}$ stands for the average value measured in the last iteration (*i.e.,* the average power with which AP_i receives the frames from STA_j at the *N*th iteration), $w\mathrm{RSSI}_{i,j}^{N}$ represents the *N*th value of the weighted RSSI, and $0 < \alpha < 1$ is the *smoothing factor*.

The value of α modifies the behaviour of the algorithm: if it is low (*e.g.,* 0.2), the algorithm is slow and gives more importance to the history, *i.e.,* it takes time to make the handover. If it is set to a high value *e.g.,* 0.8, the STA is moved fast between APs, as the importance of the most recent observation increases. In subsequent sections we will present some tests aimed at empirically finding an optimal value for this parameter.

It should be noted that the values of *w*RSSI are stored for all STA-AP pairs (not only for the AP where the STA is associated). The initial value is -99.9 dBm (its equivalent in milliwatts) until the AP receives the first frame from a STA.

Using this information, an algorithm is executed to optimally assign each STA to the best suited AP. For example, an application may decide that a handover is only needed when all of the following conditions hold: *a)* there is an AP with a better RSSI than the current one; *b)* a hysteresis time has passed; and *c)* a power level threshold is reached.

If the new STA assignment requires an AP re-allocation, the handover mechanism is performed in a similar way to the reactive one, *i.e.,* a burst of CSA beacons is sent to the STA, and then the LVAP is moved just after it. This results in a fast and seamless handover. From that moment (red dash line in Figure 19) the STA will start working in the new channel.

4.3.3 Evaluation and comparison between reactive and proactive schemes

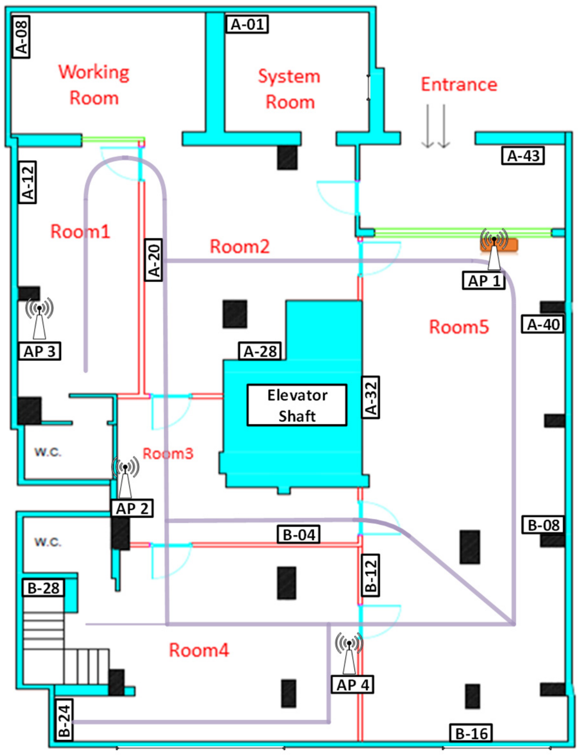
In this section we present an evaluation of the functionalities just presented. We will first describe the scenarios used for the evaluation. Then, we will present some qualitative results aimed at illustrating the advantages of the proposed architecture. Next, a series of tests comparing the reactive and proactive handover mechanisms, with their advantages and drawbacks, will be included. Finally, some quantitative measurements will be presented.

4.3.3.1 Test environments

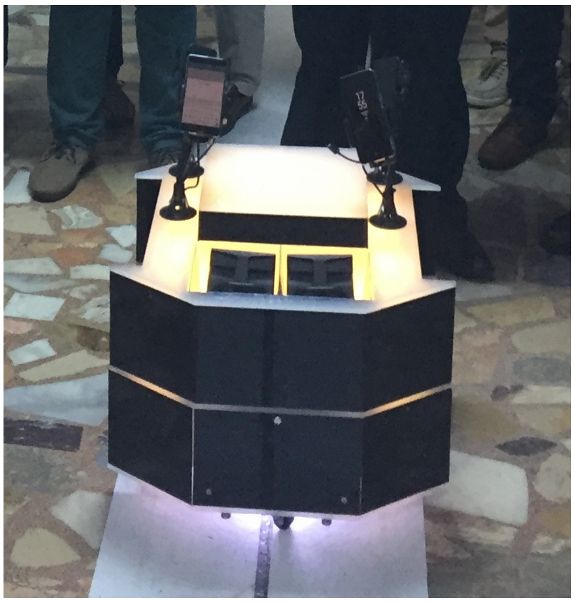
Two different scenarios have been used for the handover tests:

- AirTies's Mecidiyekoy Test House (AirTies from now), a two floor apartment in Istanbul, which provides a controlled environment that allows automated and repeatable tests by the use of robots, which move through predefined paths carrying wireless devices on them. APs can be placed in four different locations and this layout is given in Figure 20a (only the top floor has been used), a robot carrying two mobile handsets can be seen in Figure 20b.

- A corridor between different labs in a building of Unizar, see Figure 21. This is a scenario with 15 interfering APs. The tests are performed by a person walking down the corridor while carrying a wireless device. As it is a less controlled scenario (we do not control many of the APs in the environment), the repeatability of the tests is not granted, so we will only use it for obtaining qualitative results.



(a)



(b)

Figure 20: Scenarios for the tests: a) AirTies test house: b) AirTies test robot

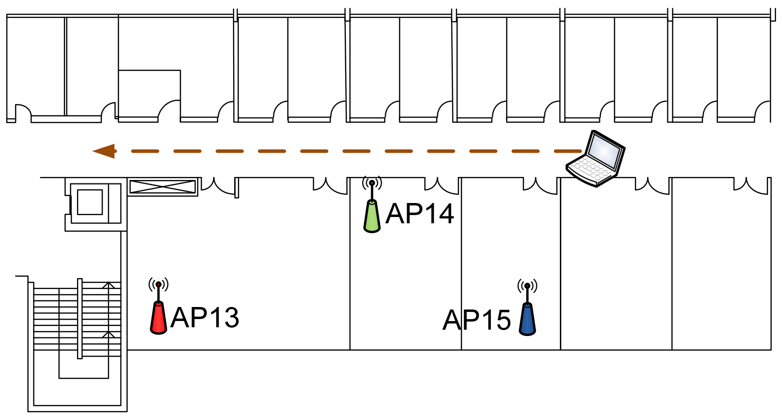


Figure 21: Scenarios for the tests: Unizar lab

We will use one scenario or the other depending on the kind of tests to be performed. Both setups include a number of off-the-shelf APs (Netgear R6100 or TP-Link 1750 Archer with OpenWrt 15.05). They include an auxiliary USB TP-LINK TL-WN722N wireless card. The APs are configured in channels 1, 6 and 11 in the 2.4 GHz band; the controller runs Debian 8.2 (Linux kernel 3.16.0.4); a DHCP server is also included.

The network scheme used in both test scenarios is shown in Figure 22. The control and data planes are separated and the traffic is sent from the moving STA to a server, located in a virtual machine in a computer that also hosts the Wi-5 controller.

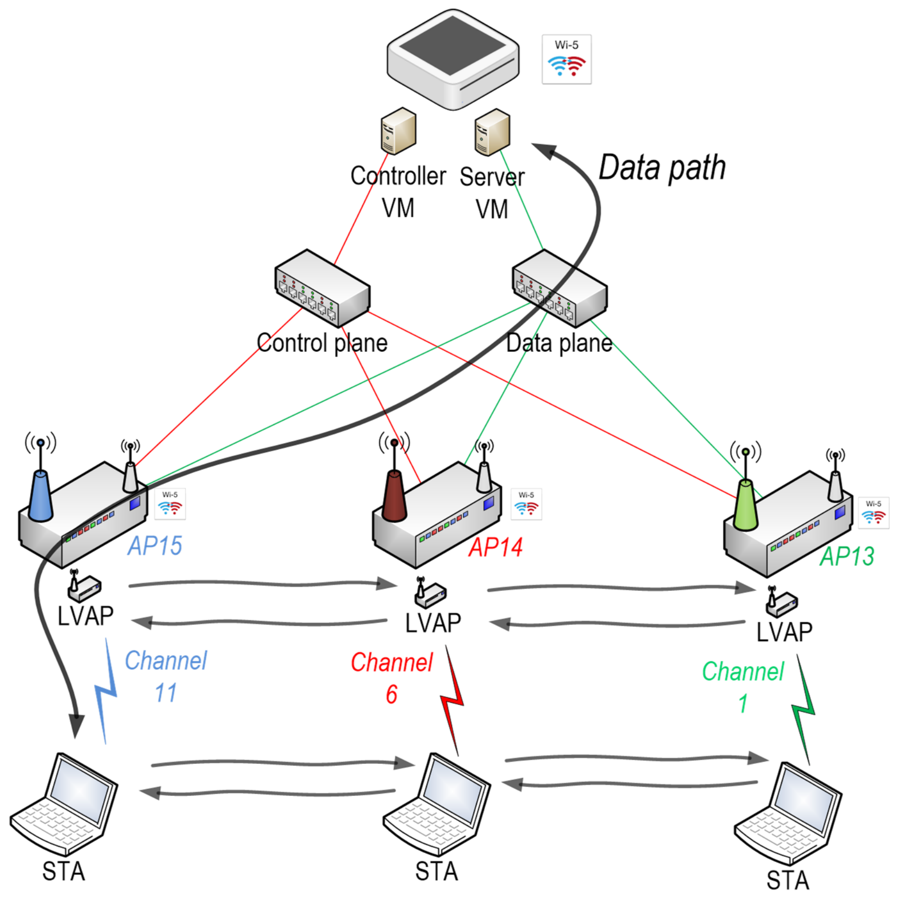


Figure 22: Network scheme used in the tests

4.3.3.2 Qualitative results

First, in order to illustrate the difference between state-of-the-art handover and our work, we have run some experiments in Unizar using a Wi-Fi network with 3 APs, with a user carrying a laptop moving within the coverage of the network. The STA runs Debian 9.1 Workstation (Linux kernel 3.16.0.4). The wireless card of the STA is a LINKSYS WUSB54GC dongle.

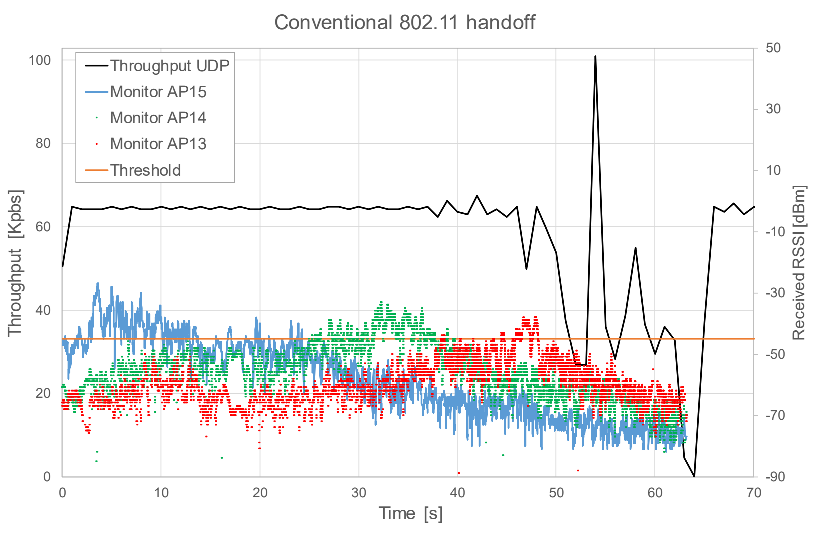
Figure 23a shows a conventional handover experiment: the laptop first connects to *AP15*, as it is the closest one when the user starts walking (see Figure 21) down the corridor (the power received in the AP is represented by the blue line *"monitor AP15"*). As the user keeps on moving, the received signal from *AP15* becomes weaker and the signal received from *AP14* becomes stronger (green line "*monitor AP14"*) as it is now closest to the client. However, the client does not initiate the handover until it loses connectivity with *AP15* (around t = 50 s), at which point the service is interrupted, as proven by the sharp drop in the measured throughput. At this moment, when the client initiates the handover procedure by scanning from closer APs, it detects that *AP13's* signal (red line *"monitor AP13"*) is stronger than that of *AP14*, and connects to it. In other words, the client has ignored *AP14* although it was the closest AP during a significant time. This is an example of the "sticky client" effect, *i.e.,* the STA prefers to stay in a known AP than to explore new options. It is also obvious that the conventional handover does not always guarantee seamless transition when a wireless client is mobile.

In contrast, when the developed Wi-5 functionalities are activated (Figure 23b), a different behaviour is observed. In this case, a reactive application runs in the controller: around t = 20 s, the signal from *AP15* drops below the predefined threshold (amber line), and this is reported to the controller. The controller then requests the two other APs to scan for the STA for 4 s. According to the results reported by the APs, the signal from *AP14* (green line) is the strongest one. Therefore, *AP14* is selected and the STAs is handed off to it. The same thing happens around t = 40 s. In both cases, the throughput at the STA (black line) is barely affected, *i.e.,* the handover of the client's connection from one AP to the other was performed seamlessly and without interruption of service.

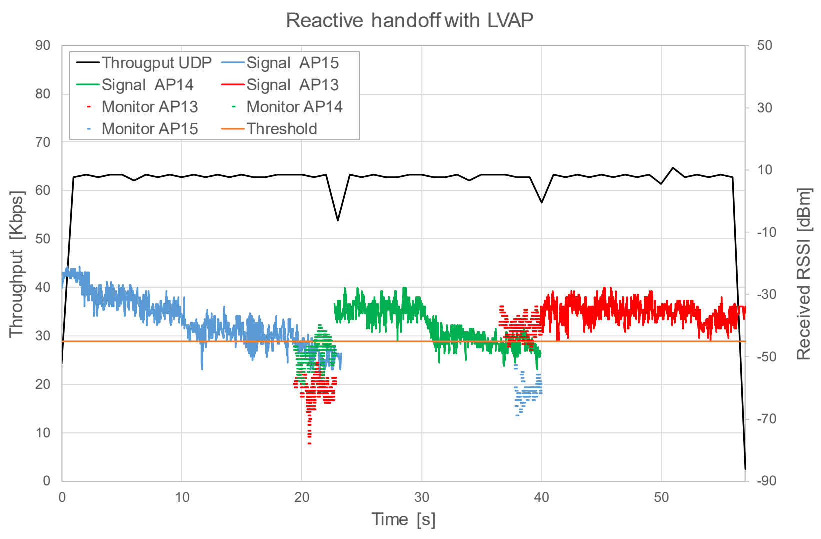
Finally, in Figure 23c we present the results of a similar experiment, in which the proactive handover application has been used. As explained before, this application performs a continuous scanning of the channels, and periodically builds a matrix of STAs and APs. The figure shows that, during the first 25 seconds, *AP15* is best suited to serve the STA: the samples obtained by the two other APs are always lower. However, at t = 25 s, the monitoring interface of *AP14* starts reporting better RSSI values. After some time, the controller selects this AP and moves the STA to it. A similar process happens at t = 35 s. It can be observed that the controller performs a complete scanning every 2-3 seconds, which is enough for giving the best service to a walking user.

It should be remarked that these three experiments have been run using the exact same hardware. The only difference is that in the first case the Wi-5 software was not running, and in the two others we activated it, with the corresponding functionalities, showing a better performance which allows the STA to use the best suited AP for it at each moment.

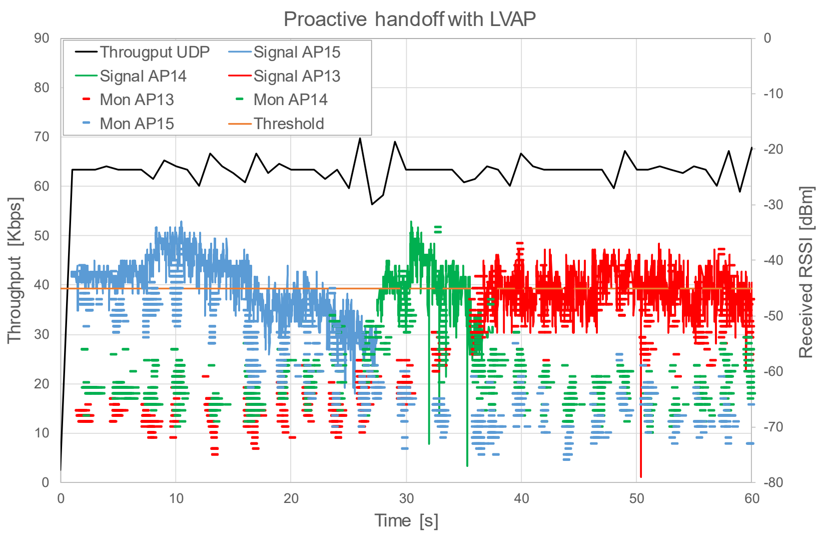
As a summary, it can be said that the "state-of-the-art" handover results in a poor performance: the client remains connected to the first AP until the signal becomes too weak. Therefore, the user will experience a bad connectivity, even if he/she is near an AP with better conditions. If the reactive application is used, the connectivity gets significantly improved, as the controller can decide which the best AP is for the client on each moment. Something similar happens if the proactive application is run. However, the proactive solution presents some interesting advantages, in terms of scalability and better integration with other functionalities, as we will see in the next subsection.



(a)



(b)



(c)

Figure 23: Comparison between Wi-Fi handovers: a) state-of-the-art; b) reactive handover; c) proactive handover

4.3.3.2.1 Limitations of the reactive mobility management

In this subsection we present the results that have been obtained after performing some scalability evaluations with the reactive handover in the AirTies test scenario (Figure 20a).

Two APs have been used, and a number of STAs are initially very close to and connected to *AP1*, sending TCP traffic (generated using *iperf*[16]) to a computer connected to the data network. Three STAs (*STA2* and *STA3* used in the experiments are mobile handsets, whereas *STA1* is a notebook computer) are carried by a robot and move slowly towards the second AP simultaneously.

The handover delays for individual STAs as well as the overall average (of all involved STAs) for each case are given in Table 1. When there is a single STA, the handover delays are reasonable with an average value of 21.9 ms. On the other hand, a significant increase in the handover delay is observed when the number of stations increases. The average handover delay with 2 STAs is around 1.5 seconds, whereas it is around 1.8 seconds with 3 STAs. The reason for this is that a number of scanning requests arrive to the AP almost simultaneously, as the AP reports that it is "loosing" the signal of the STAs. The AP has to scan for each MAC address for a period of 1 second, so the rest of the scanning requests are put in a queue which results in the handover delay increases with the number of STAs.

Table 1. Handover delays for cases with different numbers of STAs [ms]

	1 STA	**2 STAs**		**3 STAs**		
		STA1	**STA2**	**STA1**	**STA2**	**STA3**
Average	21.9	17.4	1645.8	112.9	721.8	3137.1
Std. dev.	16.1	27.5	3097.4	259.2	1271.5	4010.9
Overall avg	21.9	1562.5		1847.2		

This limitation does not appear in the proactive handover scheme, as there is a continuous loop running, regardless of the movement of the STAs. Therefore, we can conclude that the proactive handover mechanism presents some advantages:

- New parameters, in addition to RSSI, can be considered in order to make smarter algorithms. Some examples could include load balancing based on the services being run an AP.
- The mobility management functionality can be integrated with load balancing. This avoids potential problems caused by contradictory decisions between load balancing and mobility management.
- Improved scalability, as the scanning time is shared for all STAs, whereas in the reactive handover the application has to order the scanning for each terminal spending longer time. This can happen if a user is carrying two devices, *e.g.,* using a tablet and carrying a mobile handset in his/her pocket.
- The parameters that are used for making the decision are more "correlated" in time, *i.e.,* they correspond to the same interval for all the STAs.
- Some historical information can also be used, giving specific weights to new and to old data. Therefore, the decision is not just based in the information obtained in the few last seconds.

4.3.3.2.2 Optimisation of the proactive mobility management parameters

As explained in 4.3.2, the proactive mobility management has two main parameters, namely the hysteresis time, which avoids frequent handovers, and the value of α, which gives a higher or lower weight to the historical data with respect to the last measurement.

[16] iPerf, https://iperf.fr/

In this subsection we present some empirical tests aimed at obtaining the optimal values for these two parameters in a realistic scenario. Therefore, they have been performed at the AirTies test house. A tour is as follows (see Figure 20a): a user starts in front of *AP1* (room 5), goes to *AP4* following the path (room 4), then to *AP2* (room 3), to *AP3* (room 1) and finally back to *AP1* (room 2 and room 5). Therefore, the total number of handovers should ideally be 4 per tour.

Tests at different speeds have been performed: *a)* a *robot* at low speed; *b)* a person at walking speed (*normal walk*); and *c)* a person walking fast (*fast walk*). The results for different values of α with a fixed hysteresis time value of 4 seconds are presented in Table 2, including the average number of handovers per tour (an average of 4 tours), and also the average number of decisions (not all decisions result in a handover), which depends on the duration of each test. For the given layout and movement pattern, we expect to have 4 handovers each tour, as the STA moves in front of each AP once during a tour.

Table 2. Effect of α parameter: Average number of handovers per tour

	Handovers α =1	Handovers α =0.8	Handovers α =0.6	Number of decisions
Robot	7.75	4.5	4	564
Normal walk	5.75	3.75	2.5	178
Fast walk	5.75	3	2	105

We observe that a value of α=0.8 provides good results for the three speeds, as the average number of handovers is nearly 4 in every case.

Another test battery has been run in order to find the optimal value of the hysteresis time. In Table 3, we show the number of handovers obtained for α=0.8, when using different values for the hysteresis time. It can be observed that a time of 4 seconds is a good option for this parameter as the average number of handovers is again close to 4 for the different mobility speeds for that case.

Table 3. Effect of hysteresis time: Average number of handovers per tour

	Handovers time = 2s	Handovers time =4s	Handovers time =10s	Number of decisions
Robot	7.5	4.5	4	564
Normal walk	7	3.75	3.75	184
Fast walk	7	3	2.75	100

4.3.3.3 Quantitative results

This section presents some quantitative measurements of the handover delay. This parameter is of primary importance in the proposed solution as Wi-5 is intended to support real-time services in which disruption can be critical: for example, in a Skype call or in an online game party, being disconnected for 1 or 2 seconds may jeopardize the user's experience.

4.3.3.3.1 Test methodology

The methodology considers a setup including two APs and a single STA. Two parameters are varied: inter-beacon time and the wireless card. An application is set in the controller, which hands off the STA between the two APs every 3 seconds. The test environment here is the Unizar lab, *i.e.,* a noisy one.

As illustrated in Figure 24, the setup includes a traffic generator using D-ITG [21], connected to the STA with an Ethernet cable. The STA receives this traffic by the Ethernet card, and forwards it to the AP via the 802.11 interface. This allows us to include a sniffer that can capture the traffic in the link between the traffic generator and the STA, in addition to that in the control and data planes. Through this setup, we can get a synchronised and combined packet capture, which enables fine grained delay calculations.

A flow of small UDP packets (80 bytes of payload) with an inter-packet time of 10 ms is sent from the generator to the server. After obtaining a number of handovers, a Perl script obtains the delays and marks the lost packets. Finally, a Python script calculates the delay as the time between the arrival of the packets before and after the handover.

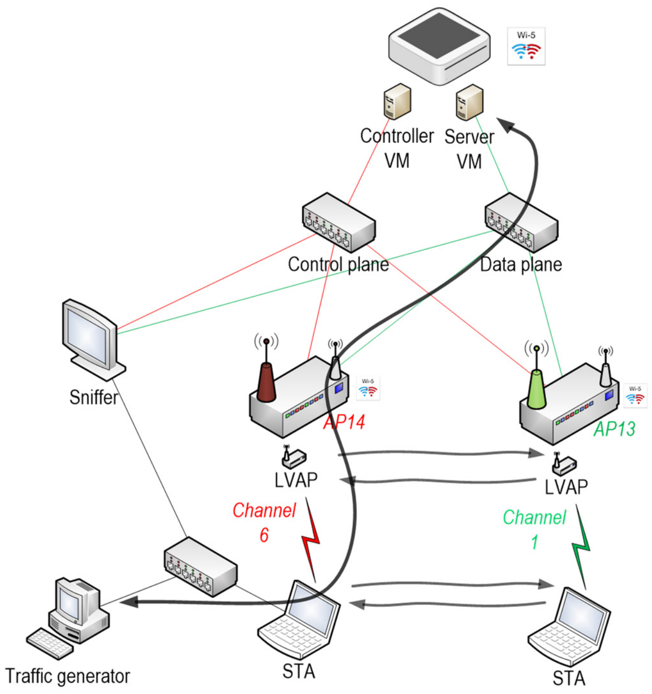


Figure 24: Test setup for measuring the handover delay

4.3.3.3.2 Modification of inter-beacon time

First, the effect of modifying the inter-beacon time is studied. This parameter is not fixed by the 802.11 specification, so it can be tuned. If the inter-beacon time is too low, it will result in a low efficiency, as a big amount of air time will just be occupied by the beacons. In our case, the use of LVAPs makes it even more difficult, as broadcast beacons cannot be used (each STA receives beacons from a different virtual MAC address). However, if the inter-beacon time is too long, some STAs may experience problems, and the handoffs will take more time, as reported in [23].

In Figure 25, we represent a *box and whisker plot* obtained after 1,779 handovers (355 on average per inter-beacon time value) using an Intel PRO/Wireless 4965 wireless card in the client. Although some values are higher, the average is 75 ms, and the vast majority of the handovers take less than 150 ms. Some outlier results can also be observed: as the measurements take place in a noisy environment with more than 15 non-Wi-5 APs, in some cases the amount of lost packets can be higher.

Curiously, the lowest value of inter-beacon time (10 ms) does not draw the best results. The reason for this can be that the device is receiving too many beacons, so it becomes too busy. A value of 60 ms can be recommended, but also 100 ms (the default value in many commercial APs) can be sufficient in order to provide fast handovers.

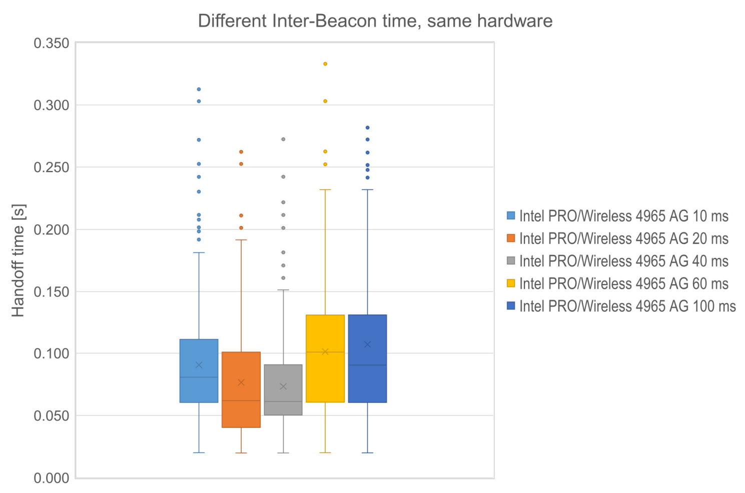


Figure 25: Box plot of the handover delay, using the same hardware (Intel PRO/Wireless 4965) and different values of inter-beacon time

4.3.3.3.3 Effect of different wireless cards

In this subsection, the behaviour of three different wireless interfaces is compared. Figure 26 shows the box plot obtained with three different low cost wireless devices ($10 to 15), namely Intel PRO/Wireless 4965, Linksys WUSB54GC and WiPi COMFAST88. A total of 893 handovers have been performed (average 297 per wireless interface). It can be observed that the three devices have good behaviour, *i.e.,* the average handover time is always below 200 ms and the median is between 50 and 75 ms.

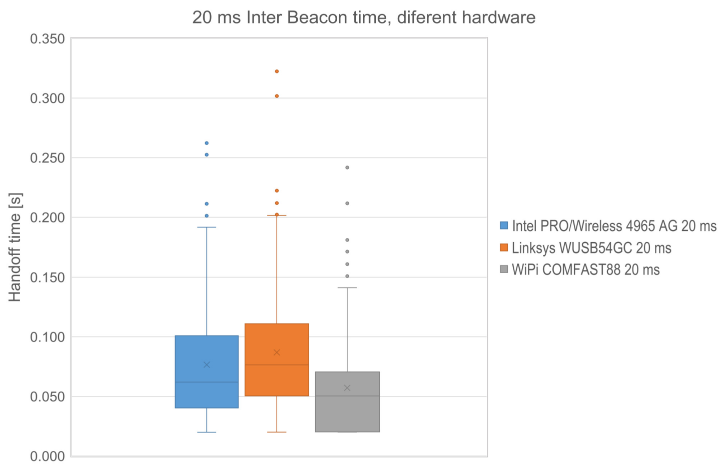


Figure 26: Box plot of the handover delay, using a constant inter-beacon time of 20 ms and different hardware

Finally, Figure 27 shows the results obtained with the same hardware, but using an inter-beacon time of 100 ms. It can be observed that in this case, the WiFi behaves poorly, *i.e.,* although the median and the average are low, some of the handovers may require up to 500 ms.

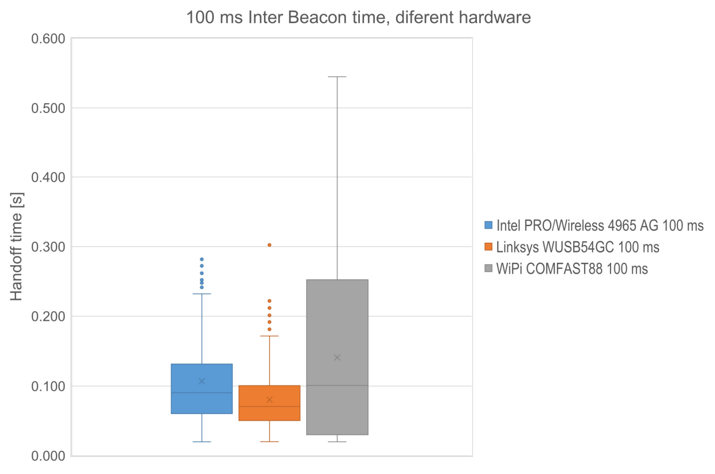


Figure 27: Box plot of the handover delay, using a constant inter-beacon time of 100 ms and different hardware

4.3.4 Integration of load balancing with mobility management

The integration between the different Wi-5 applications is crucial in order to have a coherent method of resource management. Problems may appear if mobility management and load balancing are treated in parallel, but in an uncoordinated way: for example, in a scenario with two STAs associated to the same AP, a load balancing algorithm may decide to move one of these STAs to another AP. But perhaps the mobility management algorithm can subsequently decide to put the STA back in its original AP, because it is the one heard with the highest energy.

As reported in Deliverable D3.3, a load balancing application[17] could be used to evenly distribute the number of STAs between the available APs. However, this application does not have any mechanism to coordinate with the reactive mobility management one[18]. To solve this problem, load balancing has now been integrated within the proactive mobility management application[19], so both operations can be performed in a coordinated way. For that aim, a new mode called `BALANCER`[20] can be selected by the network manager in the `poolfile`. The aim of the application is to ensure good link quality for all the STAs, *i.e.,* to have an RSSI value above a threshold (`SignalThreshold`) while at the same time, the number of STAs associated to each AP will be maintained close to the average.

The application follows the loop shown in Figure 28: at the beginning of each loop, it obtains the RSSI matrix (already explained in section 4.3.2), including the RSSI value of each STA on each AP. Then, it runs a load balancing algorithm which returns a STA that can be moved from one AP to another. A single STA is handed off per cycle, which is enough for users at walking speed, taking into account that one cycle can be completed in less than one second, or in 2-3 seconds, depending on the machine where the controller is run (see section 5.1.3). Finally, it performs the handover, following the scheme presented in section 4.3.2.

[17] Wi-5 *SimpleLoadBalancer* application, https://github.com/Wi5/odin-wi5-controller/blob/master/src/main/java/net/floodlightcontroller/odin/applications/SimpleLoadBalancer.java
[18] Wi-5 reactive *MobilityManager* application, https://github.com/Wi5/odin-wi5/wiki/Application-MobilityManager
[19] Wi-5 *SmartAPSelection*, an application integrating proactive mobility management and load balancing, https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection
[20] Wi-5 *SmartAPSelection*, BALANCER mode, https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection#balancer-mode

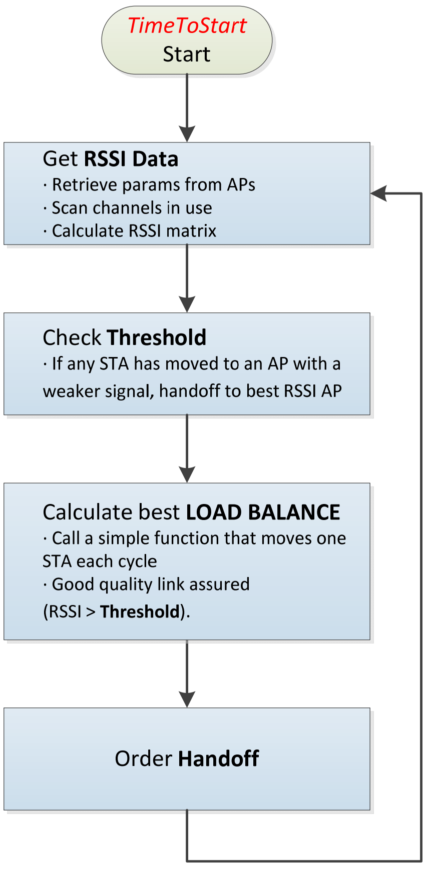

TimeToStart
Start
Get RSSI Data
· Retrieve params from APs
· Scan channels in use
· Calculate RSSI matrix
Check Threshold
· If any STA has moved to an AP with a weaker signal, handoff to best RSSI AP
Calculate best LOAD BALANCE
· Call a simple function that moves one STA each cycle
· Good quality link assured (RSSI > Threshold).
Order Handoff

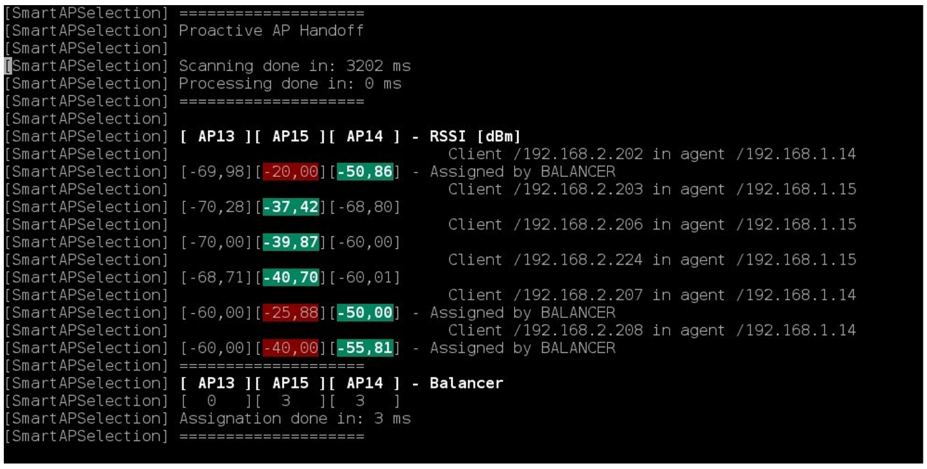
Figure 28: Operation scheme of the SmartAPSelection application when performing RSSI-based load balancing

The pseudo code implemented in the *"calculate best load balance"* algorithm is shown below:

```
Find all the APs able to hear any of the STAs with ( RSSI > SignalThreshold )

Between these APs:
      identify
              APmin: AP with the minimum number of associated STAs (=minSTAs)
              APmax: AP with the maximum number of associated STAs (=maxSTAs)

      calculate
              avgSTAs: Average number of associated STAs per AP

if ( minSTAs < avgSTAs ) AND ( maxSTAs > avgSTAs )

      STAtomove: STA that has the maximum RSSI in APmin

      handoff STAtomove to APmin
```

This behaviour is illustrated in Figure 29, obtained in a scenario with 3 APs (`AP13`, `AP15` and `AP14`) and 6 STAs. The value of the RSSI threshold has been set to -60dBm. Green values correspond to the AP where the STA is (*e.g.,* client `192.168.2.202` is in `AP14`, and client `192.168.2.203` is in `AP15`). Red values correspond to the AP where the STA would obtain the best RSSI level, but a load balancing decision has moved this STA to other AP (*e.g.,* client `192.168.2.202` would obtain -20dBm in `AP15`, although it is in fact associated to `AP14`, where the RSSI is -50.86dBm).

If the algorithm associated each STA to the AP where it obtains the best RSSI, all the six STAs would be together in `AP15`. However, it can be observed that, as a result of the load balancing decisions, three of them have been moved to `AP14`. At the same time, it can be observed that `AP13` is not used by any STA: the reason is that none of them would achieve an RSSI level higher than the threshold (-60dBm).



```
[SmartAPSelection] ====================
[SmartAPSelection] Proactive AP Handoff
[SmartAPSelection]
[SmartAPSelection] Scanning done in: 3202 ms
[SmartAPSelection] Processing done in: 0 ms
[SmartAPSelection] ====================
[SmartAPSelection]
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ] - RSSI [dBm]
[SmartAPSelection]                              Client /192.168.2.202 in agent /192.168.1.14
[SmartAPSelection] [-69,98][-20,00][-50,86] - Assigned by BALANCER
[SmartAPSelection]                              Client /192.168.2.203 in agent /192.168.1.15
[SmartAPSelection] [-70,28][-37,42][-68,80]
[SmartAPSelection]                              Client /192.168.2.206 in agent /192.168.1.15
[SmartAPSelection] [-70,00][-39,87][-60,00]
[SmartAPSelection]                              Client /192.168.2.224 in agent /192.168.1.15
[SmartAPSelection] [-68,71][-40,70][-60,01]
[SmartAPSelection]                              Client /192.168.2.207 in agent /192.168.1.14
[SmartAPSelection] [-60,00][-25,88][-50,00] - Assigned by BALANCER
[SmartAPSelection]                              Client /192.168.2.208 in agent /192.168.1.14
[SmartAPSelection] [-60,00][-40,00][-55,81] - Assigned by BALANCER
[SmartAPSelection] ====================
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ] - Balancer
[SmartAPSelection] [  0   ][  3   ][  3   ]
[SmartAPSelection] Assignation done in: 3 ms
[SmartAPSelection] ====================
```

Figure 29: Screenshot of the SmartAPSelection application in `BALANCER` mode

All in all, it can be observed that the load balancer can work in a coordinated way with the mobility manager: the values in the RSSI matrix are updated every cycle (this may take less than 1 second), so the decisions will adapt the associations of the STAs depending on this.

4.3.5 Use of the Fittingness Factor parameter in the Smart AP Selection

The Fittingness Factor (FF) is a metric that associates an AP to each new user/flow, taking into consideration its bit rate requirements [39]. In this case, the controller moves the STAs to the AP where the FF is maximised. The FF coefficient is calculated using the theoretical available throughput and the required one, more details about this metric are given in Deliverables D4.2 and D4.3.

In order to implement Smart AP Selection based on the FF, a mode called Fittingness Factor (`FF`) has been implemented in the *SmartAPSelection* application[21]. For this application, the Wi-5 controller calls a function that calculates the FFs after RSSI scans each cycle (note that each new STA connects initially to the first heard AP).

The following parameters have been included in `poolfile`, as required by this mode:

- `TxpowerSTA` (dBm): This is an estimation of the transmission power of the STAs. The FittingnessFactor algorithm uses it to calculate the available throughput. It may range between 10 and 15 dBm.
- `ThReqSTA` (kbps): This represents the bandwidth required by the STAs.

The algorithm follows the scheme shown in Figure 30. From the figure we can observe that the application first gets the input parameters, note we assume that the value of `ThReqSTA` is available in the controller. Afterwards, it calculates all the FFs for all the associated STAs following the process detailed in D4.3. Finally, it selects the AP with the maximum FF for each STA.

[21] Wi-5 *SmartAPSelection*, Fittingness Factor (`FF`) mode, https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection#ff-mode

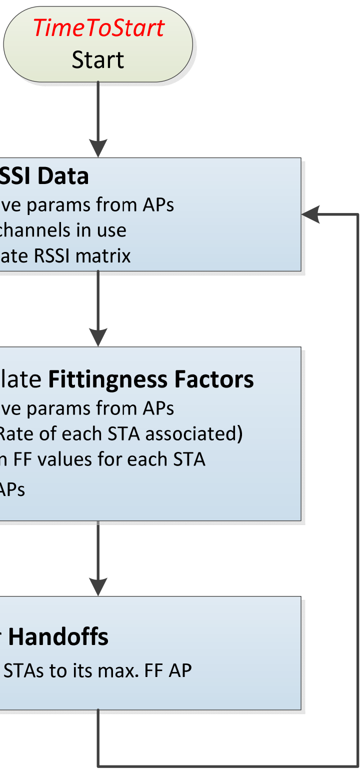
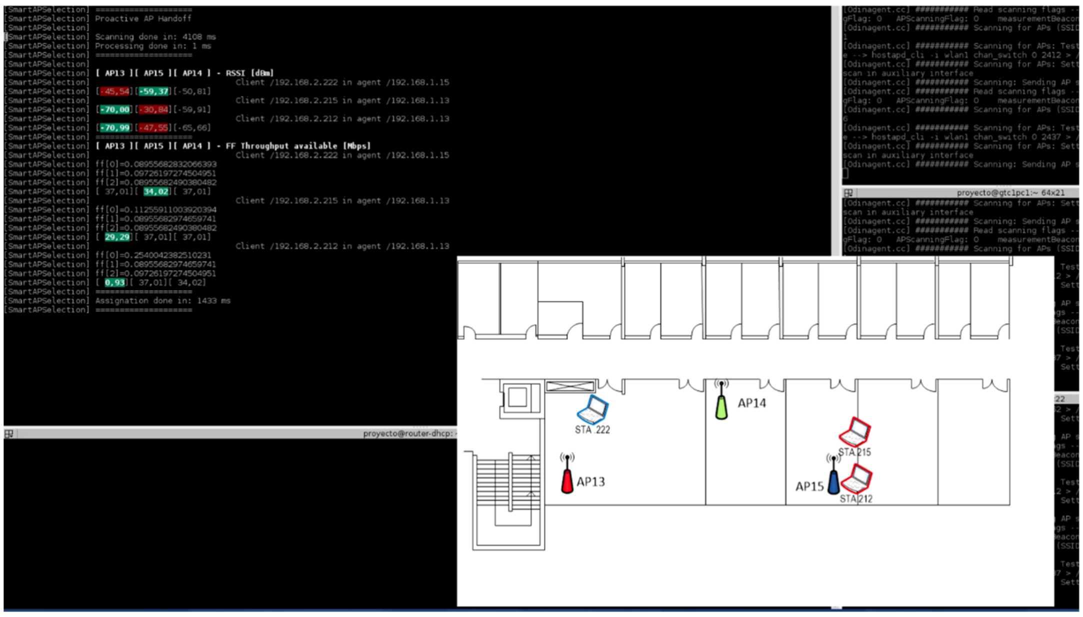

TimeToStart
Start
Get RSSI Data
· Retrieve params from APs
· Scan channels in use
· Calculate RSSI matrix
Calculate Fittingness Factors
· Retrieve params from APs
(avg. Rate of each STA associated)
· Return FF values for each STA
in all APs
Order Handoffs
· Move STAs to its max. FF AP

Figure 30: Operation scheme of the SmartAPSelection application when including the FF

In Figure 31 a screenshot of the behaviour of the application in `FF` mode is shown. It can be observed that each STA is associated to the AP where the FF is maximum (green value in the figure), which is different from the maximum RSSI (red value in the figure) used in the previous subsection. Further details on the integration of the *SmartAPSelecion* based on the FF can be found in deliverable D5.2.



```
[SmartAPSelection] ====================
[SmartAPSelection] Proactive AP Handoff
[SmartAPSelection]
[SmartAPSelection] Scanning done in: 4108 ms
[SmartAPSelection] Processing done in: 1 ms
[SmartAPSelection] ====================
[SmartAPSelection]
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ] - RSSI [dBm]
[SmartAPSelection]                                Client /192.168.2.222 in agent /192.168.1.15
[SmartAPSelection] [-45,54][-59,37][-50,81]
[SmartAPSelection]                                Client /192.168.2.215 in agent /192.168.1.13
[SmartAPSelection] [-70,00][-30,84][-59,91]
[SmartAPSelection]                                Client /192.168.2.212 in agent /192.168.1.13
[SmartAPSelection] [-70,99][-47,55][-65,66]
[SmartAPSelection] ====================
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ] - FF Throughput available [Mbps]
[SmartAPSelection]                                Client /192.168.2.222 in agent /192.168.1.15
[SmartAPSelection] ff[0]=0.08955682832066393
[SmartAPSelection] ff[1]=0.09726197274504951
[SmartAPSelection] ff[2]=0.08955682490380482
[SmartAPSelection] [ 37,01][ 34,02][ 37,01]
[SmartAPSelection]                                Client /192.168.2.215 in agent /192.168.1.13
[SmartAPSelection] ff[0]=0.11255911003920394
[SmartAPSelection] ff[1]=0.08955682974659741
[SmartAPSelection] ff[2]=0.08955682490380482
[SmartAPSelection] [ 29,29][ 37,01][ 37,01]
[SmartAPSelection]                                Client /192.168.2.212 in agent /192.168.1.13
[SmartAPSelection] ff[0]=0.2540042382510231
[SmartAPSelection] ff[1]=0.08955682974659741
[SmartAPSelection] ff[2]=0.09726197274504951
[SmartAPSelection] [ 0,93][ 37,01][ 34,02]
[SmartAPSelection] ====================
[SmartAPSelection] Assignation done in: 1433 ms
[SmartAPSelection] ====================
```

Figure 31: Screenshot of the SmartAPSelection application in `FF` mode

4.3.6 Integration of service detection

Another possibility is to assign STAs to APs as a function of the services they are running. For example, it can be interesting to place a STA running a service with real-time requirements in a concrete AP, or perhaps a sort of "VIP AP" can be available only for some STAs, *i.e.,* those with a predefined MAC address, or those running certain services.

This option has been implemented as a proof-of-concept in order to show its suitability. For that aim, the "Detection Agent," presented in Section 3.6. of Deliverable D3.1, is used. It was developed in order to make it possible for the Wi-5 controller to be aware of the different services running on the STAs associated to the APs. We reproduce here a figure from D3.1. (Figure 32: Scheme of the detection and its integration within the Wi-5 framework), in which the scheme can be observed: the router is able to detect the presence of specific flows (*e.g.,* flows with real-time requirements); the flows are duplicated by the router and forwarded to the agent; and the agent generates *flow reports* that are sent to the Wi-5 controller.

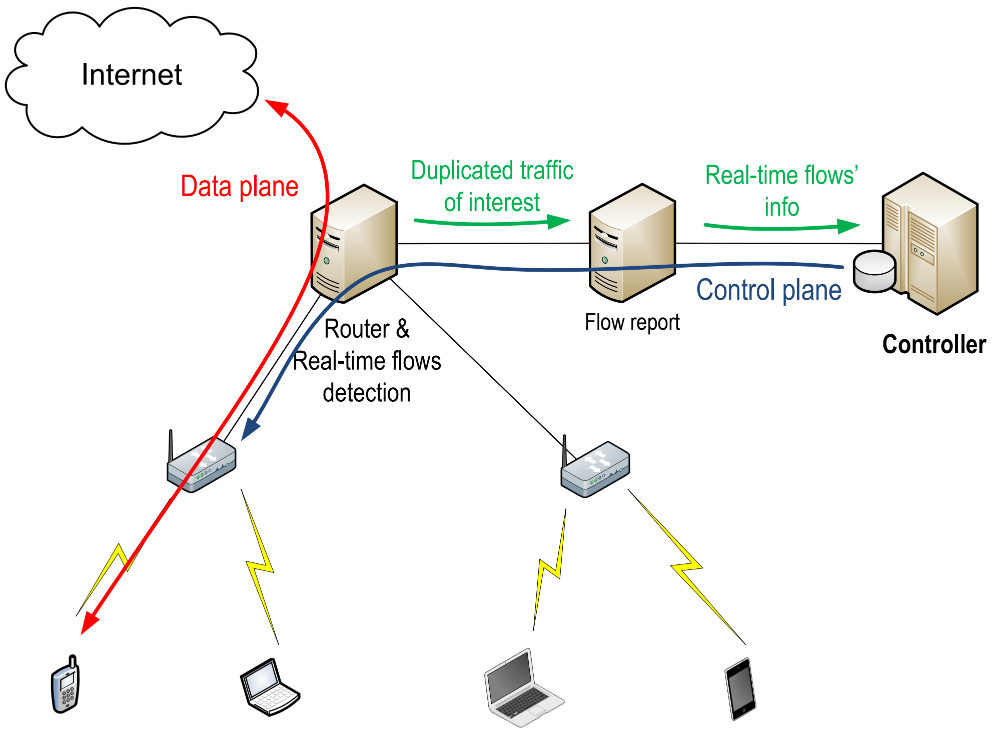


Figure 32: Scheme of the detection and its integration within the Wi-5 framework

In this document, we detail the behaviour of a Wi-5 application able to use this information about the flows in order to make handover decisions aimed at associating certain STAs (those generating special flows) into a special "VIP AP." This has been implemented as a new mode (`DETECTOR`)[22] of the *SmartAPSelecion* application. When the controller receives information about a STA starting a special service, it orders a handover of the STA to the "VIP AP." Once the flow is no longer active, the STA returns to the previous assigned AP (see Figure 33).

We next present an example, in which all STAs running SSH (port `22`) to the IP address `192.168.2.131` are moved to the "VIP AP" (`192.168.1.2` in this case). Initially, as shown in Figure 34, the STAs cannot

[22] Wi-5 *SmartAPSelection*, `DETECTOR` mode, https://github.com/Wi5/odin-wi5/wiki/Application-SmartApSelection#detector-mode

use the VIP AP (brown), as none of them is running this service. They are associated (green) to one of the other APs, according to their RSSI levels. Note that the window `Detector Agent` shows the output of this entity: in this case, no special flows have been detected: `Periodic report: Number of flows: 0`.

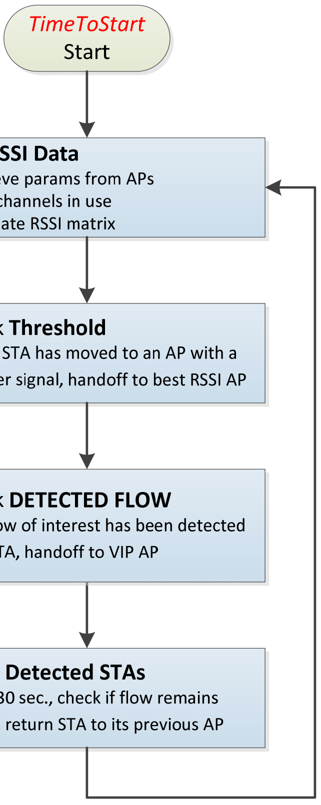
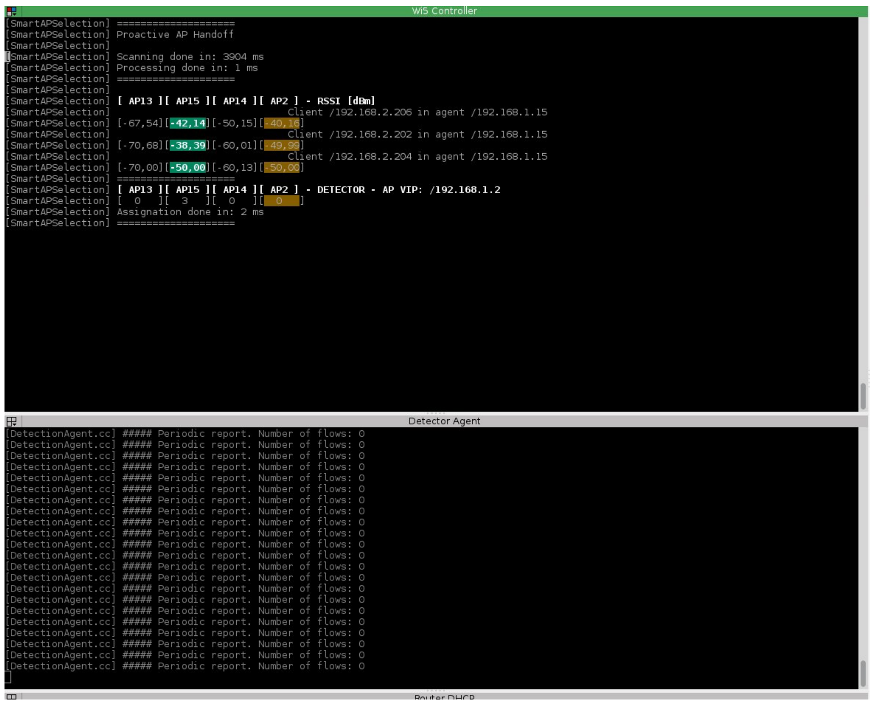

TimeToStart
Start
Get RSSI Data
· Retrieve params from APs
· Scan channels in use
· Calculate RSSI matrix
Check Threshold
· If any STA has moved to an AP with a weaker signal, handoff to best RSSI AP
Check DETECTED FLOW
· If a flow of interest has been detected in a STA, handoff to VIP AP
Clean Detected STAs
· Each 30 sec., check if flow remains
· If not, return STA to its previous AP

Figure 33: Operation scheme of the SmartAPSelection application when performing Detector mode

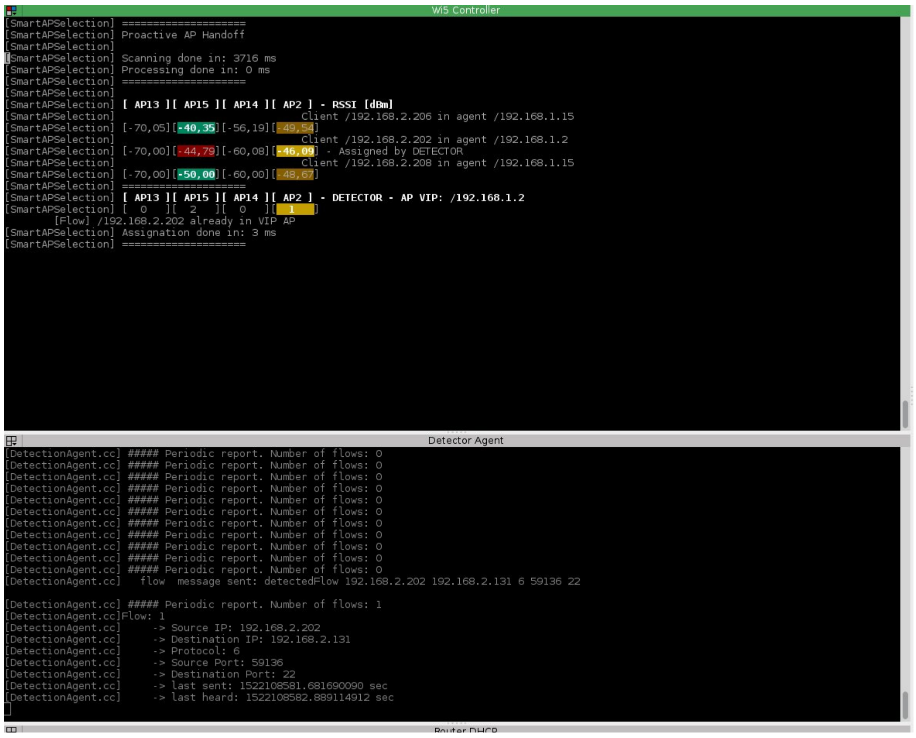


```
[SmartAPSelection] ====================
[SmartAPSelection] Proactive AP Handoff
[SmartAPSelection]
[SmartAPSelection] Scanning done in: 3904 ms
[SmartAPSelection] Processing done in: 1 ms
[SmartAPSelection] ====================
[SmartAPSelection]
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - RSSI [dBm]
[SmartAPSelection]                              Client /192.168.2.206 in agent /192.168.1.15
[SmartAPSelection] [-67,54][-42,14][-50,15][-40,16]
[SmartAPSelection]                              Client /192.168.2.202 in agent /192.168.1.15
[SmartAPSelection] [-70,68][-38,39][-60,01][-49,99]
[SmartAPSelection]                              Client /192.168.2.204 in agent /192.168.1.15
[SmartAPSelection] [-70,00][-50,00][-60,13][-50,00]
[SmartAPSelection] ====================
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - DETECTOR - AP VIP: /192.168.1.2
[SmartAPSelection] [  0   ][  3   ][  0   ][  0   ]
[SmartAPSelection] Assignation done in: 2 ms
[SmartAPSelection] ====================
```

```
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
```

Figure 34: Initial status of the STAs

Later, STA `192.168.2.202` starts an SSH flow and this event is detected (see the `Detector Agent` log) and reported to the Wi-5 controller, which moves it to the “VIP AP” (AP2, gold). From that moment, this STA will have the benefit of being associated to that AP (see Figure 35).

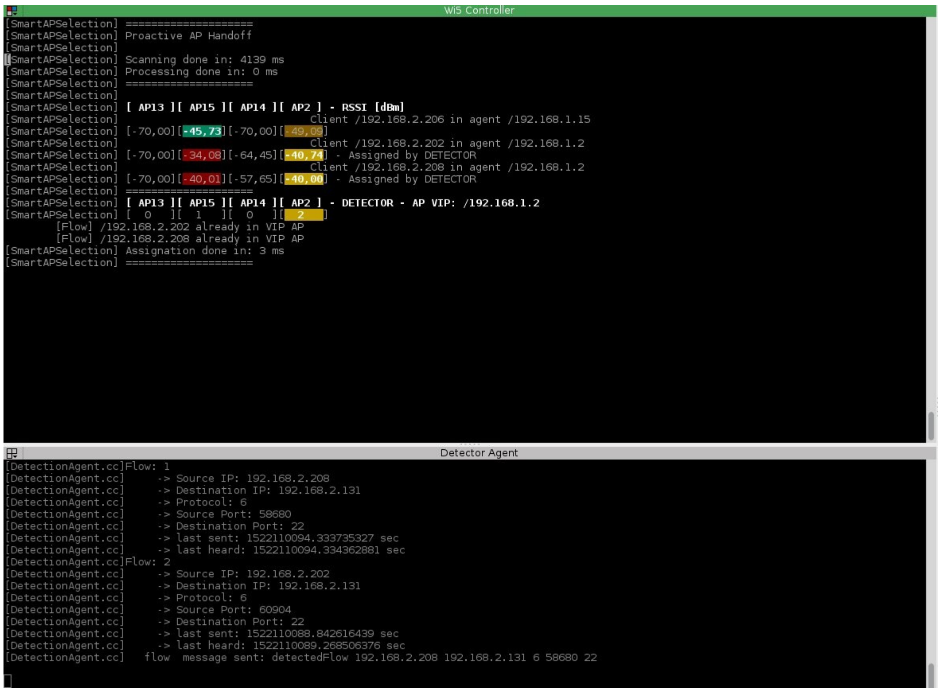


```
[SmartAPSelection] ====================
[SmartAPSelection] Proactive AP Handoff
[SmartAPSelection]
[SmartAPSelection] Scanning done in: 3716 ms
[SmartAPSelection] Processing done in: 0 ms
[SmartAPSelection] ====================
[SmartAPSelection]
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - RSSI [dBm]
[SmartAPSelection]                               Client /192.168.2.206 in agent /192.168.1.15
[SmartAPSelection] [-70,05][-40,35][-56,19][-49,54]
[SmartAPSelection]                               Client /192.168.2.202 in agent /192.168.1.2
[SmartAPSelection] [-70,00][-44,79][-60,08][-46,09] - Assigned by DETECTOR
[SmartAPSelection]                               Client /192.168.2.208 in agent /192.168.1.15
[SmartAPSelection] [-70,00][-50,00][-60,00][-48,67]
[SmartAPSelection] ====================
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - DETECTOR - AP VIP: /192.168.1.2
[SmartAPSelection] [  0   ][  2   ][  0   ][  1   ]
        [Flow] /192.168.2.202 already in VIP AP
[SmartAPSelection] Assignation done in: 3 ms
[SmartAPSelection] ====================
```

```
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc] ##### Periodic report. Number of flows: 0
[DetectionAgent.cc]   flow  message sent: detectedFlow 192.168.2.202 192.168.2.131 6 59136 22

[DetectionAgent.cc] ##### Periodic report. Number of flows: 1
[DetectionAgent.cc]Flow: 1
[DetectionAgent.cc]     -> Source IP: 192.168.2.202
[DetectionAgent.cc]     -> Destination IP: 192.168.2.131
[DetectionAgent.cc]     -> Protocol: 6
[DetectionAgent.cc]     -> Source Port: 59136
[DetectionAgent.cc]     -> Destination Port: 22
[DetectionAgent.cc]     -> last sent: 1522108581.681690090 sec
[DetectionAgent.cc]     -> last heard: 1522108582.889114912 sec
```

Figure 35: One STA has been moved to the VIP AP

Finally, another SSH flow is detected from STA `192.168.2.208`, so the controller moves it too to the “VIP AP” (gold), see Figure 36.



```
[SmartAPSelection] ====================
[SmartAPSelection] Proactive AP Handoff
[SmartAPSelection]
[SmartAPSelection] Scanning done in: 4139 ms
[SmartAPSelection] Processing done in: 0 ms
[SmartAPSelection] ====================
[SmartAPSelection]
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - RSSI [dBm]
[SmartAPSelection]                               Client /192.168.2.206 in agent /192.168.1.15
[SmartAPSelection] [-70,00][-45,73][-70,00][-49,09]
[SmartAPSelection]                               Client /192.168.2.202 in agent /192.168.1.2
[SmartAPSelection] [-70,00][-34,08][-64,45][-40,74] - Assigned by DETECTOR
[SmartAPSelection]                               Client /192.168.2.208 in agent /192.168.1.2
[SmartAPSelection] [-70,00][-40,01][-57,65][-40,00] - Assigned by DETECTOR
[SmartAPSelection] ====================
[SmartAPSelection] [ AP13 ][ AP15 ][ AP14 ][ AP2 ] - DETECTOR - AP VIP: /192.168.1.2
[SmartAPSelection] [  0   ][  1   ][  0   ][  2   ]
        [Flow] /192.168.2.202 already in VIP AP
        [Flow] /192.168.2.208 already in VIP AP
[SmartAPSelection] Assignation done in: 3 ms
[SmartAPSelection] ====================
```

```
[DetectionAgent.cc]Flow: 1
[DetectionAgent.cc]     -> Source IP: 192.168.2.208
[DetectionAgent.cc]     -> Destination IP: 192.168.2.131
[DetectionAgent.cc]     -> Protocol: 6
[DetectionAgent.cc]     -> Source Port: 58680
[DetectionAgent.cc]     -> Destination Port: 22
[DetectionAgent.cc]     -> last sent: 1522110094.333735327 sec
[DetectionAgent.cc]     -> last heard: 1522110094.334362881 sec
[DetectionAgent.cc]Flow: 2
[DetectionAgent.cc]     -> Source IP: 192.168.2.202
[DetectionAgent.cc]     -> Destination IP: 192.168.2.131
[DetectionAgent.cc]     -> Protocol: 6
[DetectionAgent.cc]     -> Source Port: 60904
[DetectionAgent.cc]     -> Destination Port: 22
[DetectionAgent.cc]     -> last sent: 1522110088.842616439 sec
[DetectionAgent.cc]     -> last heard: 1522110089.268506376 sec
[DetectionAgent.cc]   flow  message sent: detectedFlow 192.168.2.208 192.168.2.131 6 58680 22
```

Figure 36: A second STA has been moved to the VIP AP

5 Measurements of System Delays

We are using a distributed solution in which a central controller coordinates a number of APs. This scheme may introduce some delays that have to be minimised in order to grant a good behaviour and an acceptable experience to the user. As stated before, the system has been designed for users moving at walking speed, which implies that the features *e.g.,* the handovers between APs have to be fast enough in order to prevent disruption of the service. In this subsection we include a set of tests aimed at measuring the delays of the system. This includes measurements of the time required to collect monitoring information and the time to dispatch configuration changes to the equipment.

In order to obtain detailed statistical values of the frames received by the Odin agents, we have included a new option which can be used by activating the option `CAPTURE_MODE` in the file that defines the behaviour of the Agent. More details are given in Annex 3.

5.1.1 Delays incurred when collecting monitoring information

The design of the system, as shown earlier in Figure 3, includes a number of APs and a central controller. A distribution system (*i.e.,* a wired connection) exists between the Wi-5 controller and each of the APs.

The APs run *Click Modular Router* [37], including a special element called `odinagent`, where the specific Wi-5 functionalities are implemented. The scheme of Figure 37 corresponds to the elements of *Click Modular Router* used by the Wi-5 APs[23]. The frames received by both wireless devices can be observed: the main one (`from_dev`) and the auxiliary one (`from_dev1`). Both of them go through the `odinagent` element after being decapsulated (*i.e.,* the 802.11 headers are removed by `RadiotapDecap` and `ExtraDecap` elements).

The main wireless interface of the AP (`mon0`, which frames arrive to `from_dev` box) receives all the traffic of the users served by the AP. The `odinagent` element has been programmed to continuously monitor the traffic by the calculation of the average of the different values of each wireless frame traversing it (detailed information about this can be found in Deliverable D3.1 [40], Section 3.3). The monitoring information is updated and stored in real-time as long as frames are received by the AP. This information is stored until the controller requests it using the `READ odinagent.rxstats` or the `READ odinagent.txstats` messages.

This scheme has been extended by Wi-5 to include an auxiliary wireless interface (`mon1`, which frames arrive to `from_dev1` box) which also permits the real-time monitoring of the environment and avoids the interruption of the service for the users. This secondary interface can be set in a different channel from the one in which the main interface is serving the users, and this allows features such as multi-channel handover, scanning the occupation level of other channels, calculating the "distance" in dBs with respect to other Wi-5 APs, etc. These functionalities have been detailed in previous sections.

Next, we will present some measurements of the delays related to the monitoring process. The measurements have been obtained using a 3COM SuperStack II PS Hub 40 Ethernet hub[24] used for connecting the APs and the controller in the control plane (see Figure 38). These tests have been

[23] This scheme is implemented by the `.cli` file created with a python script included in https://github.com/Wi5/odin-wi5-agent/blob/master/agent-click-file-gen.py
[24] A hub has been used instead of a switch, in order to allow the sniffer to capture the traffic.

performed in a lab environment in Unizar in order to characterise the monitoring traffic in terms of bandwidth, packet size, etc.

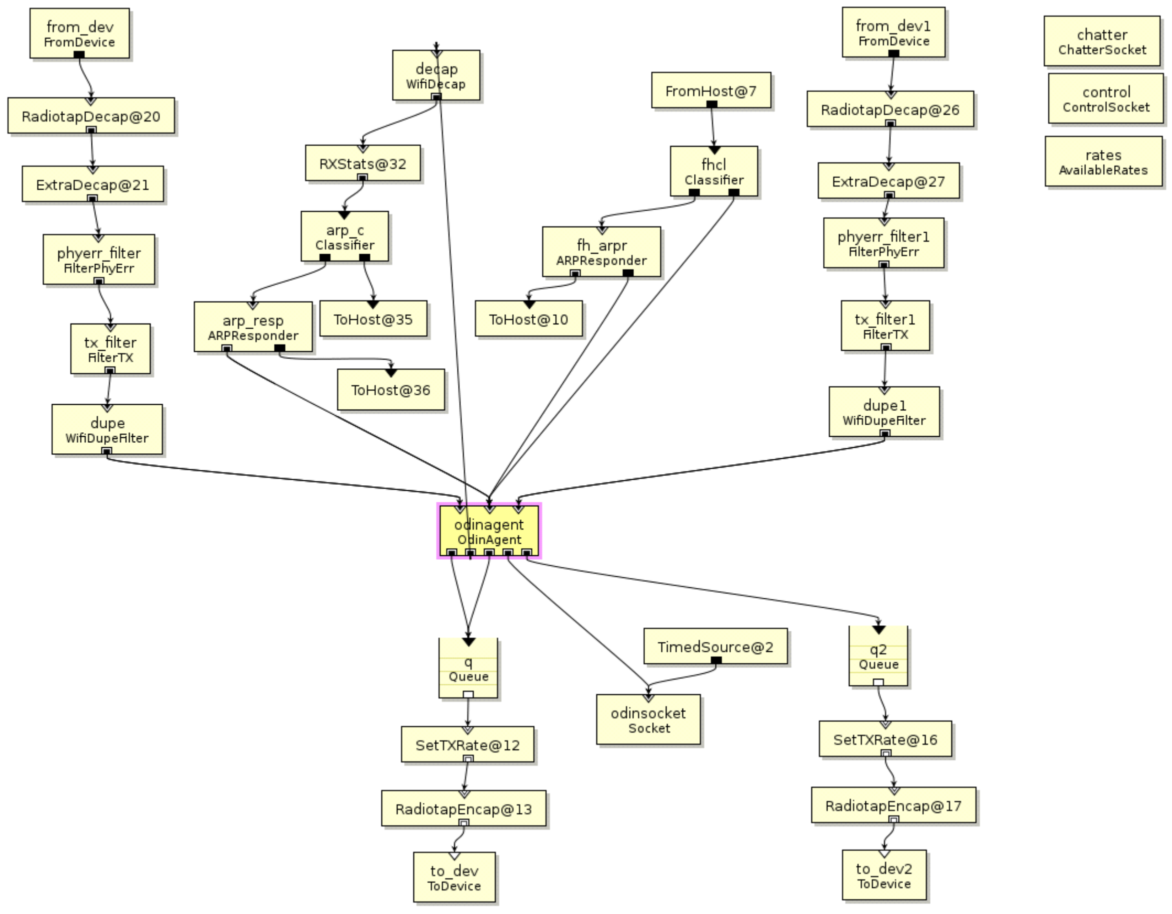


Figure 37 : Scheme of the Click Modular Router elements used by the Wi-5 Agents

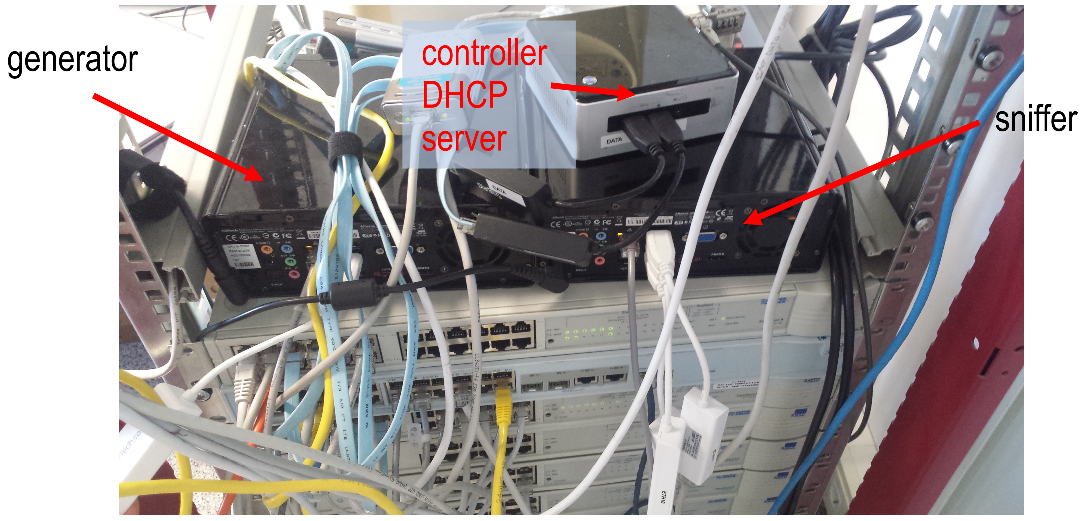


Figure 38 : Measurement environment, including the switches/hubs, the traffic generator, the controller, the DHCP server and the traffic sniffer that captures the traffic

5.1.1.1 Periodic statistics reported to the Wi-5 controller

The statistics are continuously collected and updated by each AP, *i.e.,* every frame that arrives at the AP is used to update the average values. A time interval can be defined as a parameter in order to set the interval in which the statistics will be required by the application running in the controller.

Figure 39 shows a 540 sec. capture of the control plane traffic, while statistics are being gathered by the application (running over the controller) from an AP every 15 seconds (a tick of 1 second has been used for representing the value). The red line corresponds to the traffic sent by the AP, and the green one to the traffic sent by the controller. It can be observed that the monitoring information is always below 80 kbps, which can be considered negligible in a 100 Mbps or 1 Gbps wired link.

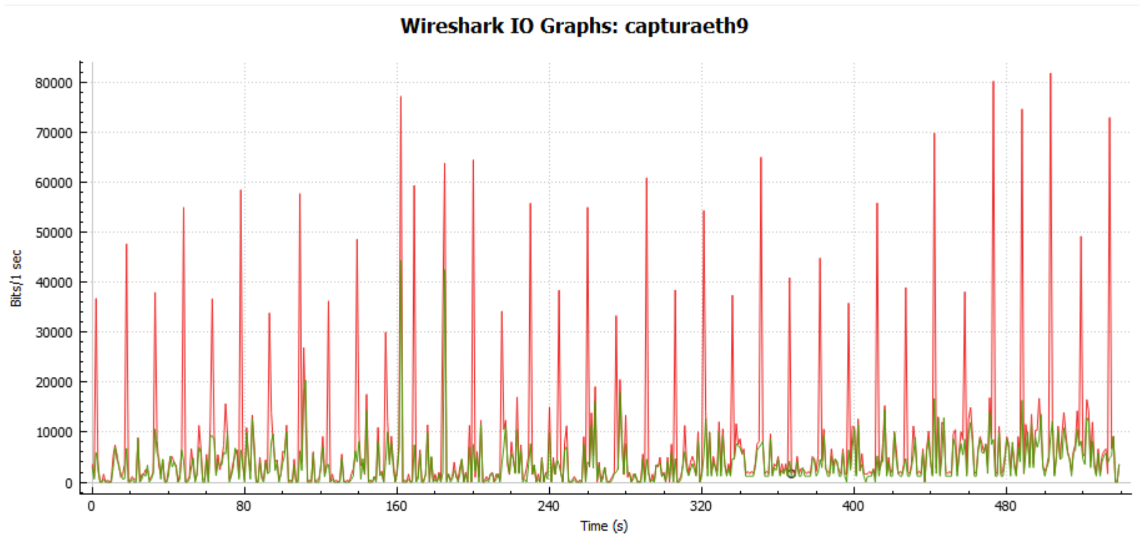


Figure 39 : Traffic capture of statistics monitoring traffic

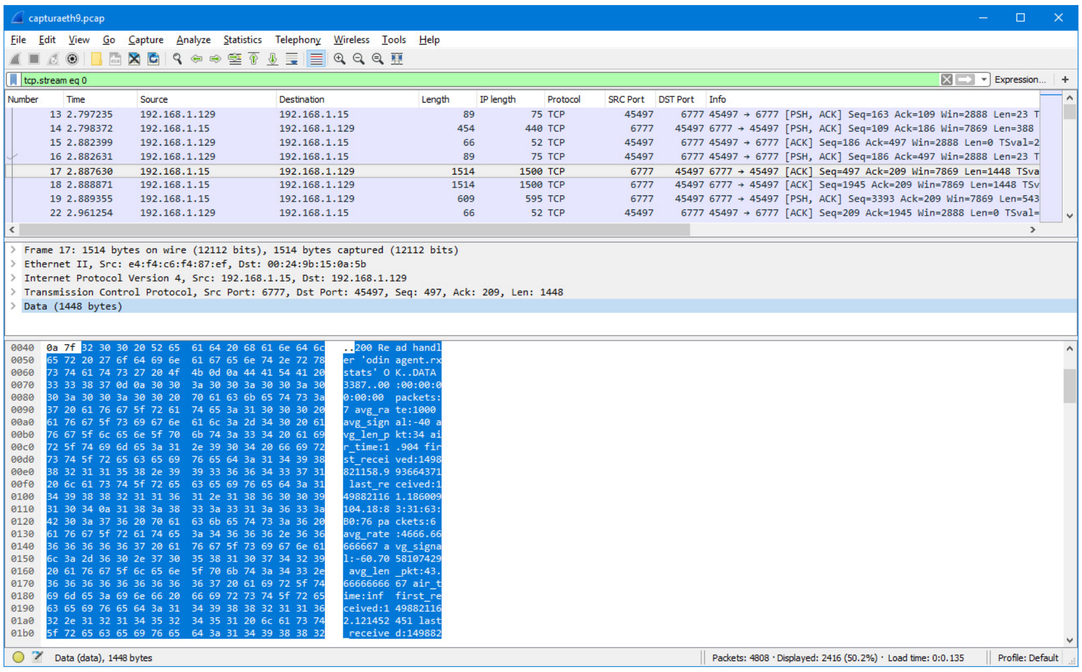
As an example, we next show (Figure 40) the exchange of information between the Wi-5 controller and one of the agents. Four packets are exchanged. First, the controller, sends a `READ odinagent.txstats` packet (89 bytes) requesting the statistics of the frames transmitted by the agent. The agent responds with a `200 Read handler 'odinagent.txstats' OK` packet (454 bytes, 337 of payload), in which the information of the packets sent to each STA is included. It can be observed that two STAs are connected to the AP: a laptop with a WiPi[25] wireless card (`40:A5:EF:05:9B:A0`), and another one with a Sony integrated card (`00:1D:E0:BF:86:45`). After this, the controller requests the information about received frames (`READ odinagent.rxstats`), and the agent sends a `200 Read handler 'odinagent.rxstats' OK` response (3387 bytes of payload, included in two 1500 bytes packets and a 595 packet). It should be noted that this response contains more information, as the agent has also received frames from a number of APs and STAs nearby.

[25] Raspberry Pi WiPi Wireless Adapter, see https://thepihut.com/products/raspberry-pi-wipi-wireless-adapter

```
READ odinagent.txstats
200 Read handler 'odinagent.txstats' OK
DATA 337
00:1D:E0:BF:86:45 packets:4 avg_rate:54000 avg_signal:25 avg_len_pkt:177 air_time:0.104888888889
first_received:1498821147.825710452 last_received:1498821159.834143214
40:A5:EF:05:9B:A0 packets:4 avg_rate:54000 avg_signal:25 avg_len_pkt:116 air_time:0.0687407407407
first_received:1498821147.140414415 last_received:1498821163.161276569
READ odinagent.rxstats
200 Read handler 'odinagent.rxstats' OK
DATA 3387
00:00:00:00:00:00 packets:7 avg_rate:1000 avg_signal:-40 avg_len_pkt:34 air_time:1.904
first_received:1498821158.993664371 last_received:1498821161.186009104
18:83:31:63:B0:76 packets:6 avg_rate:4666.66666667 avg_signal:-60.7058107429 avg_len_pkt:43.6666666667 air_time:inf
first_received:1498821162.121452451 last_received:1498821162.886581759
00:1D:E0:BF:86:45 packets:3 avg_rate:54000 avg_signal:-40 avg_len_pkt:116 air_time:0.0515555555556
first_received:1498821147.823418558 last_received:1498821159.830976780
40:A5:EF:05:9B:A0 packets:5 avg_rate:40800 avg_signal:-35.5284196866 avg_len_pkt:164.8 air_time:0.195333333333
first_received:1498821147.138191763 last_received:1498821163.152824439
00:15:AF:63:45:8A packets:1 avg_rate:1000 avg_signal:-70 avg_len_pkt:24 air_time:0.192
first_received:1498821159.341881880 last_received:1498821159.342235435
D0:7A:B5:8E:F5:30 packets:2 avg_rate:1000 avg_signal:-80 avg_len_pkt:29 air_time:0.464
first_received:1498821160.329956261 last_received:1498821160.566925761
00:22:6B:9C:94:B5 packets:21 avg_rate:3523.80952381 avg_signal:-50 avg_len_pkt:77.4285714286 air_time:12.0186666667
first_received:1498821160.249268918 last_received:1498821163.751820458
78:BA:F9:BC:45:70 packets:3 avg_rate:1000 avg_signal:-80 avg_len_pkt:218 air_time:5.232
first_received:1498821159.142346336 last_received:1498821163.529751564
78:BA:F9:BC:45:71 packets:2 avg_rate:1000 avg_signal:-80 avg_len_pkt:277 air_time:4.432
first_received:1498821159.610165521 last_received:1498821160.761974016
1C:7B:21:97:90:8B packets:1 avg_rate:6000 avg_signal:-80 avg_len_pkt:77 air_time:0.102666666667
first_received:1498821159.031854002 last_received:1498821159.031985745
54:A2:74:1D:56:00 packets:1078 avg_rate:1022.72727273 avg_signal:-40 avg_len_pkt:140.762523191 air_time:inf
first_received:1498821158.905293391 last_received:1498821163.753058613
54:A2:74:1D:56:01 packets:122 avg_rate:1934.42622951 avg_signal:-40 avg_len_pkt:209.073770492 air_time:inf
first_received:1498821158.882788771 last_received:1498821163.690585895
38:2C:4A:65:3A:FC packets:3 avg_rate:1000 avg_signal:-73.9794000867 avg_len_pkt:282 air_time:6.768
first_received:1498821159.642164774 last_received:1498821162.509634559
38:72:C0:C2:30:FF packets:5 avg_rate:1000 avg_signal:-80 avg_len_pkt:193.6 air_time:7.744
first_received:1498821159.220960188 last_received:1498821163.726670871
C0:D9:62:4C:D6:74 packets:2 avg_rate:1000 avg_signal:-50 avg_len_pkt:73 air_time:1.168
first_received:1498821161.049553799 last_received:1498821161.108989541
00:22:6B:9C:90:C2 packets:17 avg_rate:1000 avg_signal:-50.2362304535 avg_len_pkt:75 air_time:10.2
first_received:1498821159.537750628 last_received:1498821163.437625588
54:A2:74:2D:43:C0 packets:2 avg_rate:1000 avg_signal:-80 avg_len_pkt:218 air_time:3.488
first_received:1498821159.181168083 last_received:1498821161.676904034
54:A2:74:2D:43:C1 packets:2 avg_rate:1000 avg_signal:-80 avg_len_pkt:277 air_time:4.432
first_received:1498821159.113154940 last_received:1498821161.099142513
00:22:6B:9C:93:BA packets:13 avg_rate:5076.92307692 avg_signal:-50 avg_len_pkt:78.9230769231 air_time:7.21866666667
first_received:1498821158.917068476 last_received:1498821162.298221806
80:E6:50:11:E2:6A packets:5 avg_rate:24000 avg_signal:-55.5284196866 avg_len_pkt:27.2 air_time:0.0453333333333
first_received:1498821159.998219745 last_received:1498821163.755477343
```

Figure 40 : Content of the packets including statistics information, exchanged between the Wi-5 controller and a Wi-5 agent

The delay required for this exchange is small: 169 milliseconds between the first (#13) and the last (#22) packet (see the Wireshark capture in Figure 41). This should be performed for each of the APs, but it can be done periodically and does not impose a significant amount of data nor delay. The average size of the packet containing the statistics data is 1006 bytes if the whole trace is averaged.


capturaeth9.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
tcp.stream eq 0

Number Time        Source          Destination     Length  IP length Protocol SRC Port DST Port Info
    13 2.797235    192.168.1.129   192.168.1.15        89        75 TCP         45497     6777 45497 → 6777 [PSH, ACK] Seq=163 Ack=109 Win=2888 Len=23 T
    14 2.798372    192.168.1.15    192.168.1.129      454       440 TCP          6777    45497 6777 → 45497 [PSH, ACK] Seq=109 Ack=186 Win=7869 Len=388
    15 2.882399    192.168.1.129   192.168.1.15        66        52 TCP         45497     6777 45497 → 6777 [ACK] Seq=186 Ack=497 Win=2888 Len=0 TSval=2
    16 2.882631    192.168.1.129   192.168.1.15        89        75 TCP         45497     6777 45497 → 6777 [PSH, ACK] Seq=186 Ack=497 Win=2888 Len=23 T
    17 2.887630    192.168.1.15    192.168.1.129     1514      1500 TCP          6777    45497 6777 → 45497 [ACK] Seq=497 Ack=209 Win=7869 Len=1448 TSva
    18 2.888871    192.168.1.15    192.168.1.129     1514      1500 TCP          6777    45497 6777 → 45497 [ACK] Seq=1945 Ack=209 Win=7869 Len=1448 TSv
    19 2.889355    192.168.1.15    192.168.1.129      609       595 TCP          6777    45497 6777 → 45497 [PSH, ACK] Seq=3393 Ack=209 Win=7869 Len=543
    22 2.961254    192.168.1.129   192.168.1.15        66        52 TCP         45497     6777 45497 → 6777 [ACK] Seq=209 Ack=1945 Win=2888 Len=0 TSval=

> Frame 17: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: e4:f4:c6:f4:87:ef, Dst: 00:24:9b:15:0a:5b
> Internet Protocol Version 4, Src: 192.168.1.15, Dst: 192.168.1.129
> Transmission Control Protocol, Src Port: 6777, Dst Port: 45497, Seq: 497, Ack: 209, Len: 1448
> Data (1448 bytes)

0040  0a 7f 32 30 30 20 52 65  61 64 20 68 61 6e 64 6c   ..200 Re ad handl
0050  65 72 20 27 6f 64 69 6e  61 67 65 6e 74 2e 72 78   er 'odin agent.rx
0060  73 74 61 74 73 27 20 4f  4b 0d 0a 44 41 54 41 20   stats' O K..DATA 
0070  33 33 38 37 0d 0a 30 30  3a 30 30 3a 30 30 3a 30   3387..00 :00:00:0
0080  30 3a 30 30 3a 30 30 20  70 61 63 6b 65 74 73 3a   0:00:00  packets:
0090  37 20 61 76 67 5f 72 61  74 65 3a 31 30 30 30 20   7 avg_ra te:1000 
00a0  61 76 67 5f 73 69 67 6e  61 6c 3a 2d 34 30 20 61   avg_sign al:-40 a
00b0  76 67 5f 6c 65 6e 5f 70  6b 74 3a 33 34 20 61 69   vg_len_p kt:34 ai
00c0  72 5f 74 69 6d 65 3a 31  2e 39 30 34 20 66 69 72   r_time:1 .904 fir
00d0  73 74 5f 72 65 63 65 69  76 65 64 3a 31 34 39 38   st_recei ved:1498
00e0  38 32 31 31 35 38 2e 39  39 33 36 36 34 33 37 31   821158.9 93664371
00f0  20 6c 61 73 74 5f 72 65  63 65 69 76 65 64 3a 31    last_re ceived:1
0100  34 39 38 38 32 31 31 36  31 2e 31 38 36 30 30 39   49882116 1.186009
0110  31 30 34 0a 31 38 3a 38  33 3a 33 31 3a 36 33 3a   104.18:8 3:31:63:
0120  42 30 3a 37 36 20 70 61  63 6b 65 74 73 3a 36 20   B0:76 pa ckets:6 
0130  61 76 67 5f 72 61 74 65  3a 34 36 36 36 2e 36 36   avg_rate :4666.66
0140  36 36 36 36 36 37 20 61  76 67 5f 73 69 67 6e 61   666667 a vg_signa
0150  6c 3a 2d 36 30 2e 37 30  35 38 31 30 37 34 32 39   l:-60.70 58107429
0160  20 61 76 67 5f 6c 65 6e  5f 70 6b 74 3a 34 33 2e    avg_len _pkt:43.
0170  36 36 36 36 36 36 36 36  36 37 20 61 69 72 5f 74   66666666 67 air_t
0180  69 6d 65 3a 69 6e 66 20  66 69 72 73 74 5f 72 65   ime:inf  first_re
0190  63 65 69 76 65 64 3a 31  34 39 38 38 32 31 31 36   ceived:1 49882116
01a0  32 2e 31 32 31 34 35 32  34 35 31 20 6c 61 73 74   2.121452 451 last
01b0  5f 72 65 63 65 69 76 65  64 3a 31 34 39 38 38 32   _receive d:149882

Data (data), 1448 bytes    Packets: 4808 · Displayed: 2416 (50.2%) · Load time: 0:0.135    Profile: Default
```

Figure 41 : Capture of the packets including statistics information, exchanged between the Wi-5 controller and a Wi-5 agent

5.1.1.2 *Time required by the application "Distance in dBs"*

This application, explained in detail in D3.3 and available in the controller[26], is able to build a matrix including the distance in dBs between the APs under the control of a Wi-5 controller. For this aim, the auxiliary interface of the APs are used.

This is the content of the `poolfile` used for the tests:

```
# Pool-1
NAME pool-1
NODES 192.168.1.13 192.168.1.14 192.168.1.15
NETWORKS wi5-demo

####### ShowMatrixOfDistancedBs params
####### MATRIX TimeToStart(sec) ReportingPeriod(sec) ScanningInterval(sec) AddedTime(sec) Channel
APPLICATION net.floodlightcontroller.odin.applications.ShowMatrixOfDistancedBs
MATRIX 30 30 5 1 6
```

As can be observed, there is a single pool of three APs (called `Nodes`), creating the network `wi5-demo`. The parameters are:

- `TimeToStart`: the application will wait 30 seconds before starting for the first time.
- `ReportingPeriod`: the information will be reported to the controller every 30 seconds.
- `ScanningInterval`: the AP will scan for 5 seconds on each channel.
- `AddedTime`: An interval of 1 second is left between the end of the scanning in one channel, and the beginning of the scanning in the next one.
- `Channel`: The number of the channel that will be used for the measurements.

Two different kinds of communication are used by this application: before starting the sending of the beacons, the three scanning flags (namely `ClientScanningFlag`, `APScanningFlag` and

[26] Wi-5 *ShowMatrixOfDistancedBs* application, see https://github.com/Wi5/odin-wi5/wiki/Application-ShowMatrixOfDistancedBs

measurementBeaconFlag) have to be read from the AP. If they are 0, it means that the AP is not performing other scanning tasks, so it is available. We next show the content of the two packets, the one in which the controller requests the value, and then the answer of the AP:

```
READ odinagent.scanning_flags
200 Read handler 'odinagent.scanning_flags' OK
DATA 6
0 0 0
```

If the APs are available, then the controller tells one the APs to start sending beacons with the "odin_init" SSID:

```
WRITE odinagent.scan_APs odin_init 6
200 Write handler 'odinagent.scan_APs' OK
```

And at the same time the controller tells the other APs to listen to these beacons:

```
WRITE odinagent.send_measurement_beacon odin_init 6
200 Write handler 'odinagent.send_measurement_beacon' OK
```

After the ScanningInterval, the controller requests the information from the APs, and the AP sends its report:

```
READ odinagent.scan_APs
200 Read handler 'odinagent.scan_APs' OK
DATA 44
DC:EF:09:E6:9C:DB avg_signal:-57.2124639905
```

The overall time for performing this operation will obviously depend on the parameters set in the controller. Some tests have been carried out in Unizar's lab environment in order to characterise the traffic and the delays. In Figure 42, the monitoring traffic (control plane) between the Wi-5 controller and the APs is presented. The red line represents the traffic generated by the APs, and the green one, the traffic generated by the controller.

Starting from sec. 30, a peak can be observed every 6 s, corresponding to the sending of the results to the controller. The required bandwidth however is still very small (maximum 4 kbps).

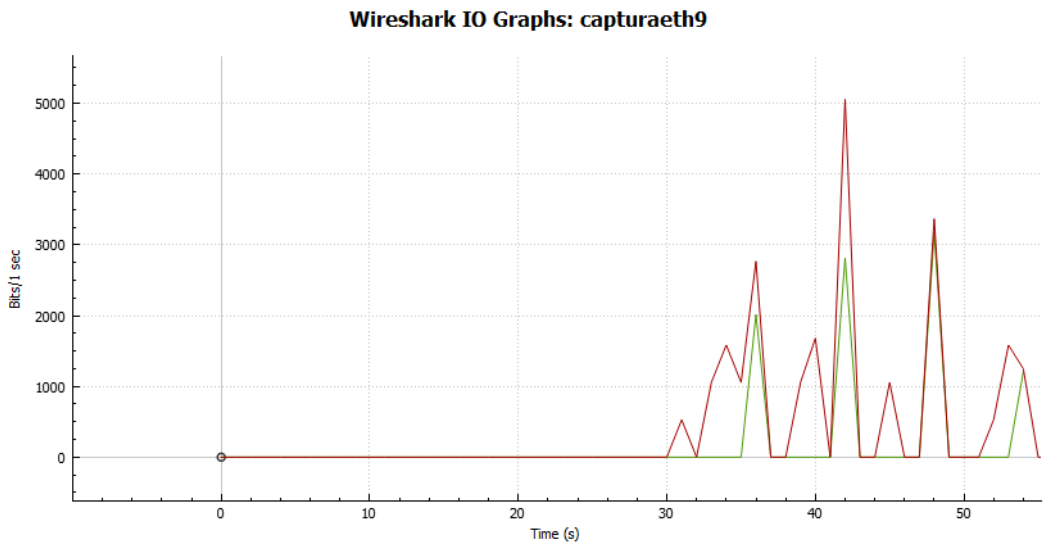
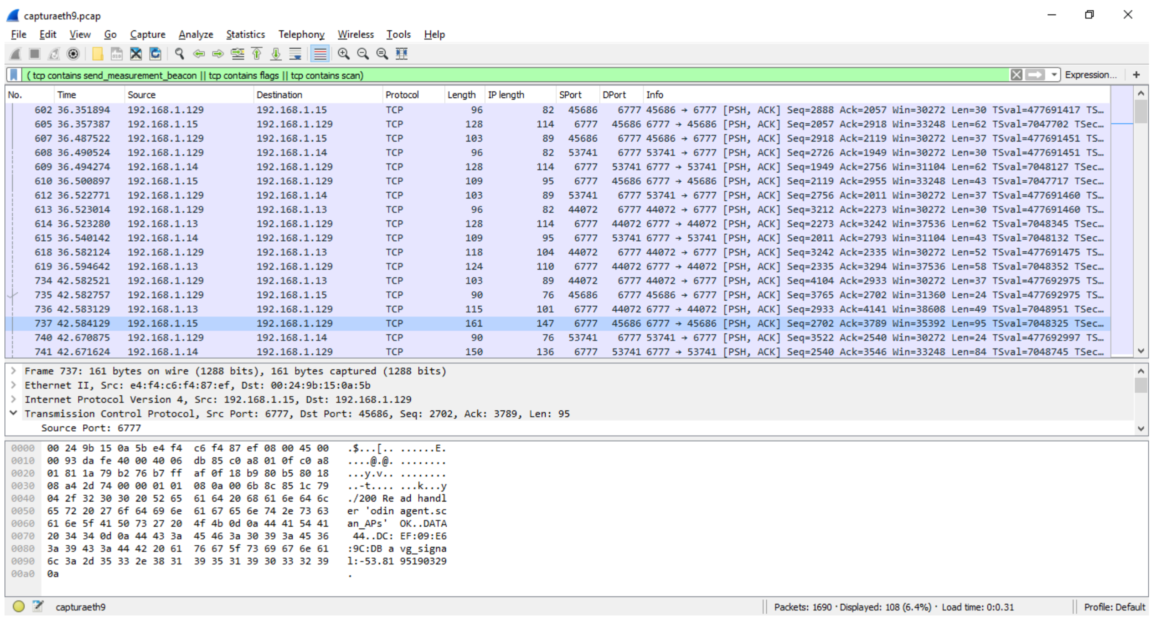


Figure 42 : Traffic capture of statistics the control traffic exchange when the *distance in dBs* application is running

All the packets involved in this communication are shown in Figure 43. It can be observed that the initial process for requesting the status of the APs' flags and starting the scanning (packets 602 to 619) requires 243 ms. 5 seconds later, the controller starts gathering the information from the APs that have been scanning (packets 734 to 741), and this requires 89 ms. So overall, the delay is 5.332 seconds. All in all, it can be said that the overall delay is of 110 ms per AP (332 ms for 3 APs), which are added to the scanning period (5 seconds in this case).


capturaeth9.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression…
+

```
( tcp contains send_measurement_beacon || tcp contains flags || tcp contains scan)

No.  Time       Source         Destination    Protocol Length IP length SPort DPort Info
602 36.351894 192.168.1.129  192.168.1.15   TCP   96  82 45686  6777 45686 → 6777 [PSH, ACK] Seq=2888 Ack=2057 Win=30272 Len=30 TSval=477691417 TS…
605 36.357387 192.168.1.15   192.168.1.129  TCP  128 114  6777 45686 6777 → 45686 [PSH, ACK] Seq=2057 Ack=2918 Win=33248 Len=62 TSval=7047702 TSec…
607 36.487522 192.168.1.129  192.168.1.15   TCP  103  89 45686  6777 45686 → 6777 [PSH, ACK] Seq=2918 Ack=2119 Win=30272 Len=37 TSval=477691451 TS…
608 36.490524 192.168.1.129  192.168.1.14   TCP   96  82 53741  6777 53741 → 6777 [PSH, ACK] Seq=2726 Ack=1949 Win=30272 Len=30 TSval=477691451 TS…
609 36.494274 192.168.1.14   192.168.1.129  TCP  128 114  6777 53741 6777 → 53741 [PSH, ACK] Seq=1949 Ack=2756 Win=31104 Len=62 TSval=7048127 TSec…
610 36.500897 192.168.1.15   192.168.1.129  TCP  109  95  6777 45686 6777 → 45686 [PSH, ACK] Seq=2119 Ack=2955 Win=33248 Len=43 TSval=7047717 TSec…
612 36.522771 192.168.1.129  192.168.1.14   TCP  103  89 53741  6777 53741 → 6777 [PSH, ACK] Seq=2756 Ack=2011 Win=30272 Len=37 TSval=477691460 TS…
613 36.523014 192.168.1.129  192.168.1.13   TCP   96  82 44072  6777 44072 → 6777 [PSH, ACK] Seq=3212 Ack=2273 Win=30272 Len=30 TSval=477691460 TS…
614 36.523280 192.168.1.13   192.168.1.129  TCP  128 114  6777 44072 6777 → 44072 [PSH, ACK] Seq=2273 Ack=3242 Win=37536 Len=62 TSval=7048345 TSec…
615 36.540142 192.168.1.14   192.168.1.129  TCP  109  95  6777 53741 6777 → 53741 [PSH, ACK] Seq=2011 Ack=2793 Win=31104 Len=43 TSval=7048132 TSec…
618 36.582124 192.168.1.129  192.168.1.13   TCP  118 104 44072  6777 44072 → 6777 [PSH, ACK] Seq=3242 Ack=2335 Win=30272 Len=52 TSval=477691475 TS…
619 36.594642 192.168.1.13   192.168.1.129  TCP  124 110  6777 44072 6777 → 44072 [PSH, ACK] Seq=2335 Ack=3294 Win=37536 Len=58 TSval=7048352 TSec…
734 42.582521 192.168.1.129  192.168.1.13   TCP  103  89 44072  6777 44072 → 6777 [PSH, ACK] Seq=4104 Ack=2933 Win=30272 Len=37 TSval=477692975 TS…
735 42.582757 192.168.1.129  192.168.1.15   TCP   90  76 45686  6777 45686 → 6777 [PSH, ACK] Seq=3765 Ack=2702 Win=31360 Len=24 TSval=477692975 TS…
736 42.583129 192.168.1.13   192.168.1.129  TCP  115 101  6777 44072 6777 → 44072 [PSH, ACK] Seq=2933 Ack=4141 Win=38608 Len=49 TSval=7048951 TSec…
737 42.584129 192.168.1.15   192.168.1.129  TCP  161 147  6777 45686 6777 → 45686 [PSH, ACK] Seq=2702 Ack=3789 Win=35392 Len=95 TSval=7048325 TSec…
740 42.670875 192.168.1.129  192.168.1.14   TCP   90  76 53741  6777 53741 → 6777 [PSH, ACK] Seq=3522 Ack=2540 Win=30272 Len=24 TSval=477692997 TS…
741 42.671624 192.168.1.14   192.168.1.129  TCP  150 136  6777 53741 6777 → 53741 [PSH, ACK] Seq=2540 Ack=3546 Win=33248 Len=84 TSval=7048745 TSec…

> Frame 737: 161 bytes on wire (1288 bits), 161 bytes captured (1288 bits)
> Ethernet II, Src: e4:f4:c6:f4:87:ef, Dst: 00:24:9b:15:0a:5b
> Internet Protocol Version 4, Src: 192.168.1.15, Dst: 192.168.1.129
v Transmission Control Protocol, Src Port: 6777, Dst Port: 45686, Seq: 2702, Ack: 3789, Len: 95
    Source Port: 6777

0000  00 24 9b 15 0a 5b e4 f4  c6 f4 87 ef 08 00 45 00   .$...[.. ......E.
0010  00 93 da fe 40 00 40 06  db 85 c0 a8 01 0f c0 a8   ....@.@. ........
0020  01 81 1a 79 b2 76 b7 ff  af 0f 18 b9 80 b5 80 18   ...y.v.. ........
0030  08 a4 2d 74 00 00 01 01  08 0a 00 6b 8c 85 1c 79   ..-t.... ...k...y
0040  04 2f 32 30 30 20 52 65  61 64 20 68 61 6e 64 6c   ./200 Re ad handl
0050  65 72 20 27 6f 64 69 6e  61 67 65 6e 74 2e 73 63   er 'odin agent.sc
0060  61 6e 5f 41 50 73 27 20  4f 4b 0d 0a 44 41 54 41   an_APs'  OK..DATA
0070  20 34 34 0d 0a 44 43 3a  45 46 3a 30 39 3a 45 36    44..DC: EF:09:E6
0080  3a 39 43 3a 44 42 20 61  76 67 5f 73 69 67 6e 61   :9C:DB a vg_signa
0090  6c 3a 2d 35 33 2e 38 31  39 35 31 39 30 33 32 39   l:-53.81 95190329
00a0  0a                                                 .

capturaeth9    Packets: 1690 · Displayed: 108 (6.4%) · Load time: 0:0.31    Profile: Default
```

Figure 43 : Capture of the packets exchanged between the Wi-5 controller and a Wi-5 agent during the running of the *distance in dBs* application

5.1.1.3 Time required for an application that scans the wireless environment

The application *ShowScannedStationsStatistics*[27], explained in section 4.1, uses the secondary interface of each AP to scan all the channels, and obtains all the frames heard, and their statistics.

The time required for this operation depends on the number of channels (1 to 11 by default), and on the time specified to scan each channel, which is a parameter of the controller's `poolfile`:

```
#Pool-1
NAME pool-1
NODES 192.168.1.13 192.168.1.14 192.168.1.15
NETWORKS wi5-demo

####### ShowScannedStationsStatistics params - Optional filename to save statistics
####### INTERFERENCES TimeToStart(seg) ReportingPeriod(seg) ScanningInterval(seg) AddedTime(seg)
Filename
APPLICATION net.floodlightcontroller.odin.applications.ShowScannedStationsStatistics
INTERFERENCES 30 30 5 1 ScannedStationsStatistics.txt
```

Again, there is a single pool of three APs (called `Nodes`) here, creating the network `wi5-demo`. The parameters are the same as those used in the application explained previously (except for `channel`, which is not used here):

- `TimeToStart`: the application will wait 30 seconds before starting for the first time.
- `ReportingPeriod`: the information will be reported to the controller every 30 seconds.

[27] Wi-5 *ShowScannedStationsStatistics* application, see https://github.com/Wi5/odin-wi5/wiki/Application-ShowScannedStationsStatistics

- `ScanningInterval`: the AP will scan for 5 seconds on each channel.
- `AddedTime`: An interval of 1 second is left between the end of the scanning in one channel, and the beginning of the scanning in the next one.

We have again performed some tests in Unizar's lab environment in order to characterise the traffic and the delays. In Figure 44, the monitoring traffic between the Wi-5 controller and the APs is presented. The red line represents the traffic sent by the APs, and the green one is the traffic sent by the controller (the monitoring traffic also includes other traffic as *e.g.,* the probe request and probe response packets sent to/from the controller).

It can be observed that after the initial period the scanning starts (the peak in second 25) in channel 1. The, every 6 seconds the results of the scanning in the other channels are reported. It is worth noting that channels 1, 6 and 11 are the ones that produce the biggest amount of information, as these are the most crowded ones (as they are orthogonal, many APs are configured to use one of them by default).

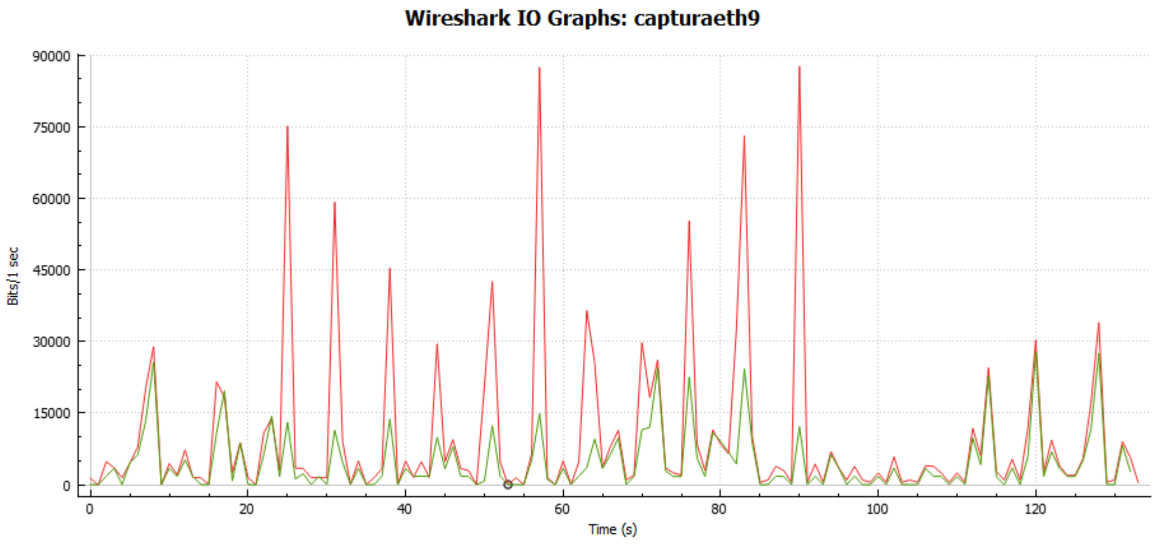
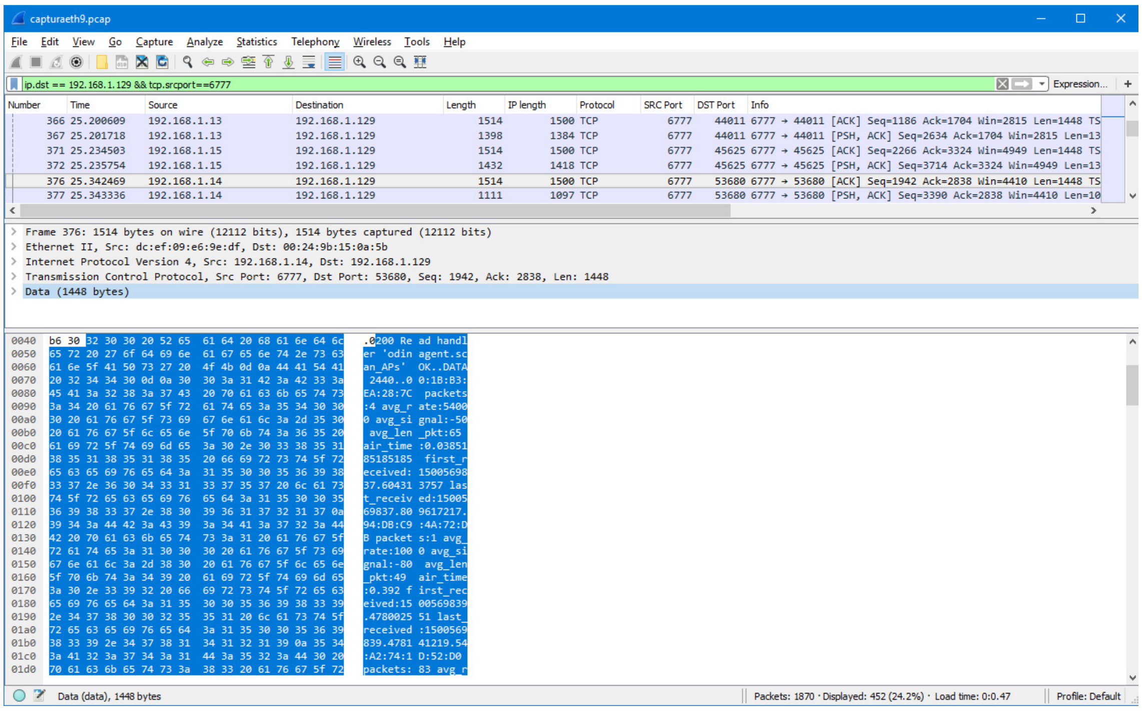


Figure 44 : Traffic capture of statistics the control traffic exchange when the scanning application is running

To further illustrate this, Figure 45 shows the six packets sent by the APs in second 25, reporting the monitoring information to the controller. It can be observed that each AP sends two packets in this case, as the data exceeds a single 1500 byte one.


capturaeth9.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

```
ip.dst == 192.168.1.129 && tcp.srcport==6777

Number Time      Source        Destination    Length IP length Protocol SRC Port DST Port Info
   366 25.200609 192.168.1.13  192.168.1.129    1514      1500 TCP          6777    44011 6777 → 44011 [ACK] Seq=1186 Ack=1704 Win=2815 Len=1448 TS
   367 25.201718 192.168.1.13  192.168.1.129    1398      1384 TCP          6777    44011 6777 → 44011 [PSH, ACK] Seq=2634 Ack=1704 Win=2815 Len=13
   371 25.234503 192.168.1.15  192.168.1.129    1514      1500 TCP          6777    45625 6777 → 45625 [ACK] Seq=2266 Ack=3324 Win=4949 Len=1448 TS
   372 25.235754 192.168.1.15  192.168.1.129    1432      1418 TCP          6777    45625 6777 → 45625 [PSH, ACK] Seq=3714 Ack=3324 Win=4949 Len=13
   376 25.342469 192.168.1.14  192.168.1.129    1514      1500 TCP          6777    53680 6777 → 53680 [ACK] Seq=1942 Ack=2838 Win=4410 Len=1448 TS
   377 25.343336 192.168.1.14  192.168.1.129    1111      1097 TCP          6777    53680 6777 → 53680 [PSH, ACK] Seq=3390 Ack=2838 Win=4410 Len=10

> Frame 376: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: dc:ef:09:e6:9e:df, Dst: 00:24:9b:15:0a:5b
> Internet Protocol Version 4, Src: 192.168.1.14, Dst: 192.168.1.129
> Transmission Control Protocol, Src Port: 6777, Dst Port: 53680, Seq: 1942, Ack: 2838, Len: 1448
> Data (1448 bytes)

0040  b6 30 32 30 30 20 52 65  61 64 20 68 61 6e 64 6c   .0200 Re ad handl
0050  65 72 20 27 6f 64 69 6e  61 67 65 6e 74 2e 73 63   er 'odin agent.sc
0060  61 6e 5f 41 50 73 27 20  4f 4b 0d 0a 44 41 54 41   an_APs'  OK..DATA
0070  20 32 34 34 30 0d 0a 30  30 3a 31 42 3a 42 33 3a    2440..0 0:1B:B3:
0080  45 41 3a 32 38 3a 37 43  20 70 61 63 6b 65 74 73   EA:28:7C  packets
0090  3a 34 20 61 76 67 5f 72  61 74 65 3a 35 34 30 30   :4 avg_r ate:5400
00a0  30 20 61 76 67 5f 73 69  67 6e 61 6c 3a 2d 35 30   0 avg_si gnal:-50
00b0  20 61 76 67 5f 6c 65 6e  5f 70 6b 74 3a 36 35 20    avg_len _pkt:65 
00c0  61 69 72 5f 74 69 6d 65  3a 30 2e 30 33 38 35 31   air_time :0.03851
00d0  38 35 31 38 35 31 38 35  20 66 69 72 73 74 5f 72   85185185  first_r
00e0  65 63 65 69 76 65 64 3a  31 35 30 30 35 36 39 38   eceived: 15005698
00f0  33 37 2e 36 30 34 33 31  33 37 35 37 20 6c 61 73   37.60431 3757 las
0100  74 5f 72 65 63 65 69 76  65 64 3a 31 35 30 30 35   t_receiv ed:15005
0110  36 39 38 33 37 2e 38 30  39 36 31 37 32 31 37 0a   69837.80 9617217.
0120  39 34 3a 44 42 3a 43 39  3a 34 41 3a 37 32 3a 44   94:DB:C9 :4A:72:D
0130  42 20 70 61 63 6b 65 74  73 3a 31 20 61 76 67 5f   B packet s:1 avg_
0140  72 61 74 65 3a 31 30 30  30 20 61 76 67 5f 73 69   rate:100 0 avg_si
0150  67 6e 61 6c 3a 2d 38 30  20 61 76 67 5f 6c 65 6e   gnal:-80  avg_len
0160  5f 70 6b 74 3a 34 39 20  61 69 72 5f 74 69 6d 65   _pkt:49  air_time
0170  3a 30 2e 33 39 32 20 66  69 72 73 74 5f 72 65 63   :0.392 f irst_rec
0180  65 69 76 65 64 3a 31 35  30 30 35 36 39 38 33 39   eived:15 00569839
0190  2e 34 37 38 30 30 32 35  35 31 20 6c 61 73 74 5f   .4780025 51 last_
01a0  72 65 63 65 69 76 65 64  3a 31 35 30 30 35 36 39   received :1500569
01b0  38 33 39 2e 34 37 38 31  34 31 32 31 39 0a 35 34   839.4781 41219.54
01c0  3a 41 32 3a 37 34 3a 31  44 3a 35 32 3a 44 30 20   :A2:74:1 D:52:D0 
01d0  70 61 63 6b 65 74 73 3a  38 33 20 61 76 67 5f 72   packets: 83 avg_r

Data (data), 1448 bytes    Packets: 1870 · Displayed: 452 (24.2%) · Load time: 0:0.47    Profile: Default
```

Figure 45 : Capture of the packets including monitoring information of the environment, exchanged between the Wi-5 controller and a Wi-5 agent

The content of the packets has already been shown in section 4.1. Here we will characterise the amount of traffic and the delays incurred. In Table 4 we show the characteristics of the traffic when 3 APs monitor the wireless environment at the same time, spending 5 seconds scanning per channel and 1 second moving between channels. Once each channel has been scanned for 5 seconds, the delay of the exchange of information between the APs and the controller is between 80 and 120 milliseconds in our setup.

Table 4. Parameters of the environment scanning application's traffic

	Traffic sent by the APs	**Traffic sent by the controller**
Number of packets	22 x 3 APs = 66 pkts	66 pkts
Packet size	747.5 bytes	95.5 bytes
Bandwidth	5.532 kbps	0.702 kbps
pps	0.9 pps	0.9 pps

5.1.2 Delays when dispatching configuration changes to the equipment

In the previous subsection we have studied the delays incurred by the monitoring processes. In this one we will focus on the delays required to apply configuration changes to the equipment. This includes *a)* the handover of the STAs between APs, and *b)* switching the channel of an AP (and its associated STAs).

5.1.2.1 Handover of a STA between APs

The handover delay was the main topic of the journal paper *"Building a SDN Enterprise WLAN Based on Virtual APs,"* which appeared in IEEE *Communications Letters* [23] in February 2017. Detailed information about the delay of this process was also provided in Deliverable D3.3 "Specification of

Smart AP solutions version 2." These results were obtained using the controller application `HandoverMultichannel` [28].

As a summary, we include Table 5 which shows the handover delay for each burst inter-beacon time, and the cumulative percentage of the detected handovers for three different WiFi dongles, namely Linksys WUSB54GC, WiPi WLAN USB b/g/n, and TP-LINK TL-WN722N. The meaning of the content of the table is this: as an example, for the Linksys device, when the inter-packet time of the burst is set to 10 ms, 90% of the handovers last less than 73.80 ms.

Table 5: Handover delay: percentage (Acc) of times that handover time can reach a time value.

Burst (ms)	**Handover time**					
	Linksys		**WiPi**		**TP-Link**	
	Acc* (%)	**Time (ms)**	**Acc* (%)**	**Time (ms)**	**Acc* (%)**	**Time (ms)**
5	90.00%	223.89	80.00%	28.55	0.00%	89.86
10	90.00%	73.80	70.00%	22.29	0.00%	87.01
20	75.00%	53.13	65.00%	35.44	0.00%	89.98
30	70.00%	59.00	50.00%	30.96	0.00%	92.58
40	35.00%	46.26	35.00%	43.27	0.00%	93.10
50	10.00%	66.80	10.00%	61.80	0.00%	109.10

Different delay values are thus obtained depending on the device used: while the TP-Link has the longest handover time, the other devices show significantly lower values. However, all the handovers are fast, ranging between 22 and 224 milliseconds. In many cases, this will be totally transparent for the user (a seamless handover).

5.1.2.2 Channel Assignment of the APs, including their associated STAs

The cooperative algorithms running in the controller use the monitoring information to make decisions about the channel assigned to each of the Wi-5 APs, based on *Internal* and *External Interference Impact*. Once this decision is made, the controller instructs the AP to move to a certain channel. This has to be done without disrupting the ongoing communications of the STAs. We have resorted to the standard 802.11 method for doing this: the use of beacons including the Channel Switch Announcement (CSA) element. Note that this element is also used for the horizontal handover.

The scheme of the process has been shown earlier in Figure 17: at the beginning, the AP and the STAs are in channel 1: frames and layer-2 ACKs are exchanged normally between them. At the same time, the AP sends monitoring information to the controller.

Periodically, the controller runs a channel assignment algorithm, and makes a decision which may imply the modification of the channel of some of the Wi-5 APs. To implement this decision, the controller sends a `switch_AP_channel` command to the AP.

When the AP receives this command, it has to instruct the STAs to switch to a new channel so it includes the CSA element in the beacons. A sort of "countdown" is implemented, meaning *"in 3 beacons, the AP will switch to channel 6."* When the countdown reaches 0, the STAs switch their channel and wait

[28] Wi-5 *HandoverMultichannel* application, see https://github.com/Wi5/odin-wi5-controller/blob/Development/src/main/java/net/floodlightcontroller/odin/applications/HandoverMultichannel.java

until they hear the beacons from the AP in channel 6. Therefore, some delay appears. In order to measure this delay, we have run some tests with our Wi-5 implementation, employing the *ChannelAssignment* application[29] in the controller.

The delay scheme[30] is shown in Figure 46: after the initial wait for agents' connections (`TimeToStart`), the application orders a scan, waits `ScanningInterval + AddedTime` seconds and gets the *distance* in dBm, doing that for each agent. It repeats the loop after `PauseBetweenScans` for `NumberOfScans` times, each iteration saving the cumulative moving average. Note: A detailed explanation of the meaning of each parameter has been provided in section 4.2.

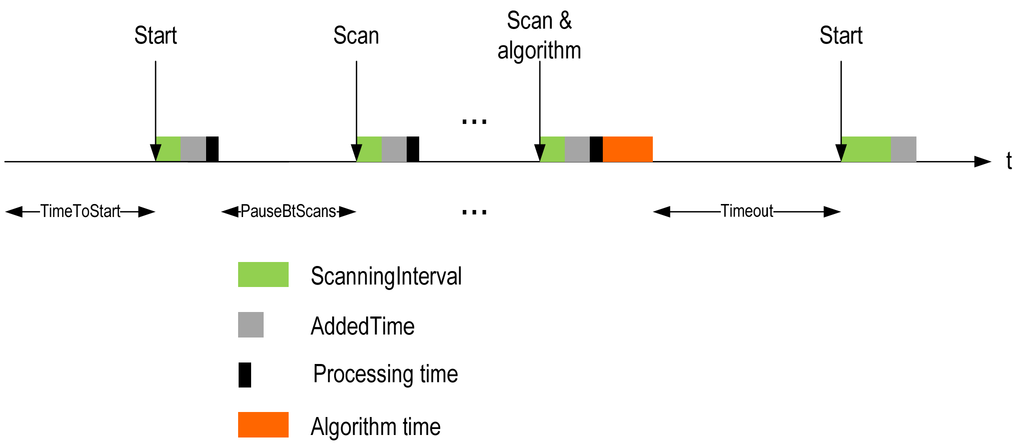


Figure 46: Meaning of the parameters of the *ChannelAssignment* application

Using this average, it runs the channel assignment algorithm and sends the orders to the APs in order to set the new values of the channel according to the output of the algorithm. This assignment is in the order of milliseconds. Then, it waits `Timeout` until it restarts the process.

This will result on the algorithm being run every:

$$\mathrm{Timeout} + \mathrm{AlgorithmTime} +$$

$$+ \mathrm{NumberOfScans} * [\mathrm{NumberOfAgents} * (\mathrm{ScanningInterval} + \mathrm{AddedTime} + \mathrm{ProcessingTime})]$$

$$+ (\mathrm{NumberOfScans} - 1) * \mathrm{PauseBetweenScans}$$

`AlgorithmTime` is the time required for running the Matlab code, which in our setup (Intel Core i3) varies between 1.5 and 8 seconds. `ProcessingTime` is the time required after each scanning. In our setup it is about 0.5 seconds.

Taking into account that the periodicity of the channel assignment algorithm will be in the order of hours (*e.g.,* every 2 hours, every 24 hours), these delays are not significant. As an example, the line `CHANNEL 30 30 2 1 3 7200 6 1` in `poolfile`, with 3 agents, will result in the channel assignment algorithm being run every `7299.5` seconds (roughly 2 hours).

The parameters `Timeout` and `PauseBetweenScans` have been introduced in order to let the manager decide a period for performing the channel assignment. As this is not a time critical application, it does not

[29] This application is available at https://github.com/Wi5/odin-wi5/wiki/Applications-for-Channel-Assignment
[30] This scheme corresponds to the initial version of the channel assignment application, which is only based on the "distance in dBs" between the APs. See section 4.2.2.

have to be performed continuously, but with a periodicity. We think this could be in the order of hours, *i.e.,* revising the channel assignment every day or twice a day.

5.1.3 Delays of the applications combining monitoring and configuration changes

The Wi-5 controller includes a series of applications able to gather information about the wireless environment and to make decisions based on it. In this subsection we present some measurements aimed at evaluating the combined delays of all these operations. The selected application is *SmartAPSelection* (see section 4.3), an application that performs a continuous *scan - gather results - make assignment decision* loop.

It is important that this loop runs very fast so it can perform a form of load balancing, which can at the same time be in charge of assigning the STA to the best suited AP for it.

In Figure 47 we present the box plot of the average scan time in the controller when the APs are in three different channels (1, 6 and 11) and the controller runs in a virtual machine (running with VMWare) inside an Intel Nuc 5i3RYH, with an Intel Core 3 processor. Different numbers of STAs have been used (1, 2 and 3). As can be observed, the vast majority of the periodic scans can be done every 2.5 seconds, which is enough for users at walking speed. It is also observed that the scanning time is not significantly increased with the number of STAs. A total amount of 1098 scans were used for obtaining the graph.

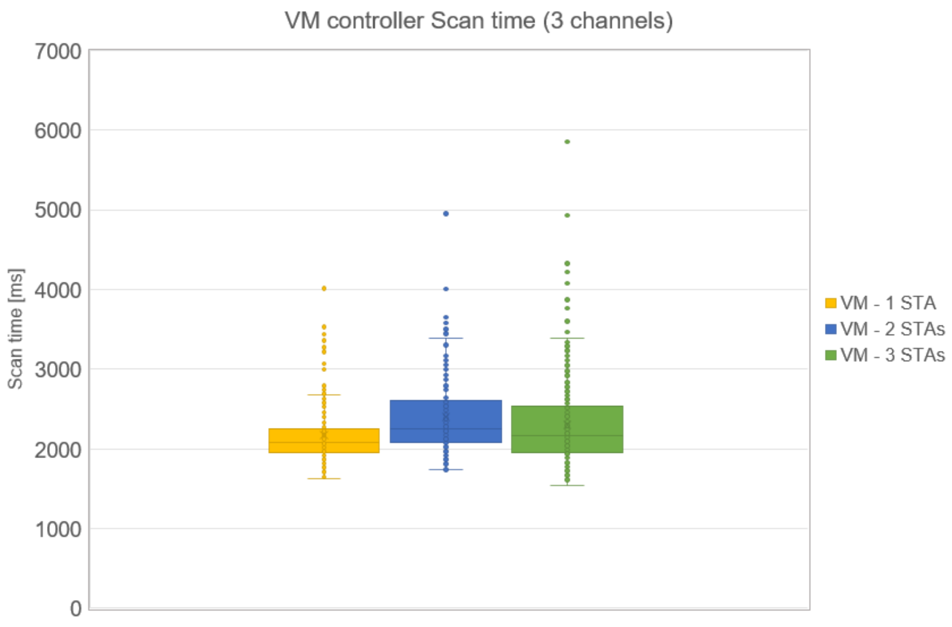


Figure 47: Delays of the scanning using a virtual machine for the controller

In Figure 48, the same tests have been run, but in this case the controller runs in a Raspberry Pi 3 Model B (Quad Core 1.2GHz, 1Gb RAM). Here, it can be observed that the period is reduced to 640 ms, and the variation is significantly reduced. We can conclude that, even when running in a low cost machine, the application can be performant for managing mobility of users at walking speed. A total number of 1041 scan periods have been averaged in this case.

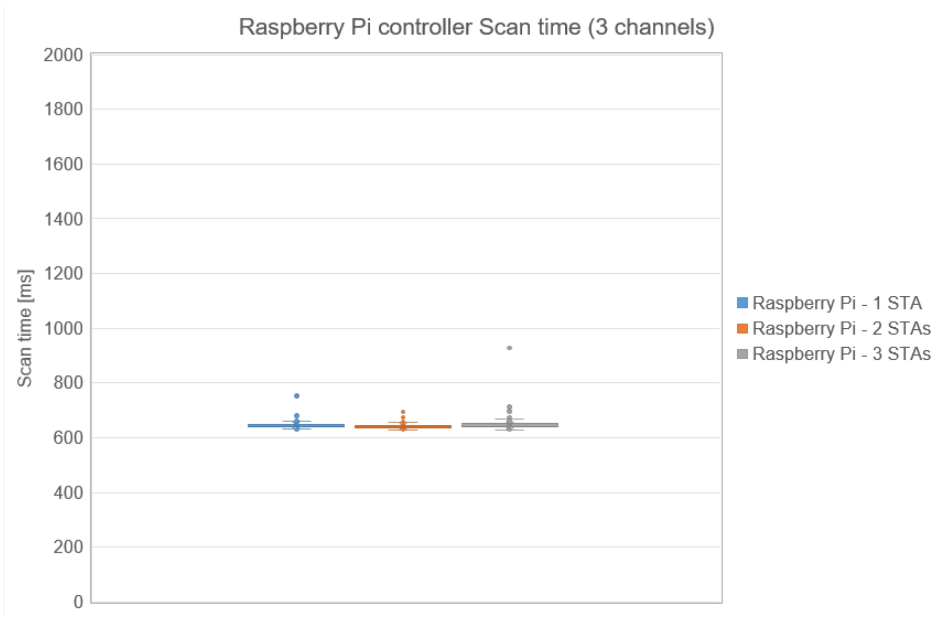


Figure 48: Delays of the scanning using a Raspberry Pi for the controller

6 Packet Grouping

In order to meet the growing traffic demand in Wi-Fi scenarios, a set of improvements are being included in the recent 802.11 amendments. One of them is frame aggregation, which was first introduced in 802.11n as a way to improve efficiency at the MAC level. In normal operation media access control is used before sending each frame but when frame aggregation is employed, a large frame can be assembled by concatenating multiple small ones: therefore, a number of sub-frames can be sent together, *i.e.,* the channel sensing and exponential backoff algorithm will only be run once. In addition, to the Layer2 mechanisms introduced in section 2.5, Layer 3 optimisation has also been explored: in certain scenarios, legacy 802.11 devices (prior to 802.11n) exist, so this optimisation is not possible at Layer 2.

In D3.2, a detailed study of Layer 3 optimisation was included, considering the scenarios where it can be useful (*e.g.,* community networks including a number of 802.11 hops) and the delay limits that appear depending on the applications. In addition, an analytical calculation of the achievable savings was presented, considering the optimisation in terms of packets per second and bandwidth. Finally, a series of tests were performed in the lab in order to measure the savings, and the modification of the traffic profiles were run and presented.

In D3.3, two studies were included: first, a detailed study of the achievable savings when using the aggregation functionalities of 802.11, including the development of testing software that can aggregate frames at Layer 2. Second, a study of the impact of aggregation on subjective quality. This test was run in cooperation with the University of Zagreb and included tests with real volunteers playing a real-time service (an online game) while the packets were aggregated. This study was also published in [41].

In this document, we include a detailed study of the throughput *vs*. latency trade-off in the context of centrally controlled solutions, and the proposal of three solutions that can be employed (separately or jointly), in order to achieve a good balance: *a)* prioritisation [42] for real-time services; *b)* setting a limit on the AMPDU size; *c)* employing real-time central coordination of frame aggregation in all the APs, depending on the traffic demand, the presence of real-time flows, etc.

Part of the results presented in this subsection have been published in an article titled *"Frame Aggregation in Central Controlled 802.11 WLANs: the Latency vs. Throughput Trade-off,"* which appeared in IEEE *Communications Letters*, November 2017 [43]. To the best of our knowledge, this was the first study proposing the centralized coordination and control of 802.11 frame aggregation functionalities.

6.1.1 Layer 2 aggregation mechanisms

Figure 49 presents a normal 802.11n frame structure including the PLCP (Physical Layer Convergence Protocol) headers, the MPDU, some tail bits and padding. An MPDU includes the MAC addresses, and the MSDU, which is the "payload" being carried.

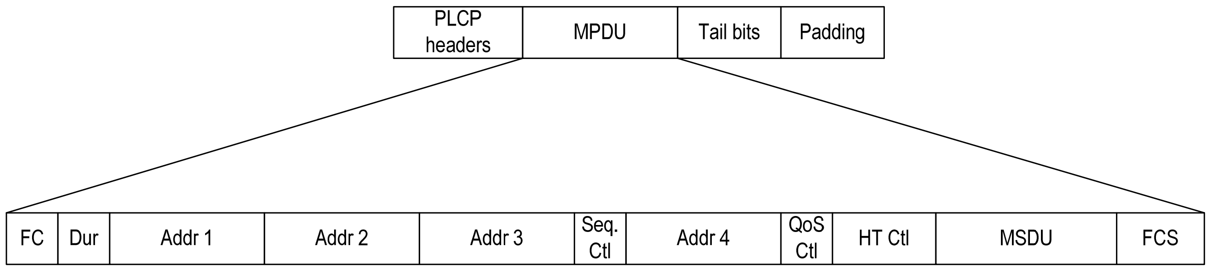


Figure 49: Structure of a normal 802.11 frame

In order to improve efficiency, new versions of Wi-Fi (from 802.11n) include mechanisms for frame aggregation: A-MPDU and A-MSDU (see Figure 50):

- A-MPDU: a number of MPDU delimiters each followed by an MPDU.
- A-MSDU: multiple payload frames share not just the same PHY, but also the same MAC header.

It should be noted that both aggregation mechanisms can also be combined (*Two Level Aggregation)*.

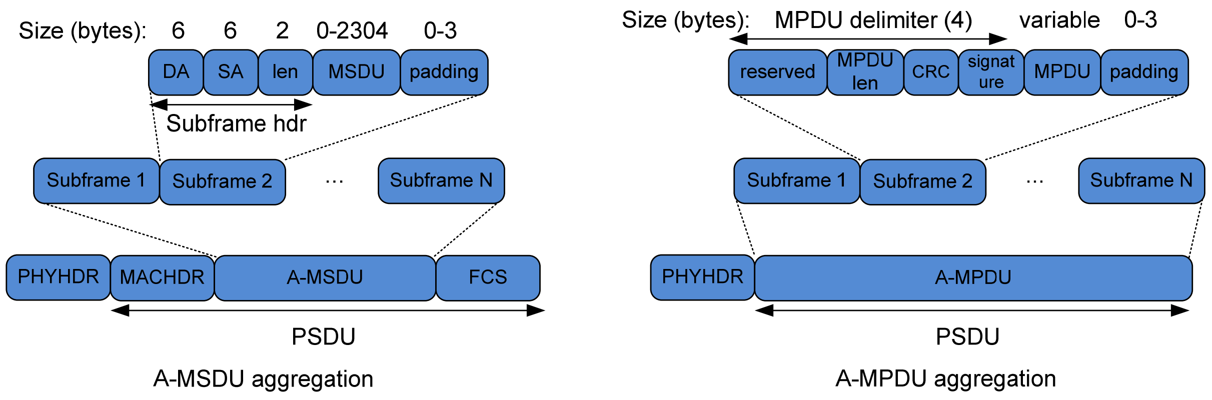


Figure 50: Frame aggregation schemes in 802.11n: a) A-MSDU; b) A-MPDU

Frame aggregation is becoming a common feature, in 802.11ac all the frames must have an A-MPDU format even when a single sub-frame is transported. This success of A-MPDU is because in lossy environments, when a number of sub-frames travel together and some of them are received in the wrong way (*i.e.,* the CRC does not match), only the wrong ones have to be sent again. That is not the case with A-MSDU, which requires the whole multi frame to be resent. Therefore, our study will mainly focus on A-MPDU, as it is the most used aggregation mechanism.

A frame including a number of A-MPDUs has the structure shown in Figure 51, with a number of sub-frames sharing the same PLCP headers. A *delimiter* and a *padding* have to be added for each of the MPDUs being carried. The delimiter includes the length of the MPDU that comes next.

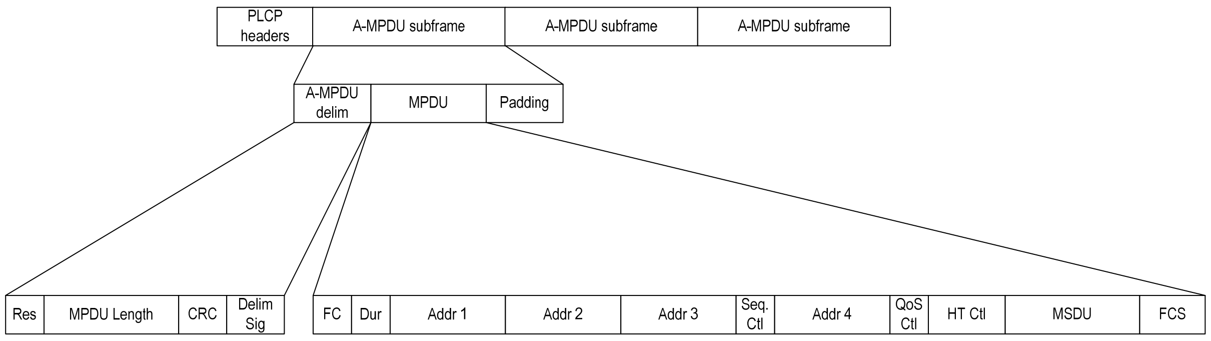


Figure 51: Structure of an A-MPDU 802.11 frame with many subframes

The efficiency increase achievable when aggregating frames is significant. Figure 52 shows an example: we have used the analytical model developed in [31] in order to represent the efficiency increase when these two aggregation mechanisms are used for packet sizes of 1500 bytes. It should be noted that A-MPDU allows a higher number of frames to be aggregated and the efficiency can be increased up to 90% for UDP packets, and 80% for TCP.

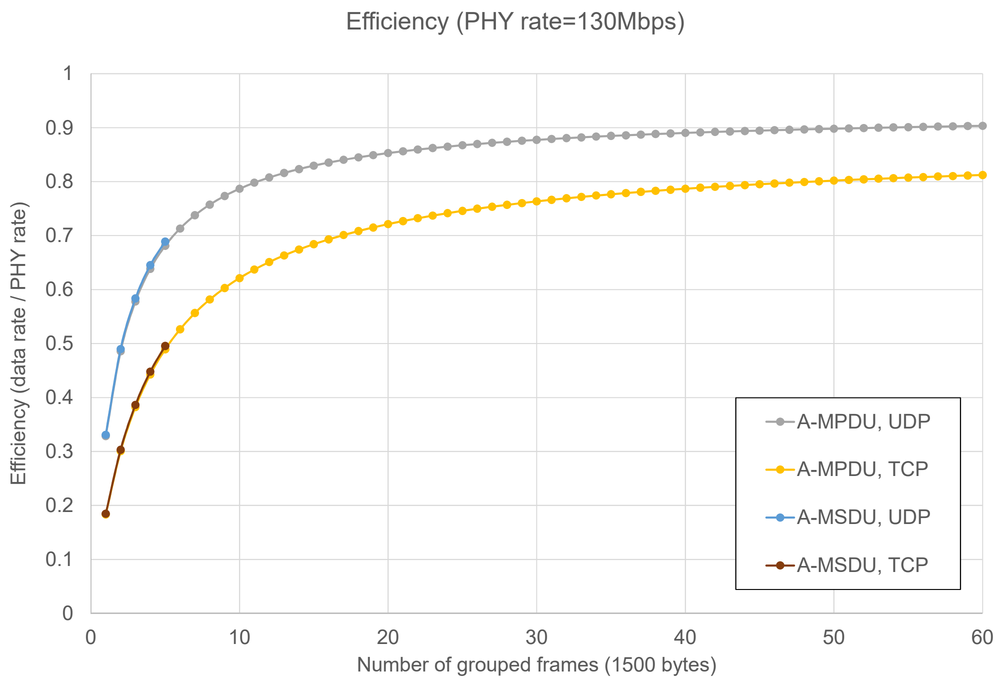


Figure 52: Efficiency improvement in 802.11 when using aggregation schemes

6.1.2 Frame Aggregation in Central Controlled 802.11 WLANs: the Latency vs. Throughput Trade-off

Frame aggregation is now widely used in 802.11 WLANs in order to provide a significant throughput improvement. However, the latency increase that comes as a result can reduce the quality experienced by the users of applications with real-time constraints. We will explore the throughput *vs.* latency trade-off in the context of centrally controlled solutions, as the SDWN-based ones developed within the Wi-5 project. First, a scenario with a single Access Point will be used to illustrate the problem, and to propose two possible solutions. Then, a centralised algorithm that dynamically activates or deactivates aggregation will be tested in a scenario with a number of APs. The results aim to show that aggregation parameters can be tuned in order to keep latency at low levels, with a low throughput penalty.

Services of very different natures may coexist in these scenarios: while some of the users are downloading a video, browsing the web or using an app, others may be running services with real-time constraints such as *e.g.,* VoIP, video conferencing or online games (many titles have been ported to smartphones or tablets, and some others have been designed for them). Therefore, a distinction between two kinds of services can be established: on the one hand, bandwidth-demanding services that want to get as much throughput as possible; on the other hand, low-latency services, which may use very little bandwidth.

However, as a counterpart of the throughput increase, when a STA gains access to the medium and transmits a long frame including a number of sub-frames, the rest of the STAs connected to the AP have to wait until the end of the transmission. This delay can be significant, especially if the STA is far from the AP and it is transmitting at a low rate. This results in delay and also in jitter, as some frames will wait more time than others. All in all, a trade-off appears: frame aggregation is positive for bandwidth-demanding services, but it can negatively affect services with real-time constraints.

6.1.3 Tests and Results

In this subsection we will first present the trade-off under study, and the results obtained when applying different solutions. All the results have been obtained with the ns-3[31] simulator. The simulation parameters are detailed in Table 6. The source code of the simulation script has been publicly shared in the Wi-5 repository in GitHub[32].

Table 6. Parameters of the ns3 simulations of frame aggregation

ns3 version	ns-3.26	**UDP packet size**	60 bytes
Mobility model	Random Waypoint	**UDP rate**	50 pps
Walking speed	1.5 m/s with pause time 2 s	**TCP packet size**	1,500 bytes
Channel model	YANS	**TCP variant**	New Reno
Channels	36 to 128 (20 MHz channels)	**Simulation time**	300 s (600s for Figure 56)
WiFi phy standard	80211n_5GHZ	**RTS/CTS**	When enabled, it is used for all packet sizes
WiFi rate control model	*Idealwifi* Manager	**Inter-AP distance**	100 m (30 m to the border)
Propagation model	Friis loss model Constant speed	**Short guard**	Not enabled

The tests are repeated a number of times (average 15) in order to obtain the 95% confidence intervals, which are shown in the figures below.

6.1.3.1 The Throughput vs. Latency Trade-off

The trade-off of interest is studied in a scenario starting with a single AP that is shared by 2N users where $N = 5$. N are using VoIP (simulated with UDP upload flows) and N are downloading a file with TCP (see Figure 53). The users move at walking speed, following a Random Waypoint Model in a

[31] ns3 network simulator, https://www.nsnam.org/, [Accessed Oct 2017].

[32] The ns3 source code developed for the simulations, and the scripts for obtaining each graph, are available at https://github.com/Wi5/ns3_802.11_frame_aggregation

60x60 meters square and we measured throughput, delay, and loss in each case. Each test has been run twice: with and without RTS/CTS activated. When relevant, both results will be presented.

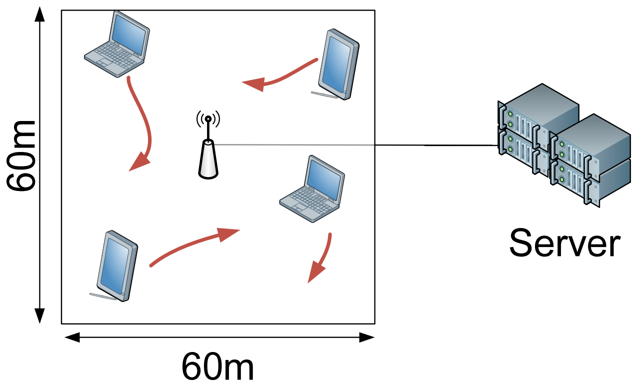
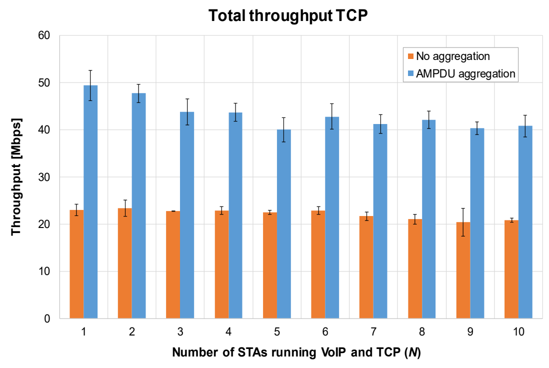


Figure 53: Test Scenario with 1 AP

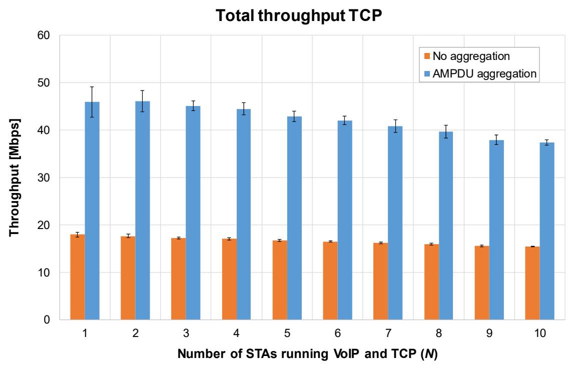
The results are shown in Figure 54. It can be observed that the throughput obtained by the TCP users is much higher if aggregation is employed (a1 and a2), as could be expected [31]. However, as a counterpart, a meaningful increase of the one-way delay is also observed, which can rise up to 25 ms when aggregation is employed (b1). This delay is caused by the waiting of the VoIP packets while the TCP AMPDUs are transmitted. This can be considered significant if we take into account the requirements of next generation networks, which *"will need to be able to support a roundtrip latency of about 1 ms"* [44]. Obviously, adding a significant amount of latency in the first hop of the communication is not desirable. The use of RTS/CTS can reduce this delay (b2), but this is still significant (about 10 ms).

The packet loss rate for VoIP is below 0.15% (see c1 and c2).

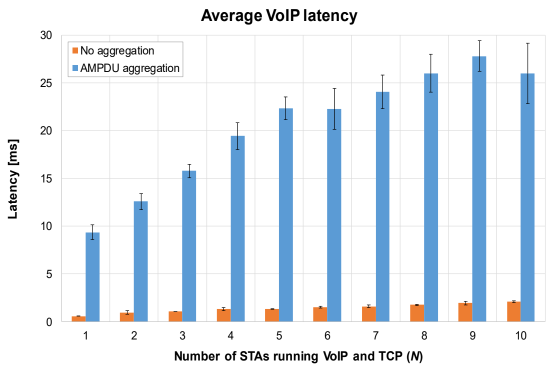
The cause of the delay increase when using aggregation is illustrated in Figure 55, in which the histogram of the latency experienced by VoIP packets is shown for the RTS/CTS disabled case. It can be observed that, when aggregation is not employed, the delay is really small (1.5 ms in average), and very constant (a), as the packets only have to wait until the end of single frames. However, if VoIP packets have to wait till the end of aggregated frames (AMPDUs), the delay increases to 22 ms, and so does its variability (b).



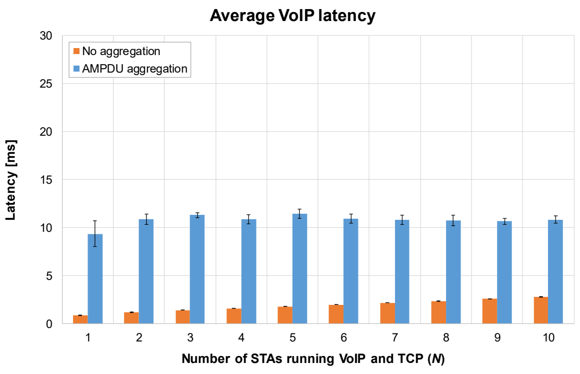
(a1) RTS/CTS not enabled



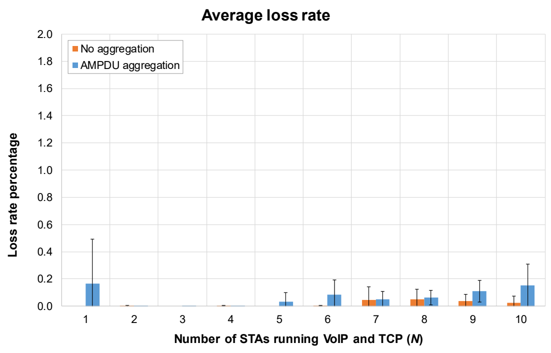
(a2) RTS/CTS enabled



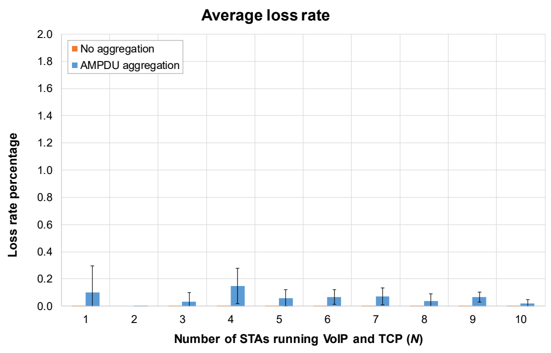
(b1) RTS/CTS not enabled



(b2) RTS/CTS enabled

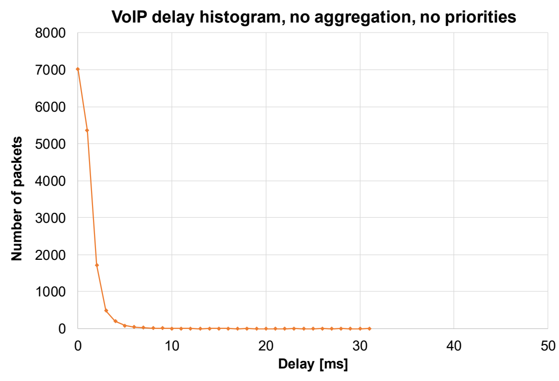


(c1) RTS/CTS not enabled

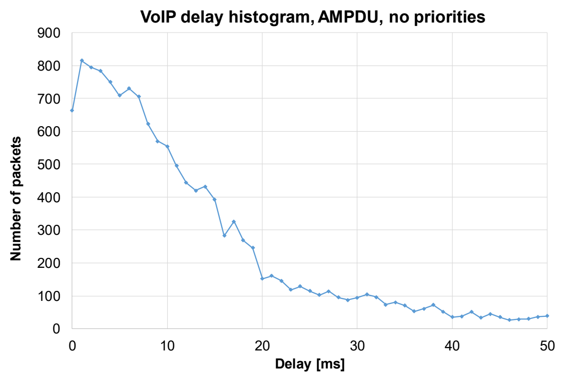


(c2) RTS/CTS enabled

Figure 54: A single AP shared by 2N of users. a) total throughput of the *N* TCP users; b) average one-way latency experienced by the N VoIP users; c) packet loss rate



(a)



(b)

Figure 55: A single AP shared by 10 users. Delay histogram of VoIP packets: a) No aggregation; b) AMPDU aggregation

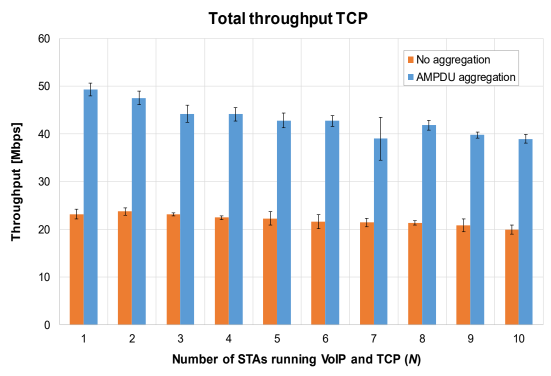
6.1.3.2 EDCA prioritisation

EDCA (Enhanced Distributed Channel Access) prioritisation, introduced by IEEE 802.11e [42], is based on the use of four queues with different values of the contention window. Data packets are divided according to four categories, namely *voice*, *video*, *best-effort* and *background*, each of them with different values for the *interframe space*, CW_{min}, and CW_{max} values. Only frames belonging to the same category can be aggregated together. The mechanism relies on the *Differentiated Service CodePoint* (DSCP) value in the Differentiated Service (DS) field of the IP header. Therefore, one limitation of this solution is that it will only work if the required value of the DS field is set, which not always happens in real networks.

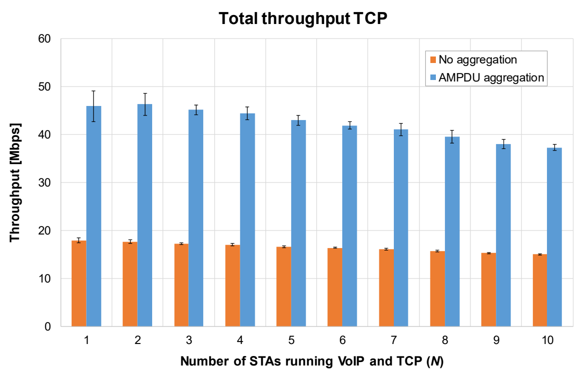
In our simulation scenario, we have assigned *Voice* (VO) priority to the VoIP flows and *Best Effort* (BE) to the TCP ones, obtaining the results shown in Figure 56. It can be observed that the results in terms of throughput of the TCP flows are quite similar to the previous ones (see Figure 54 a).

At the same time, the latency gets reduced in both cases as the VoIP packets have a higher priority: if aggregation is not enabled the latency is again low; if it is enabled, the latency is smaller than before, especially when the number of flows grows. However, it still results in significant delay (8-9 ms) in some cases. It can also be observed that the use of RTS/CTS in this case is less relevant, and can even increase latency slightly for VoIP flows, as the packets have to wait some additional time during the RTS/CTS process. In fact, the results with RTS/CTS are quite similar to those obtained in Figure 54: the use of priorities does not provide a big benefit when combined with RTS/CTS.

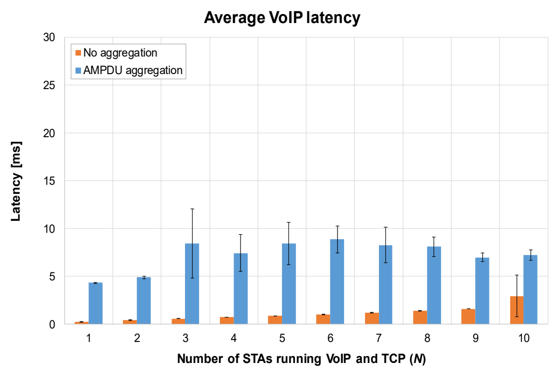
The packet loss rate for VoIP is again very low: it is below 0.1% in all cases.



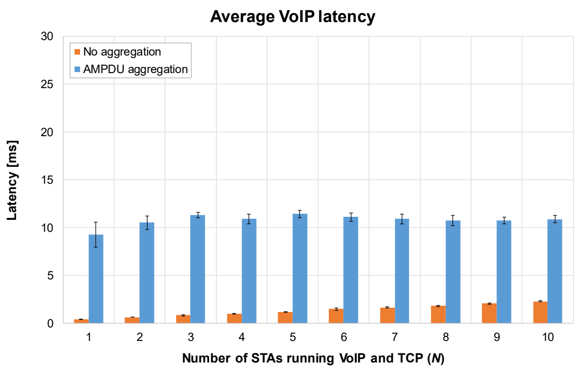
(a1) RTS/CTS not enabled



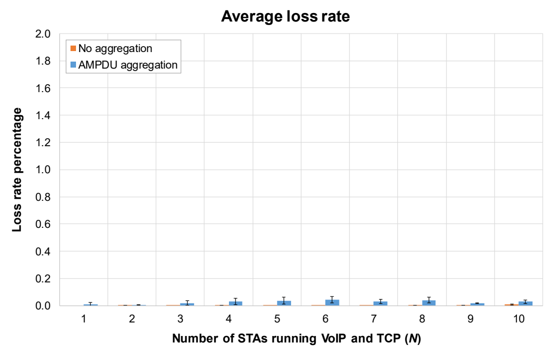
(a2) RTS/CTS enabled



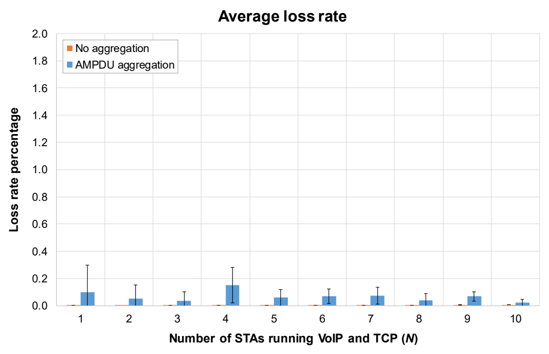
(b1) RTS/CTS not enabled



(b2) RTS/CTS enabled



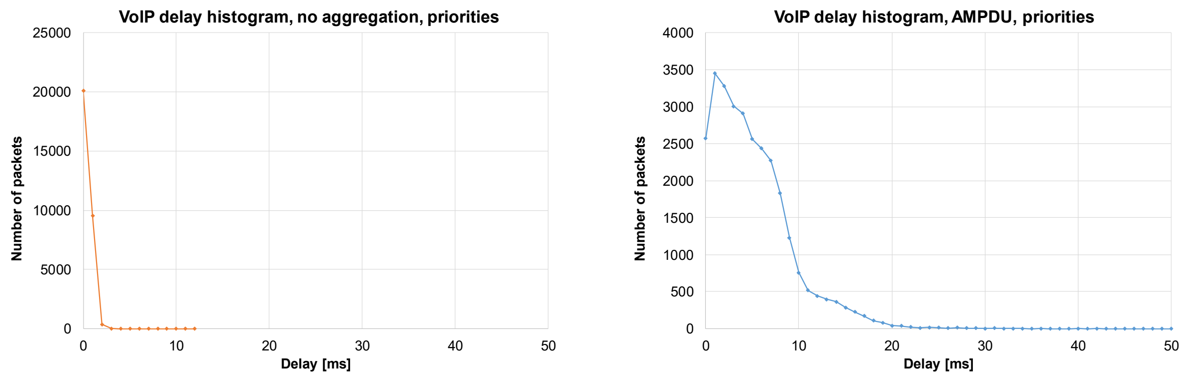
(c1) RTS/CTS not enabled



(c2) RTS/CTS enabled

Figure 56: A single AP shared by a number of users, using EDCA priorities: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) packet loss rate

Again, we review the histogram of the delay of the VoIP packets (RTS/CTS disabled) in Figure 57. A significant delay increase can still be observed when AMPDU aggregation is employed. In addition, if we compare the results with Figure 55, it can be seen that EDCA priorities can in this case reduce the delay and its variability.



(a) (b)

Figure 57: A single AP shared by 10 users, using EDCA priorities. Delay histogram of VoIP packets: a) No aggregation; b) AMPDU aggregation

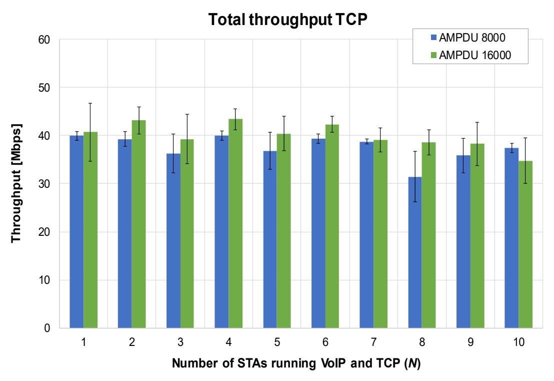
6.1.3.3 Setting a limit on the AMPDU size

Another possibility for reducing VoIP latency is to set a limit for the maximum AMPDU size, instead of using the default limit of 65,535 bytes as defined in 802.11n. In our case, two different values have been used: 8 kB, which allows AMPDUs to include 5 packets of 1,500 bytes; and 16 kB, which allows the inclusion of 10 packets into a single AMPDU. The results are shown in Figure 58.

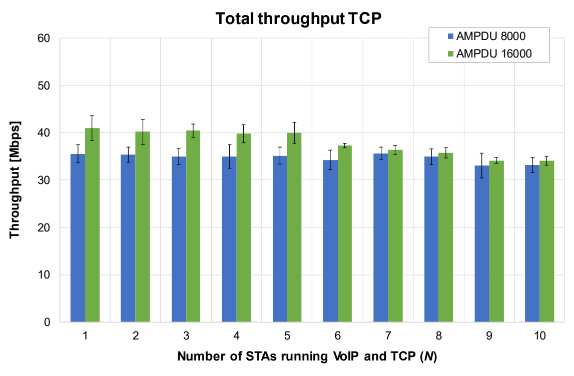
If we compare Figure 58 b with Figure 54 b, a significant reduction of the latency experienced by the VoIP applications can be observed, which is now always below 7 ms. The counterpart is a reduction of the throughput, which is between 80 and 95% of the amount obtained with the default limit (compare Figure 58 a and Figure 54 a in terms of *AMPDU Aggregation* bars). It can therefore be said that the higher the maximum size of the AMPDU, the better the throughput and the worse the latency.

If we enable RTS/CTS, the results are quite similar in terms of throughput and delay. However, it can be observed that the average loss rate is better with a bigger AMPDU size, especially when the number of users is high, so the hidden node problem can be more common in this scenario. However, if the number of users is small, the use of RTS/CTS deems a worse behaviour as it is introducing more overhead. Regardless, it can be observed that the packet loss rate for VoIP increases slightly, but it is still moderate: below 0.4% in all cases.

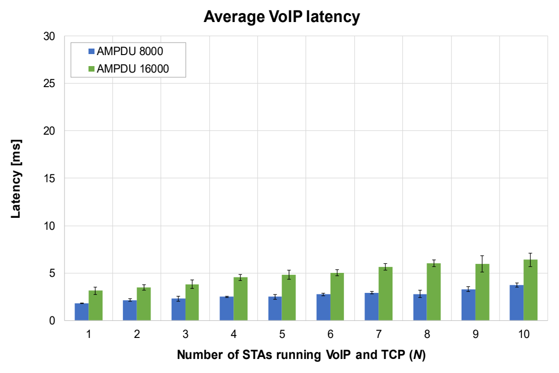
We finally present the histogram of the delay of the VoIP packets in Figure 59, with *N*=5 (total 10 users). If we compare it with Figure 55 we observe that the histogram can be widened or narrowed by the use of the AMPDU limit parameter: Figure 59a shows a very narrow histogram (almost similar to the non-aggregation case (Figure 55a). If we increase the limit (Figure 59b), the histogram looks similar to the one shown in Figure 55b, where the default limit of 65,535 bytes was used.



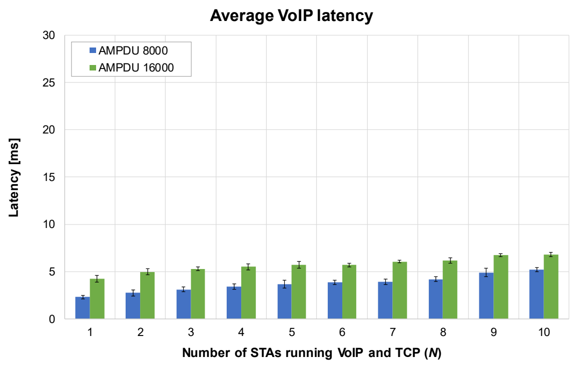
(a1) RTS/CTS not enabled



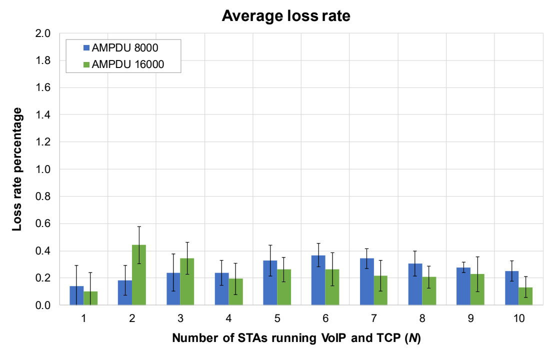
(a2) RTS/CTS enabled



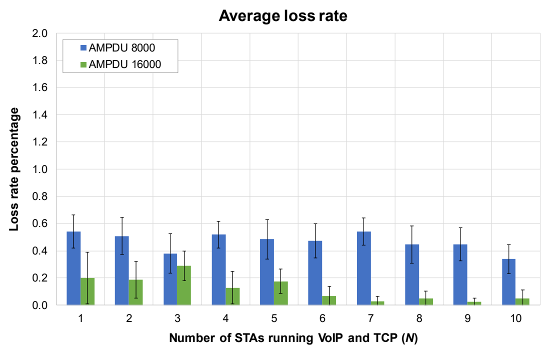
(b1) RTS/CTS not enabled



(b2) RTS/CTS enabled

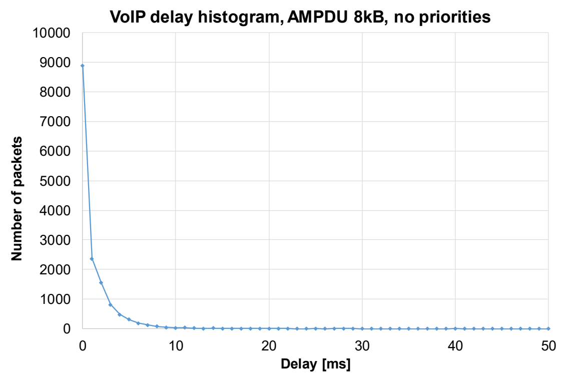


(c1) RTS/CTS not enabled

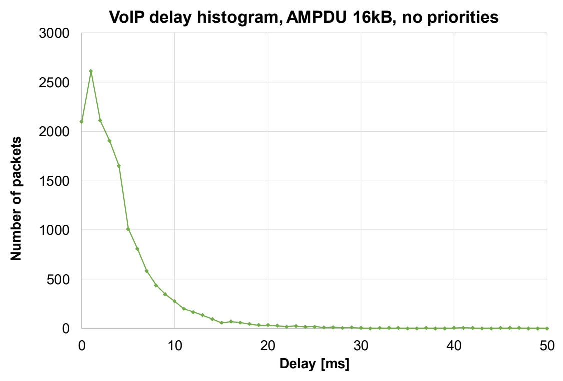


(c2) RTS/CTS enabled

Figure 58: A single AP shared by a number of users, using a limit of 8 kB and 16 kB for the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users ; c) packet loss rate



(a)



(b)

Figure 59: A single AP shared by a 10 of users. Delay histogram of VoIP packets using a limit of a) 8 kB and b) 16 kB for the size of the AMPDU.

All in all, the maximum AMPDU size parameter can be tuned in order to modify the trade-off: if a higher delay can be tolerated by the applications, we can obtain a better throughput. However, the difference in throughput is smaller than the difference in latency, *i.e.,* with a throughput reduction between 10-20%, the latency can be reduced by a factor between 4 and 8. The cause of this is the asymptotic behaviour of the efficiency increase when aggregating (see *e.g.,* Figure 52, and the results presented in [31]): there is a moment where the increase in the number of aggregated frames only provides a small throughput improvement, but the added delay still grows linearly.

Finally, it should be remarked that this solution (modifying the maximum AMPDU size) presents an advantage with respect to prioritisation: it is easier to deploy, *i.e.,* it just requires an adjustment of the AMPDU maximum size parameter in the APs.

6.1.3.4 Central coordination

The solutions already presented (prioritisation and setting a limit on the AMPDU size) have been illustrated using a scenario with a single AP. However, in order to test the central coordination of aggregation, a bigger scenario is required. We will next present the results in three different scenarios, with 4, 9 and 16 APs, which have been implemented in our ns3 script (see Figure 60). As in [14], [23], we assume that the APs are coordinated by a central controller (*e.g.,* based on SDWN) able to tune different network parameters.

As before, *N=5* and *2N* users will walk through the scenario: *N* are using VoIP and *N* are downloading a file with TCP. Each AP uses a different channel, and a fast handover mechanism has been included in the simulation environment: as soon as a STA gets disconnected, it looks for the closest AP and requests association. Note that no other Wi-5 functionality is running in the simulation scenario.

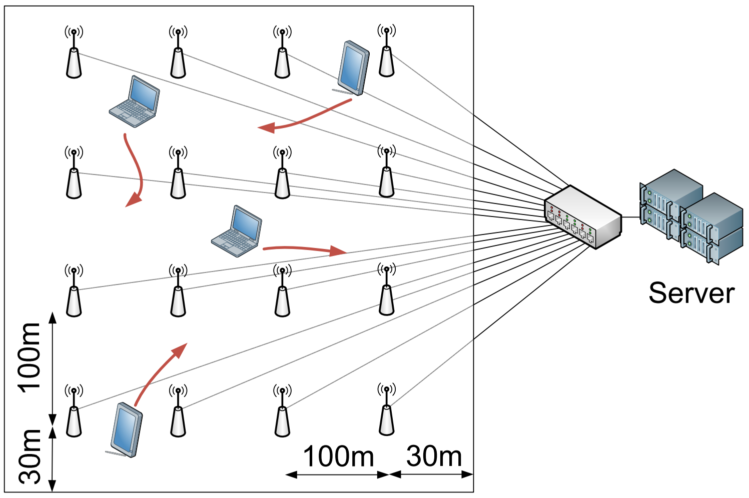


Figure 60: Test Scenario with 16 APs

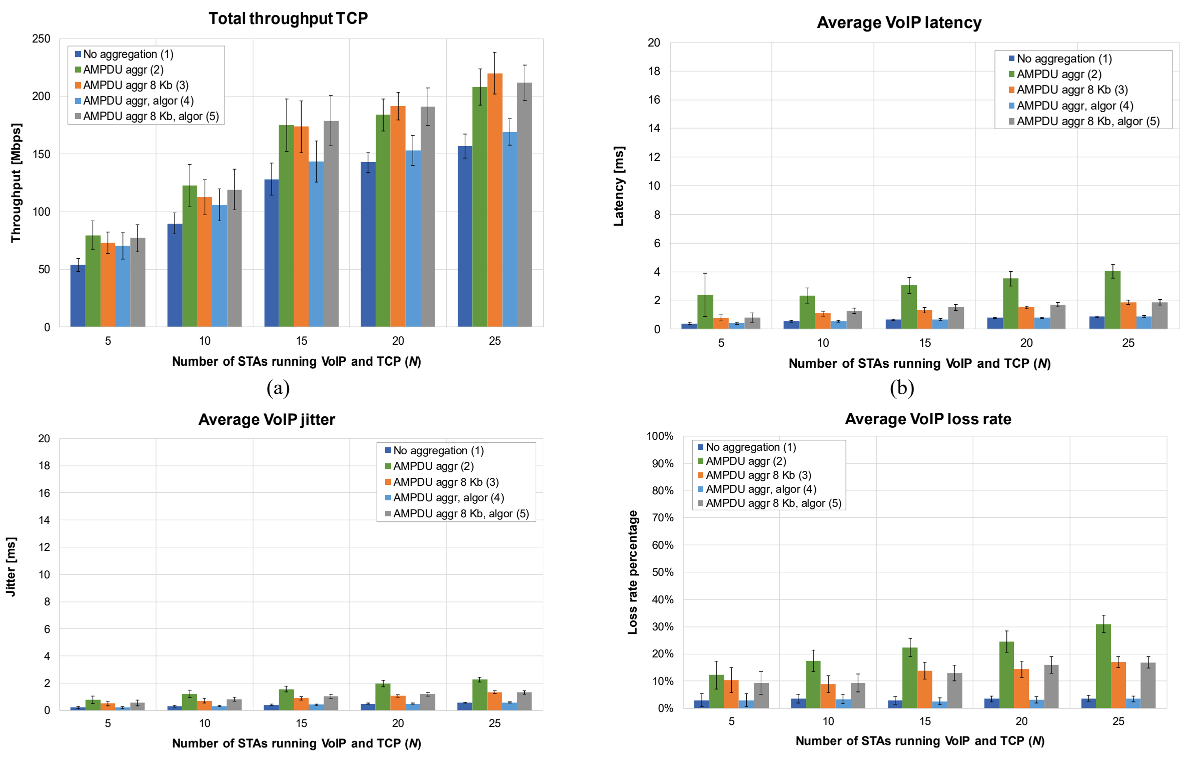
In this case, the central controller runs an algorithm that dynamically disables aggregation in an AP if a VoIP flow appears, and enables it when the VoIP user leaves again. Therefore, VoIP users will always *see* a non-aggregating AP, whereas TCP users will receive non-aggregated frames in some moments, thus experiencing some throughput penalty.

It should be noted that some knowledge about the nature of the flows present in the network is required in order to run this algorithm. Although this does not represent a problem in the simulations (all the information is available), in a real network it would require the use of a traffic detection mechanism. This is feasible as solutions based on machine learning can detect the presence of real-time flows just using the packet size and inter-packet time [45], thus maintaining user's privacy.

6.1.3.4.1 Scenario with 16 APs

The results obtained in this scenario, when EDCA priorities are used, are shown in Figure 61, where five different cases are compared:

1. No aggregation.
2. AMPDU aggregation is always active (with the default limit of 65 kB).
3. Aggregation with a fixed limit of 8 kB.
4. Aggregation, with the algorithm deactivating it when a VoIP STA is associated to an AP.
5. The algorithm is used but, instead of deactivating the aggregation, a maximum size of 8 kB is set when a VoIP flow is present in an AP, in order to limit the added delay.



(a) (b)

(c) (d)

Figure 61: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) Total throughput of the TCP users; b) Average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS not used

It can be observed that *No aggregation* (1) is always the worst option in terms of throughput, but it maintains a low value of latency and packet loss. Regarding AMPDU aggregation (2), the throughput is improved but at the cost of an increased latency and packet loss rate. If the AMPDU limit is set to 8 kB (3), the latency gets reduced but, although the throughput gets reduced when the number of users is low, it is improved for 20 and 25 users. This can be caused by the difference in terms of packet loss rate.

If the algorithm controlling aggregation is running (4), a throughput penalty between 13 and 19% with respect to (2) is seen, but this corresponds into very low values of latency (a reduction factor between 4 and 7) and packet loss (factor between 4 and 9). Finally, the combination of the algorithm and the limit of 8 kB (5) gives very similar results to (3): the throughput penalty is minimal, and the VoIP users will experience a similar delay and packet loss rate.

If EDCA priorities are not enabled, we obtain the results shown in Figure 62, where a significant increase in VoIP latency and jitter are observed, as the packets have to wait for a longer time in the queues. In addition, as RTS/CTS is not used, the packet loss rate becomes very high, even unacceptable.

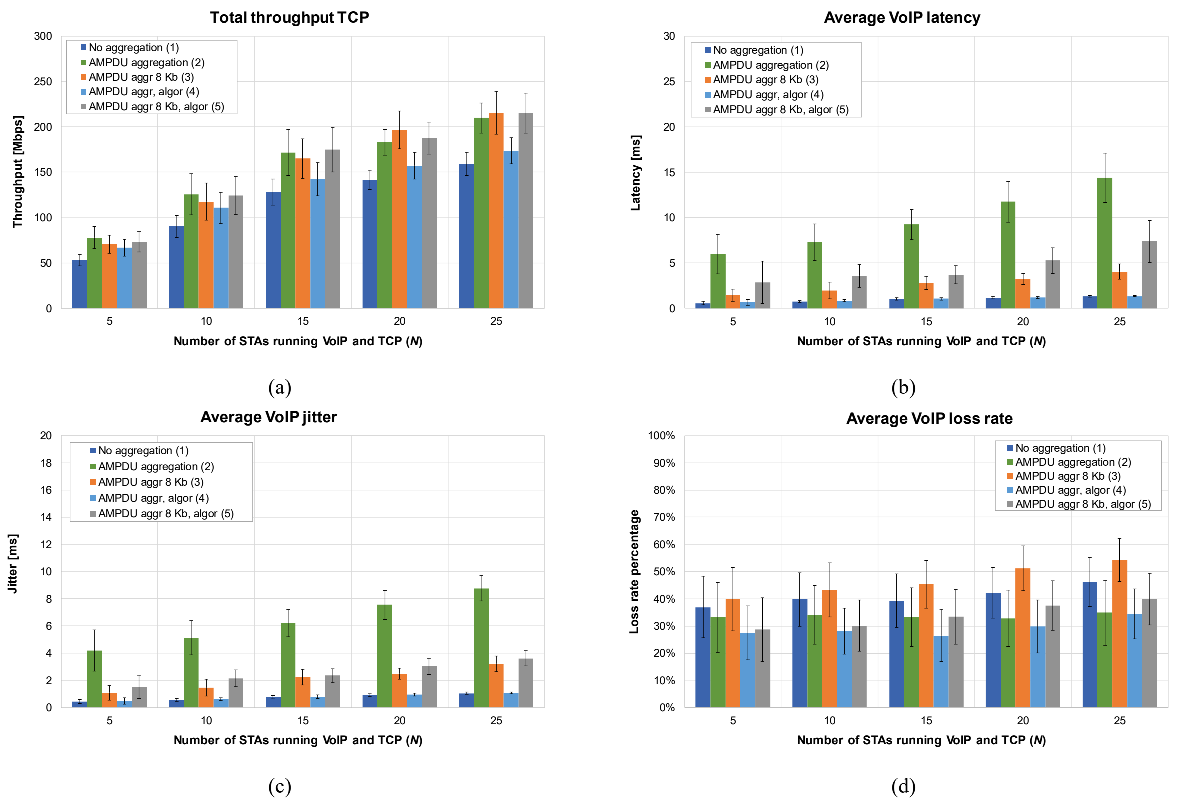


Figure 62: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. No EDCA priorities are used. RTS/CTS not used

However, if EDCA priorities are not enabled, but RTS/CTS is enabled, we obtain the results shown in Figure 63. The throughput is not significantly modified, and the additional latency is again observed: for example, with 25 users, the rise is significant in all cases: as an example, latency for (2) grows from 14 to 21 ms.

The jitter also shows a moderate increase, but the real difference appears in the packet loss graphs: if RTS/CTS is employed, packet loss drops to acceptable levels, although it can be high if the number of users is 20 or 25.

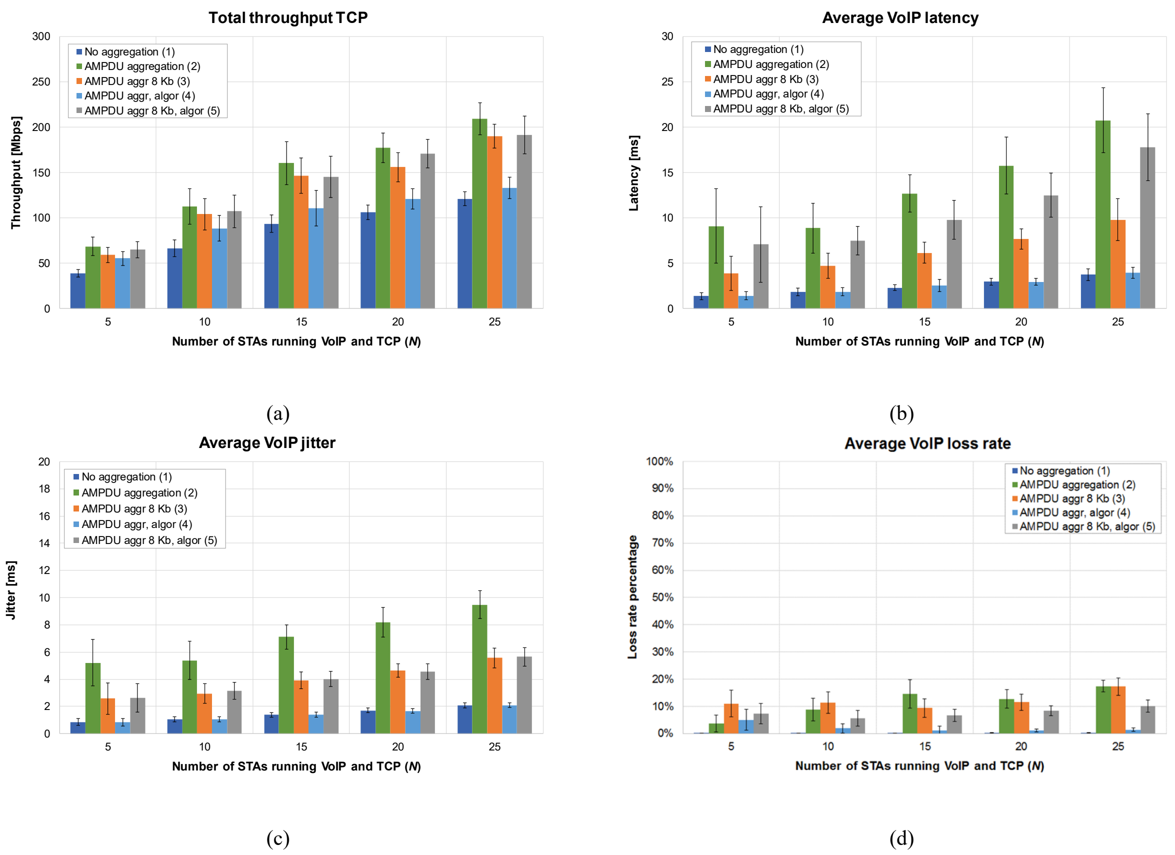


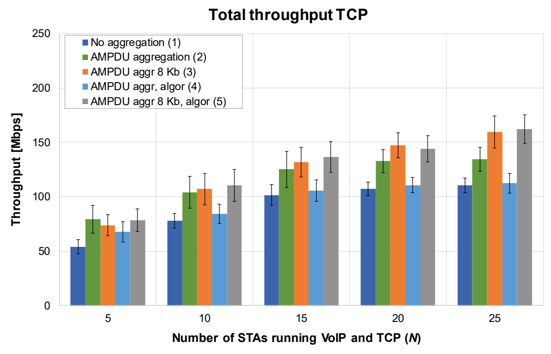
Figure 63: A network including 16 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. No EDCA priorities are used. RTS/CTS is activated

One conclusion is that, in absence of EDCA priorities, the use of RTS/CTS is a must in order to grant an acceptable quality level to VoIP users: packet loss rate is reduced, at the cost of a latency increase.

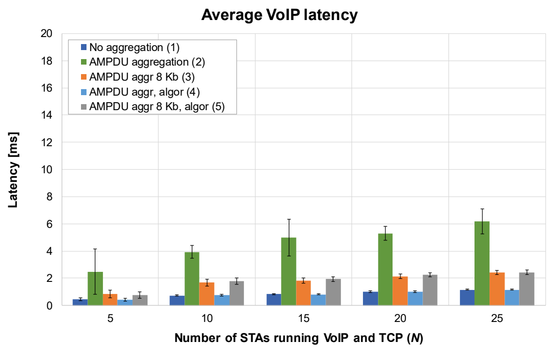
6.1.3.4.2 Scenario with other numbers of APs

In this subsection we present some tests aimed at capturing the effect of the number of users (density). For that aim, we have reduced the number of APs and maintained the number of users. Results with smaller scenarios including 9 and 4 APs respectively are presented and analysed.

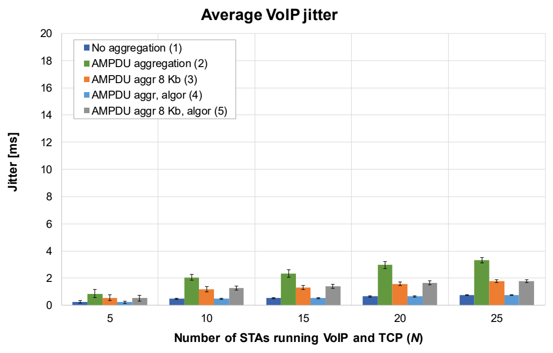
First, the results obtained with 9 APs are presented in Figure 64 (EDCA priorities are enabled). When compared with Figure 61, we observe that the total throughput is lower: which is logical, taking into account that the number of APs has been reduced. In addition, the average delay when using AMPDU aggregation is slightly higher, and the same happens with the jitter and packet loss rates.



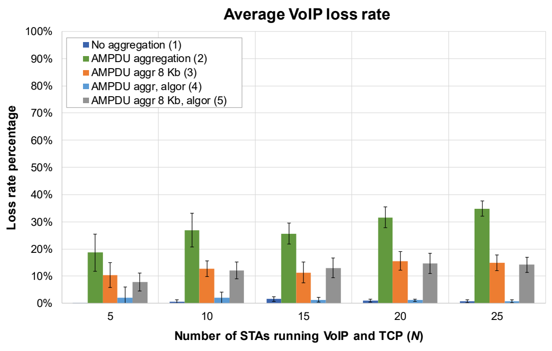
(a)



(b)



(c)



(d)

Figure 64: A network including 9 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS is disabled

If we further reduce the number of APs to 4, we obtain the results shown in Figure 65. In this case, we observe a slightly different behaviour regarding the throughput: when the number of users is high (*e.g.,* 25 users), AMPDU aggregation can even provide worse results than *No Aggregation,* in terms of throughput, latency and packet loss. This is caused by the saturation and the effect of hidden terminals. In addition, the best behaviour is obtained when the maximum AMPDU size is set to 8 kB, with or without the central algorithm activated.

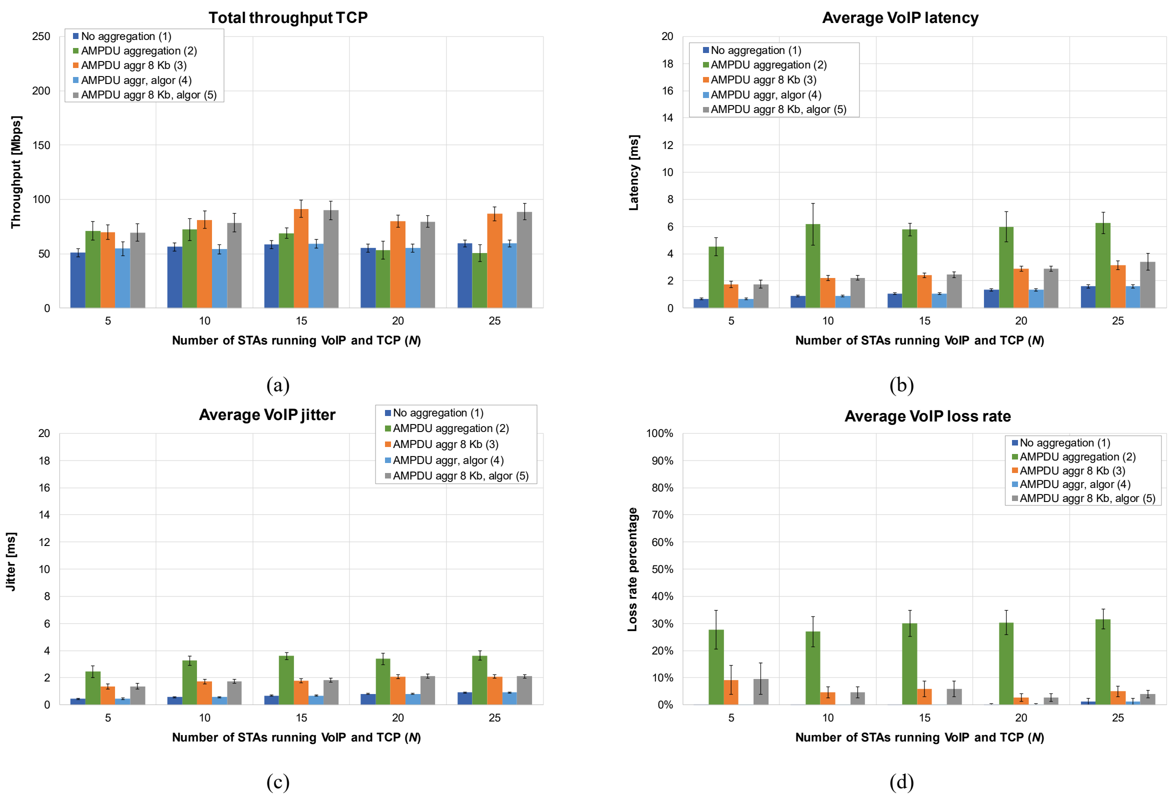


Figure 65: A network including 4 APs shared by a number of users, using different options for controlling the size of the AMPDU: a) total throughput of the TCP users; b) average one-way latency experienced by the VoIP users; c) average jitter of the VoIP users; d) average loss rate experienced by the VoIP users. EDCA priorities are used. RTS/CTS is disabled

***

All in all, the decision of which the five aggregation solutions is better on each case will first be determined by the availability of information about the services present on each AP. Nevertheless, the network manager can always tune the maximum AMPDU size, in order to limit the maximum delay and packet loss rate that the users will experience. A penalty in terms of throughput can be expected as a counterpart of the improvement of the quality of real-time services, but this trade-off can be tuned in order to get a good balance between the users of both services.

If the user density is higher, it can be observed that the benefit of aggregating is significantly reduced: as a consequence of saturation and hidden terminal problems, in some cases aggregation can provide a very similar overall throughput, but with an increased latency.

7 Conclusions

This document has presented the final version of the Smart Access Point Solutions developed within the WP3 of the Wi-5 Project. An updated Literature Review has surveyed the relevant research fields, such as the architectural proposals for SDWN; the use of virtual Wi-Fi APs combined with SDN; the frameworks providing fast handovers in these scenarios; the resource management in Wi-Fi WLANs and packet/frame grouping for improving network efficiency. A global view of the Wi-5 architecture has also then been provided based on D2.5, which outlines the different entities and the description of the functionalities being considered here and the coordination between the functionalities developed in WP3 (Smart AP Solutions) and WP4 (Cooperative Functionalities).

Then, the main section of this deliverable (section 4) is devoted to explaining the functionalities enabling all the Wi-5 features, with a detailed explanation of the following points:

- The framework being used for the implementation of the Wi-5 functionalities based on the use of Light Virtual APs (LVAPs). The controller creates an LVAP for each terminal, which is dynamically assigned to the physical AP where the terminal is located at each moment. Therefore, the AP will use a different LVAP (which includes a specific MAC) for communicating with each terminal. Thus, a terminal will only "see" a single AP, even if it is moving between different APs, thus avoiding the need for re-association.
- The framework includes a central controller which has all the information available, and is therefore able to make smart decisions about the assignment of clients to APs. This avoids the problem of the "sticky client," which remains connected to the original AP it associated with, rather than moving to a nearby one that would be a better choice for it.
- The horizontal handover scheme integrating multi-channel APs with the LVAPs approach. This includes extensive tests of the handover latency, illustrating that they can really be *seamless, i.e.,* not disrupting the ongoing connections. The handover mechanism is integrated with a set of active and passive monitoring tools and other functionalities, resulting in a solution that is able to provide smart functionalities using low-cost commercial APs.
- Two different test scenarios (a lab environment in UNIZAR and a test house in AirTies) have been used to compare proactive and reactive handover mechanisms in realistic conditions, and the advantages of the proactive approach have been highlighted. The inter-channel handover delay has been measured with three different devices, using five values for the inter-beacon time, proving that fast and seamless handovers are possible in the scenarios, even with low cost off-the-self equipment.
- The implementation of the different applications including Channel Assignment, Mobility Management (in a reactive and a proactive way) and Load Balancing based on RSSI, Fittingness Factor, which use monitoring information to understand the services being run by the users.

Section 5 is devoted to report the measurements for the overall delays incurred by the system. This includes the delays when collecting monitoring information; when dispatching configuration changes to the equipment; and of the applications combining both functionalities (monitoring and subsequent configuration). The results show that the delays incurred by the system are low enough so as not to pose significant limitations: the throughput of the traffic exchanged between the controller and the APs is very low; the delay required to dispatch configurations to the APs is low; and the applications can make decisions fast enough, even running in a low-cost single-board computer.

A simulation environment has been developed in order to test different ways of performing coordinated control of the frame aggregation mechanisms in 802.11n and 802.11ac. This is reported in section 6. It has allowed us to study the trade-off between the throughput improvement and the extra latency that come as a consequence of frame aggregation. Three solutions, based on prioritisation, limiting the maximum AMPDU size and using a central coordination of the aggregation have been studied and discussed. A trade-off appears: if the maximum delay of real-time services has to be reduced, a penalty in the throughput of TCP connections will appear.

Two innovative aspects can be highlighted: firstly, the development of a method able to proactively manage the mobility of the users, also combining this with load balancing in real time. The obtained results show that running these two functionalities at the same time is possible, even using a single-board computer as a controller. This avoids the so-called *"sticky client"* problem, which happens when a client remains connected to the original AP it associated with, rather than moving to a nearby one that would be a better choice for it. And second, the proposal of a central coordination of frame aggregation, which can provide a significant improvement in efficiency, while respecting the real-time requirements at the same time.

As future work, the improvement of the scalability of the system can be considered, by means of a hierarchy of controllers: a first layer of controllers close to the APs would be in charge of managing the handoffs with low delay, and other layers could perform coordination between a number of controllers. Another possibility is the integration with other technologies as *e.g.,* mobile networks, as required by 5G. Finally, better management of the QoS could be achieved by monitoring the services being managed by the APs, and tuning the priorities accordingly.

References

[1] K. Schneider, D. Turgut, M. Chatterjee, “An experimental study on Layer 2 roaming for 802.11 based WLANs,” 2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Espoo, Finland, 2007, pp. 1-6.

[2] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, “Performance Anomaly of 802.11b”, INFOCOM 2003.

[3] A. Mishra, M. Shin, W. A. Arbaugh, “An empirical analysis of the IEEE 802.11 MAC layer handoff process,” ACM SIGCOMM Computer Communication Review. 2003 Apr 1;33(2):93-102.

[4] R. Riggio, T. Rasheed, R. Narayanan, “Virtual network functions orchestration in enterprise WLANs,” in Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on , pp.1220-1225, 11-15 May 2015.

[5] E.Haleplidis, K. Pentikousis, S. Denazis, J.H. Salim, D. Meyer, O. Koufopavlou, (2015). Software-defined networking (SDN): Layers and architecture terminology (No. RFC 7426).

[6] Open Networking Foundation, “Sdn architecture,” vol. 1.0, Jun 2014. [Online]. Available: https://www.opennetworking.org/ images/stories/downloads/sdn-resources/technical-reports/TRSDN ARCH_1.0_06062014.pdf

[7] J. Schultz, R. Szczepanski, K. Haensge, M. Maruschke, N. Bayer and H. Einsiedler, “OpenGUFI: An Extensible Graphical User Flow Interface for an SDN-Enabled Wireless Testbed,” 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, 2015, pp. 770-776.

[8] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, D. Walker, “Frenetic: A network programming language,” SIGPLAN Not., vol. 46, no. 9, pp. 279–291, Sep. 2011.

[9] A. Voellmy, H. Kim, N. Feamster, “Procera: A language for highlevel reactive network control,” in Proc. of ACM HotSDN, 2012.

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.

[11] R. Riggio, K.M. Gomez, T. Rasheed, J. Schulz-Zander, S. Kuklinski, M.K. Marina, “Programming Software-Defined wireless networks,” in Network and Service Management (CNSM), 2014 10th International Conference on, pp.118-126, Nov 2014.

[12] Y. Grunenberger, F. Rousseau, “Virtual Access Points for Transparent Mobility in Wireless LANs,” In Wireless Communications and Networking Conference (WCNC), 2010 IEEE (pp. 1-6).

[13] 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.

[14] 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.

[15] A. Zubow, S. Zehl and A. Wolisz, "BIGAP — Seamless handover in high performance enterprise IEEE 802.11 networks," NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, 2016, pp. 445-453. doi: 10.1109/NOMS.2016.7502842

[16] IEEE 802.11n, IEEE Standard for Information technology- Local and metropolitan area networks- Specific requirements- Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput, IEEE Std 802.11n-2009, 2009.

[17] IEEE 802.11ac, IEEE Standard for Information technology-- Telecommunications and information exchange between systems-Local and metropolitan area networks-- Specific requirements--Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications--Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz

[18] IEEE 802.11h, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 5: Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe.

[19] H. Hwang and Y. T. Kim, “Enhanced fast BSS transition on enterprise WLAN with SDN-based distribution system,” 2017 13th International Conference on Network and Service Management (CNSM), Tokyo, 2017, pp. 1-5.

[20] J. P. Jeong, Y. D. Park, Y. J. Suh, “An Efficient Channel Scanning Scheme With Dual-Interfaces for Seamless Handoff in IEEE 802.11 WLANs,” in IEEE Communications Letters, vol. 22, no. 1, pp. 169-172, Jan. 2018.

[21] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, C. Peylo, “Cloudmac: Towards software defined wlans,” SIGMOBILE Mob. Comput. Commun. Rev., vol. 16, no. 4, pp. 42–45, Feb. 2013.

[22] Anyfi Networks, “Software-defined wireless networking: Concepts, principles and motivations,” April 2015. [Online]. Available: http://www.anyfinetworks.com/get_resource/anyfi-sdwn-concepts-whitepaper

[23] 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, vol.21, no.11, pp. 374-377, Feb. 2017, doi 10.1109/LCOMM.2016.2623602.

[24] 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

[25] I. Papanikos, M. Logothetis, “A study on dynamic load balance for IEEE 802.11 b wireless LAN,” in Proc. 8th International Conference on Advances in Communication and Control (COMCON), Crete, 25-29 June 2001.

[26] Cisco Systems Inc., “Data sheet for Cisco Aironet 1200 series,” 2004, http://www.cisco.com/c/en/us/products/collateral/wireless/aironet-1200-access-point/product_data_sheet09186a00800937a6.pdf/, [Accessed December 2015]

[27] R. Murty, J. Padhye, R. Chandra, A. Wolman, B. Zill, “Designing high performance enterprise wi-fi networks,” Proc. 5th USENIX Symposium on Networked Systems Design and Implementation, San Francisco, CA, USA, 16-18 April 2008.

[28] R. Chandra, P. Bahl, “MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card,” in Proc. INFOCOM 2004. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, 7-11 Mar. 2004.

[29] 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.

[30] The CAIDA UCSD equinix-chicago- 20150219-130000, https://data. caida.org/datasets/passive-2015/equinix-chicago/20150219-130000.UTC/equinix-chicago.dirA.20150219-125911.UTC.anon.pcap.gz

[31] 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.

[32] J. Liu, M. Yao, Z. Qiu, “Enhanced Two-Level Frame Aggregation with Optimized Aggregation Level for IEEE 802.11n WLANs,” in IEEE Communications Letters, vol.19, no.12, pp.2254-2257, Dec. 2015.

[33] A. Raschella, F. Bouhafs, J. Saldana, “Wi-5 Deliverable D4.3. Final Specification of Cooperative Functionalities,” April 2018.

[34] D. Monderer and L. S. Shapley, “Potential games,” Games and economic behavior, vol. 14, no. 1, pp. 124–143, 1996.

[35] J. Ortin, J. R. Gallego, and M. Canales, “Joint cell selection and resource allocation games with backhaul constraints,” Pervasive and Mobile Computing, 2016.

[36] V. Mhatre, K. Papagiannaki, Using smart triggers for improved user per- formance in 802.11 wireless networks., in: P. Gunningberg, L. ke Larzon, M. Satyanarayanan, N. Davies (Eds.), MobiSys, ACM, p. 246-259.

[37] 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.

[38] J. Saldana, R. Munilla, S. Eryigit, O. Topal, J. Ruiz-Mas, J. Fernández-Navajas, L. Sequeira, “Unsticking the Wi-Fi Client: Smarter Decisions using a Software Defined Wireless Solution,” under review by IEEE Access, April 2018.

[39] A. Raschellà, F. Bouhafs, M. Seyedebrahimi, M. Mackay, Q. Shi, “Quality of Service Oriented Access Point Selection Framework for Large Wi-Fi Networks”, IEEE Transactions on Network and Service Management, Vol. 14, Issue 2, Pages 441-455, Jun. 2017.

[40] J. Saldana, L. Sequeira, J. Ruiz-Mas, J. Fernández-Navajas, A. Arsal, “What to do With the Wi-Fi Wild West - D3.1, Definition of the performance monitoring mechanisms,” Dec 2016.

[41] 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.

[42] R. Moraes, P. Portugal, F. Vasques, R.F. Custódio, “Assessment of the IEEE 802.11 e EDCA protocol limitations when dealing with real-time communication,” EURASIP Journal on Wireless Communications and Networking, 2010, 29.

[43] J. Saldana, J. Ruiz-Mas, Jose Almodovar, “Frame Aggregation in Central Controlled 802.11 WLANs: the Latency vs. Throughput Trade-off,” in IEEE Communications Letters, vol.21, no. 11, pp. 2500-2530, Nov. 2017. ISSN 1089-7798. doi: 10.1109/LCOMM.2017.2741940.

[44] J. G. Andrews et al., “What Will 5G Be?,” in IEEE Journal on Selected Areas in Communications, vol. 32, no. 6, pp. 1065-1082, June 2014. doi: 10.1109/JSAC.2014.2328098

[45] 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.

Annex 1. Estimating the *distance* in dBs between the Wi-5 Access Points

An application able to estimate the "distance" in dBs between different Wi-5 APs has been developed and tested. Its name is *ShowMatrixOfDistancedBs*[33]. During a time interval, the Wi-5 controller requests an AP to send beacons with a special SSID (*e.g.*, `odin_init`), while the rest of APs listen for them. They can then calculate the average signal level of these special beacon frames, and send a report to the controller. Both operations (sending and receiving) are performed by the auxiliary wireless interface. The aim of this operation is to make this information available to the controller, which can then run the algorithms for channel assignment, in order to minimise the interference between adjacent APs. This can be done in different channels. The scheme is shown in Figure 66, using channel 6 and a scanning time of 5 seconds.

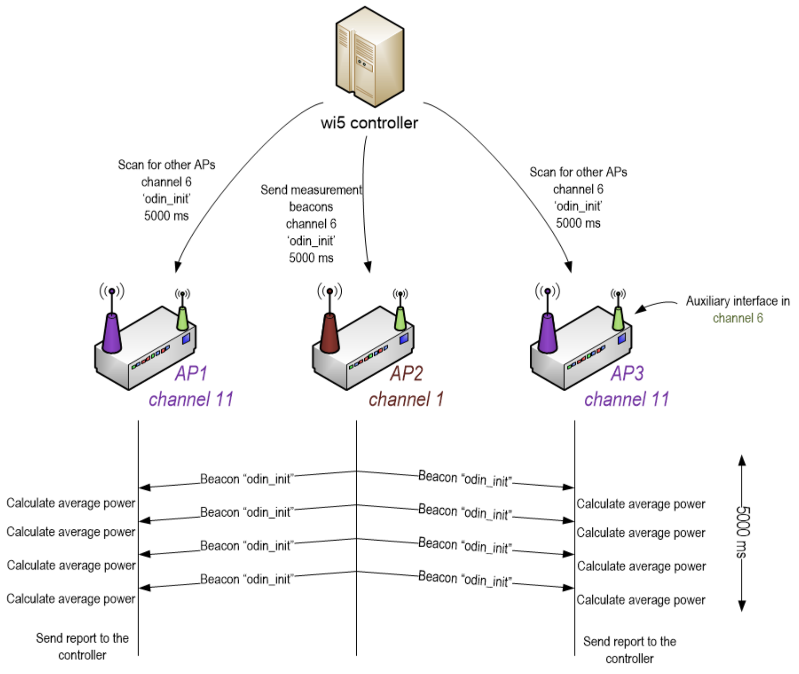

wi5 controller
Scan for other APs channel 6 'odin_init' 5000 ms
Send measurement beacons channel 6 'odin_init' 5000 ms
Scan for other APs channel 6 'odin_init' 5000 ms
Auxiliary interface in channel 6
AP1 channel 11
AP2 channel 1
AP3 channel 11
Beacon "odin_init"
Calculate average power
Send report to the controller
5000 ms

Figure 66: Scheme of the operation of *ShowMatrixOfDistancedBs* application

This is the output of an AP, first sending beacons and then scanning:

```
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Send mesurement beacon (SSID odin_init) in channel 6
[Odinagent.cc] ########### Send mesurement beacon: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Send mesurement beacon: command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Changing scanning flags --> ClientScanningFlag:0 APScanningFlag:0 MesurementBeaconFlag:0
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Scanning for APs (SSID odin_init) in channel 6
[Odinagent.cc] ########### Scanning for APs: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Scanning: Sending AP scanning values

[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Send mesurement beacon (SSID odin_init) in channel 6
[Odinagent.cc] ########### Send mesurement beacon: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Changing scanning flags --> ClientScanningFlag:0 APScanningFlag:0 MesurementBeaconFlag:0
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Scanning for APs (SSID odin_init) in channel 6
```

[33] Wi-5 *ShowMatrixOfDistancedBs* application, see https://github.com/Wi5/odin-wi5/wiki/Application-ShowMatrixOfDistancedBs

```
[Odinagent.cc] ########### Scanning for APs: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Scanning: Sending AP scanning values
```

This is the output of an AP, scanning first and then sending the beacons:

```
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Scanning for APs (SSID odin_init) in channel 6
[Odinagent.cc] ########### Scanning for APs: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Scanning for APs: Setting channel to scan in auxiliary interface
[Odinagent.cc] ########### Scanning: Sending AP scanning values
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Send mesurement beacon (SSID odin_init) in channel 6
[Odinagent.cc] ########### Send mesurement beacon: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Changing scanning flags --> ClientScanningFlag:0 APScanningFlag:0 MesurementBeaconFlag:0

[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Scanning for APs (SSID odin_init) in channel 6
[Odinagent.cc] ########### Scanning for APs: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Scanning: Sending AP scanning values
[Odinagent.cc] ########### Read scanning flags --> ClientScanningFlag: 0 APScanningFlag: 0 MesurementBeaconFlag: 0
[Odinagent.cc] ########### Send mesurement beacon (SSID odin_init) in channel 6
[Odinagent.cc] ########### Send mesurement beacon: Testing command line --> hostapd_cli -i wlan1 chan_switch 0 2437 >
/dev/null
[Odinagent.cc] ########### Changing scanning flags --> ClientScanningFlag:0 APScanningFlag:0 MesurementBeaconFlag:0
```

This is the output of the controller application:

```
[ShowScannedStationsStatistics] Matrix of Distance
[ShowScannedStationsStatistics] ==================
[ShowScannedStationsStatistics]
[ShowMatrixOfDistancedBs] Scanning channel 6
[ShowScannedStationsStatistics]
[ShowMatrixOfDistancedBs] Agent to send mesurement beacon: /192.168.1.13
[ShowMatrixOfDistancedBs] Request for scanning during the interval of  5000 ms in SSID odin_init
[ShowMatrixOfDistancedBs] Agent: /192.168.1.15

[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.15 in channel 6
         AP MAC: DC:EF:09:E6:9C:DB
                  avg signal: -69.3930215965 dBm
[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.14 in channel 6
         AP MAC: DC:EF:09:E6:9C:DB
                  avg signal: -52.4795155218 dBm

[ShowMatrixOfDistancedBs] Agent to send mesurement beacon: /192.168.1.15
[ShowMatrixOfDistancedBs] Request for scanning during the interval of  5000 ms in SSID odin_init
[ShowMatrixOfDistancedBs] Agent: /192.168.1.13

[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.13 in channel 6
         AP MAC: E4:F4:C6:F4:87:EF
                  avg signal: -69.3930215965 dBm
[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.14 in channel 6
         AP MAC: E4:F4:C6:F4:87:EF
                  avg signal: -50 dBm
```

This is another example, in which three APs are used. The output seen by the console of the Wi-5 controller shown below:

```
14:51:37.587 [pool-3-thread-2] INFO  n.f.odin.master.OdinAgent - Sending WRITE_HANDLER_SCANING_FLAGS 0 0 0
[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.13 in channel 6
         AP MAC: DC:EF:09:E6:9E:DF
         avg signal: -50 dBm
[ShowMatrixOfDistancedBs]
[ShowMatrixOfDistancedBs] Agent: /192.168.1.15 in channel 6
         AP MAC: DC:EF:09:E6:9E:DF
         avg signal: -50 dBm

[ShowMatrixOfDistancedBs] ==================
192.168.1.13       ----------         -53.81 dBm         -50 dBm
192.168.1.15       -50.06 dBm         ----------         -50.48 dBm
192.168.1.14       -50 dBm            -50 dBm            ----------
```

The three last lines show the matrix of distances between the three APs. The diagonal is null, as an AP does not measure its distance with itself.

Annex 2. Parameters of SmartAPSelection in poolfile

As has been explained in Section 4, the Wi-5 application SmartAPSelection has four different modes, namely `RSSI`, Fittingness Factor (`FF`), `BALANCER` and `DETECTOR`. In order to make it possible to use them, a number of parameters have been added to the `poolfile`, where the system manager has to define the elements of the system. In this list we detail the function of each of the parameters used by the SmartAPSelection section of the `poolfile`:

- `TimeToStart` (sec): There is an initial time interval after starting the application, which gives the user a margin to start the APs.
- `ScanningInterval` (msec): It is the interval during which the AP scans on each channels, in order to build the RSSI matrix.
- `AddedTime` (msec): Interval after each scan to give time the agent to perform other tasks. For testing, recommended value = 0.
- `SignalThreshold` (dBm): It is used in RSSI, BALANCER and DETECTOR modes. It represents the minimum RSSI value which is considered acceptable for moving a STA to an AP.
- `Hysteresis` (sec): After a handover, a node will not be moved again until this time has passed. Recommended value = 4.
- `Alpha` (0-1): A parameter that gives a different weight to historical RSSI data. Recommended value = 0.8.
- `Pause` (sec): Interval after each assignation to give time the controller to perform other tasks. For testing, recommended value = 0.
- `Mode`: It selects the different modes for running the application. It can be RSSI, FF, BALANCER, DETECTOR.
- `TxpowerSTA` (dBm): Only used in FF mode. It is an estimation of the transmission power of the STAs. The FittingnessFactor algorithm uses it to calculate the available throughput. It may range between 10 and 15 dBm.
- `ThReqSTA` (kbps): Only used in FF mode. It represents the bandwidth required by the STAs. The FittingnessFactor algorithm uses it.
- `Filename`: Name of the file where the log will be stored.

Annex 3. New mechanism for storing information in the AP

In order to obtain detailed statistical values of the frames received by the Odin agents, we have included a new option which can be used by activating the option `CAPTURE_MODE` in the `.cli` file that defines the Agent. If this value is set to `1`, two files will be generated in the Agent, one for each interface (`mon0` and `mon1`) storing Radiotap statistics. Also, using the option `MAC_CAPTURE`, a mask can be defined in order to only monitor some STAs in the wireless interface *e.g.,* `60:E3:27:4F:C7:AA`. To capture all the traffic, `FF:FF:FF:FF:FF:FF` can be used.

This is the format of the file that is obtained:

```
Time(sec);Src;Dst;Bssid;Seq;Rate(Mbs);Signal(mW);
1505918409,463767559;00:1D:E0:BF:86:45;00:1B:B3:BF:86:45;00:80:C8:3C:BF:7B;3104;54;206;
1505918409,480170868;00:1D:E0:BF:86:45;00:1B:B3:BF:86:45;00:80:C8:3C:BF:7B;3105;54;204;
(...)
```

Each row corresponds to a frame with the following parameters:

- `Time` (sec): A timestamp.
- `Src`: Source MAC address.
- `Dst`: Destination MAC address.
- `Bssid`: BSSID.
- `Seq`: Sequence number.
- `Rate` (Mbps): Wireless rate
- `Signal` (dBm): Power level